Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 3799–3821
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

A hybrid machine learning approach to network anomaly detection

Taeshik Shon [a,*], Jongsub Moon [b]

[a] IP Lab, TN R&D Center, Dong Suwon P.O. Box 105, Maetan-3dong, Suwon 442-600, Republic of Korea
[b] CIST/GSIS, Korea University, 5-ga, Anamdong, Seongbukgu, Seoul 136-701, Republic of Korea

Received 16 October 2005; received in revised form 12 March 2007; accepted 14 March 2007

Abstract

Zero-day cyber attacks such as worms and spy-ware are becoming increasingly widespread and dangerous. The existing signature-based intrusion detection mechanisms are often not sufficient in detecting these types of attacks. As a result, anomaly intrusion detection methods have been developed to cope with such attacks. Among the variety of anomaly detection approaches, the Support Vector Machine (SVM) is known to be one of the best machine learning algorithms to classify abnormal behaviors. The soft-margin SVM is one of the well-known basic SVM methods using supervised learning. However, it is not appropriate to use the soft-margin SVM method for detecting novel attacks in Internet traffic since it requires pre-acquired learning information for supervised learning procedure. Such pre-acquired learning information is divided into normal and attack traffic with labels separately. Furthermore, we apply the one-class SVM approach using unsupervised learning for detecting anomalies. This means one-class SVM does not require the labeled information. However, there is downside to using one-class SVM: it is difficult to use the one-class SVM in the real world, due to its high false positive rate. In this paper, we propose a new SVM approach, named *Enhanced SVM*, which combines these two methods in order to provide unsupervised learning and low false alarm capability, similar to that of a supervised SVM approach.

We use the following additional techniques to improve the performance of the proposed approach (referred to as Anomaly Detector using Enhanced SVM): First, we create a profile of normal packets using Self-Organized Feature Map (SOFM), for SVM learning without pre-existing knowledge. Second, we use a packet filtering scheme based on Passive TCP/IP Fingerprinting (PTF), in order to reject incomplete network traffic that either violates the TCP/IP standard or generation policy inside of well-known platforms. Third, a feature selection technique using a Genetic Algorithm (GA) is used for extracting optimized information from raw internet packets. Fourth, we use the flow of packets based on temporal relationships during data preprocessing, for considering the temporal relationships among the inputs used in SVM learning. Lastly, we demonstrate the effectiveness of the Enhanced SVM approach using the above-mentioned techniques, such as SOFM, PTF, and GA on MIT Lincoln Lab datasets, and a live dataset captured from a real network. The experimental results are verified by *m*-fold cross validation, and the proposed approach is compared with real world Network Intrusion Detection Systems (NIDS).

Keywords: Network security; Machine learning; Anomaly attack; Pattern recognition; Genetic algorithm; Intrusion detection

* Corresponding author.
E-mail addresses: 743zh2k@korea.ac.kr, ts.shon@samsung.com (T. Shon).

0020-0255/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2007.03.025

1. Background

The Internet has become a part of our daily life. It is now an essential tool in surviving in the World and aids people in many areas such as business, education and many more. Businesses in particular use the Internet as an important component of their business model. Businesses not only use internet application such as web and e-mail to generate revenue and communicate with their customers, but customers often use machines accessible through the internet to store important and proprietary information. While this may help businesses operate more efficiently, it also makes them extremely vulnerable to attacks, which include stealing data and obstructing the operations of the business. It is common to find environments on the Internet that have few risks in carrying out an attack. These risks are a result of the high degree of anonymity, and lackluster police interest. However, attacks can be easily circumvented by attack prevention schemes. In order to prevent such attacks, multitude of systems has been designed or proposed to thwart Internet-based attacks. Many of the most commonly used systems today are based on attack signatures, which are unique patterns or conditions that may indicate an attack. These systems may seem effective and efficient; however, there are many drawbacks to signature-based Network Intrusion Detection Systems (NIDS). The first problem is that these systems become a single point of failure. If the deployed NIDS system is disabled for any reason, then it often gives the attacker the time needed to compromise the systems and possibly gain a foothold in the network. Moreover if NIDS systems are not properly configured or maintained, they may provide a false sense of security. In addition, research has shown that if the signatures are not sufficiently robust in describing the attack conditions, then simple modifications can be made that will allow the attack to succeed [44]. Besides the robustness of the signature, signature-based intrusion detection systems rely on human intervention to create, test, and deploy the signatures. Thus, it may take hours or days to generate a new signature for an attack which can be too long when dealing with rapidly moving attacks, such as worm propagation. Some effort has been put into automatic signature generation, which does not require human intervention, but these systems are not yet ready for large scale deployment.

In order to solve the problem mentioned above, systems that do not rely on human intervention were invented. These are anomaly detection systems based on machine learning, data mining or statistical algorithms. These systems use a "normal behavior" model for detecting unexpected behavior. These approaches fall under two categories: supervised methods, which make use of pre-existing knowledge, and unsupervised methods, which do not make use of this knowledge. Several efforts to design anomaly detection algorithms using supervised methods have been discussed in [2,7,26,36].

The research of Anderson et al. [2] and Cabrera et al. [7] deal with statistical methods for intrusion detection. Although the two researchers used the same method, the method was used in a different way. The work of Anderson et al. [2] is a comprehensive intrusion-detection system that performs real-time monitoring of user activity on multiple target systems connected via Ethernet. On the other hand, the statistical method of Cabrera et al. [7] is the application of statistical traffic modeling for detecting novel attacks against computer networks. Differing from the other researchers, Wenke Lee's research [26] focuses on theoretical measures for anomaly detection. His work uses several information-theoretic measures, namely, entropy, conditional entropy, relative conditional entropy, information gain, and information cost for anomaly detection. Also, Ryan et al. [36] uses artificial neural networks with supervised learning, based on the back-propagation neural network, called Neural Network Intrusion Detector, was trained in the identification task and tests experimentally on a system of 10 users. In contrast with the other methods used by researchers that are under supervised schemes, unsupervised schemes can create automatically appropriate labels for a given dataset. Anomaly detection methods with unsupervised features are explained in [14,15,32,33,39,46].

Among those unsupervised schemes, MINDS [14] is based on data mining and data clustering methods such as Filtering/Preprocessing/Known attack detection module, Scan detector, Anomaly detection algorithms, and Summarization of attacks using association pattern analysis. MINDS is still developing a suite of data mining techniques to automatically detect attacks against computer networks and systems. Another one of unsupervised methods, the research of Eskin et al. [15] and Portnoy et al. [32] are used to detect anomalous attacks without pre-existing knowledge. In the case of Eskin's research, data elements are mapped to a feature space, which is a typically vector space, in order to process unlabelled data. Also, Portnoy presented a new type of intrusion detection algorithm, which takes a set of unlabeled data as inputs and attempts to find intrusion patterns within the data. The basic idea is that since the intrusions are both not normal, and are rare, they will appear as outliers in the data. Staniford [39] is the author of SPADE for anomaly-based port scan detection in Snort. SPADE utilizes a statistical anomaly detection method with Bayesian probability, and uses a simulated annealing method to cluster anomalous packets together into port scans using heuristics developed from real scans. Ramaswamy et al. [33] uses an outlier calculation with data mining to propose a novel formulation for distance-based outliers, which is based on the distance of a point from its ‘*k*’ and declares the top ‘*n*’ points to be outliers in this ranking calculation.

Irrespective of whether good anomaly detection methods are used, the problems such as high false alarm rates, difficulty in finding proper features, and high performance requirements still exist. Therefore, if we are able to mix the advantages of both learning schemes in machine learning methods, according to their characteristics in the problem domain, then the combined approach can be used as an efficient means for detecting anomalous attacks.

In this paper, we have chosen to focus on the Support Vector Machine (SVM) among various machine learning algorithms. The SVM is already known as the best learning algorithm for binary classification [5,6,12,17,20,23,28,37,40,42]. However, it is not the reason that we have chosen SVM. The most significant reason we chose the SVM is because it can be used for either supervised or unsupervised learning. The SVM, originally a type of pattern classifier based on a statistical learning technique for classification and regression with a variety of kernel functions [12,23,37,42], has been successfully applied to a number of pattern recognition applications [6]. Recently, it has also been applied to inform security for intrusion detection [17,20,40]. Another positive aspect of SVM is that it is useful for finding a global minimum of the actual risk using structural risk minimization, since it can generalize well with kernel tricks even in high-dimensional spaces under little training sample conditions. The SVM can select appropriate setup parameters because it does not depend on traditional empirical risk such as neural networks. In the case of supervised SVM learning, it has relatively fast processing and high detection performance when compared to existing artificial neural networks and the unsupervised SVM, as shown in [13,30]. However, one of the main disadvantages of the supervised method is that it requires labeled information for efficient learning. Moreover, it can not deal with the relationship between consecutive variations of learning inputs without additional preprocessing.

Therefore, we propose an *Enhanced SVM approach* which sits between the standard supervised SVM and the unsupervised SVM as a core component of the hybrid machine learning approach for network anomaly detection. The enhanced SVM approach inherits the advantages of both SVM approaches, namely high performance and unlabelled capability. Also, additional learning and packet filtering techniques are applied to the hybrid machine learning framework such as SOFM for profiling normal packets, PTF for filtering unknown packets, GA for selecting more appropriate packet fields, and considering the relationship of packet sequences in order to support the proposed SVM approach. We expect that the hybrid machine learning approach, based on this Enhanced SVM and supplemented by the preprocessing and testing techniques described in the next section, will demonstrate overall better performance than any previously proposed NIDS.

2. Overall framework

The overall structure of the anomaly detection framework is comprised of four major steps, as depicted in Fig. 1. The first step related to on-line processing includes a real-time traffic filtering using PTF, off-line processing includes data clustering using SOFM, and a packet field selection using GA. We use PTF to drop malformed packets, which by definition are not valid for the network. This offers better performance for packet preprocessing and diminishes the number of potential attacks of the raw traffic. Off-line field selection using GA and packet clustering using SOFM are performed before working the overall framework. GA selects optimized fields in a packet through the natural evolutionary process and the selected fields are applied to filtered packets in real-time. SOFM-based packet clustering is performed off-line to create packet profiles for SVM learning, which in turn are used to make more appropriate training data. In the second step, the filtered-packets are preprocessed for high detection performance. In this data preprocessing step, the packets that passed the first step are preprocessed with packet relationships based on traffic flow for SVM learning inputs. The reason we consider the relationships between the packets is to charge SVM inputs with temporal characteristics during preprocessing, using the concept of a sliding window in

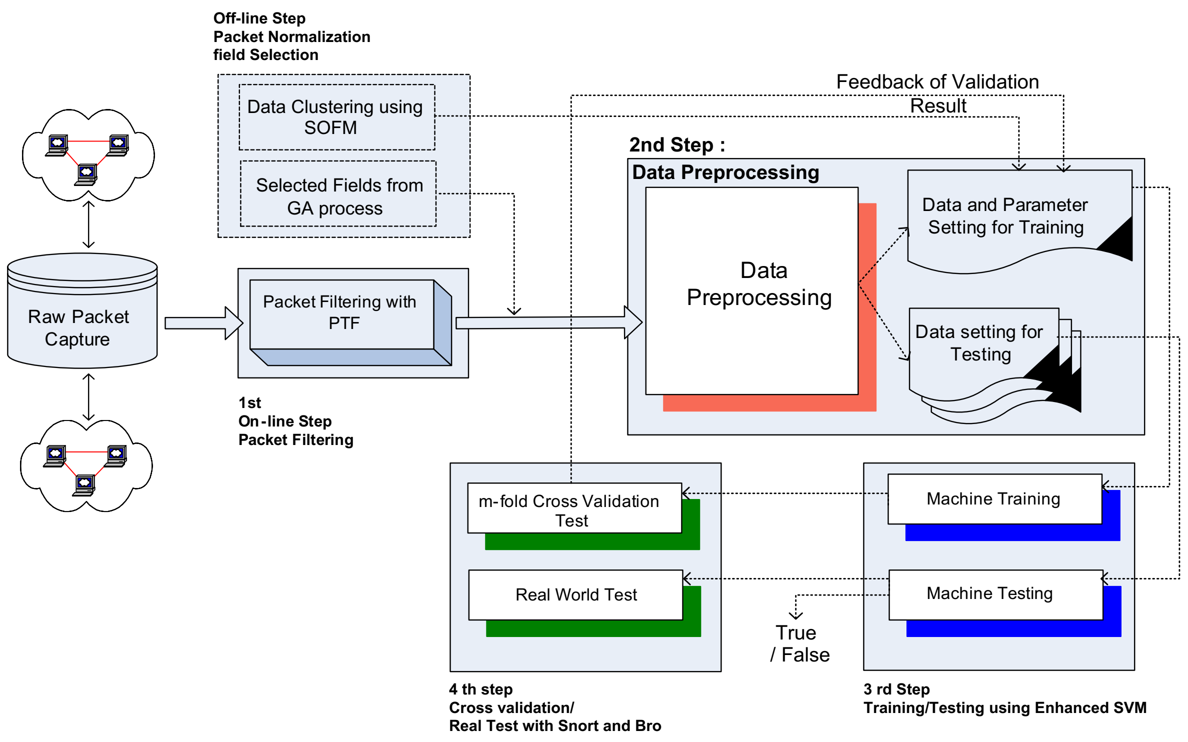

Fig. 1. The Overall Structure of The Proposed Machine Learning Framework.

accordance with the IP identification number. The third step includes the Enhanced SVM machine learning approach (for training and testing), as described in chapter 1. The SVM model combines two kinds of machine learning methods: soft margin SVM (supervised method) and one-class SVM (unsupervised method). In doing so, the Enhanced SVM approach inherits the high performance of soft margin SVM, and the novelty detection capability of one-class SVM. The final step entails verifying the approach using both an *m*-fold cross validation test, and comparing the proposed framework with real world NIDSs, such as Bro [31] and Snort [34].

The rest of this paper is organized as follows. In Section 3, we introduce SOFM, PTF, and GA, which are supplemental techniques employed to enhance the performance of the proposed approach. In Section 4, we present combine supervised and unsupervised SVM methods and the proposed machine learning approach. In Sections 5 and 6, we explain experimental methods along with data preprocessing, data description, and parameter settings. In the last section, we conclude with a summary and discussion of future work.

3. Packet profiling, filtering and field selecting approach

3.1. Packet profiling using SOFM

Collecting normal data is one of the most important aspects in the field of machine learning because this collected data is required for training a supervised learning system. Moreover, the normal data can provide an intelligent criterion to an unsupervised machine learning system. In the proposed machine learning framework, packets for normal learning is needed as a measure of the correct learning. Thus, we have to provide a reasonable rule in order to show the packets used for learning have normal characteristics. However, this is not a simple task; for this reason, a new method is needed, such as a data mining scheme, to construct a normalized group (normal profile) with normal packets (attack-free). If we provide a reasonable method for creating a normal profile, it will enhance the generalized detection capability of the framework. Another tool required in the data mining area is data clustering. In this section, we use the machine learning algorithm

Self-Organized Feature Map (SOFM) which is capable of making clusters as seen in [8,25,43]. The aim is to generate several clusters and distinguish these with normal features.

SOFM is an unsupervised neural network model for analyzing and visualizing high-dimensional data into two dimensional lattices. In other words, SOFM may be defined formally as a non-linear, ordered, smooth mapping of high-dimensional input data manifolds onto the elements of a regular low-dimensional array. A SOFM converts non-linear statistical relationships between data points in high-dimensional space into geometrical relationships between points in two-dimensional map.Also, it differs from competitive learning in that neighboring neurons in SOFM learn to recognize neighboring sections of the input space while it belongs to a family of competitive learning algorithms,. Thus, SOFM learns from both the distribution and topology of the input vectors used for training. The learning procedure of SOFM is similar to that of competitive learning networks. In other words, a similarity measure is selected and the winning unit is considered to be the one with the largest activation. However, for Kohonen feature maps, we update all of the weights, including the winning unit's weights, in the neighborhood surrounding the winning units. Generally the size of neighborhood decreases slowly with each iteration. A sequential description of how to train the Kohonen self-organizing network [25,43] is as follows:

Step 1: Select the winning output unit as the one with the largest similarity measure (or smallest dissimilarity measure) between all weight vectors w_i and input vector x. If the Euclidean distance is chosen as the dissimilarity measure, then the winning unit c satisfies the following equation:

$$\|x - w_c\| = \min_i \|x - w_i\|,$$

where the index c refers to the winning unit and the index i refers to the output unit.

Step 2: Let NB_c denote a set of indices corresponding to a neighborhood around winner c. The weights of the winner and its neighboring units are then updated by

$$\Delta w_i = \eta(x - w_i), i \in \mathrm{NB}_c,$$

where η is the small positive learning rate. Instead of defining the neighborhood of a winning unit, we use a neighborhood function $\Omega_c(i)$ around the winning unit c. For instance, the Gaussian function can be used as the neighborhood function:

$$\Omega_c(i) = \exp\left(\frac{-\|p_i - p_c\|^2}{2\sigma^2}\right),$$

where p_i and p_c are the positions of the output units i and c, respectively, and σ reflects the scope of the neighborhood. By using the neighborhood function, the update formula can be rewritten as

$$\Delta w_i = \eta\Omega_c(i)(x - w_i),$$

where i is the index for all output units. To achieve better convergence, the learning rate η and the size of neighborhood (or σ) should be decreased gradually each iteration.

When we provide high-dimensional input described above, SOFM can use topology and distance to transform it into two-dimensional space and generate the various clusters. In the problem domain, a variety of SOFM clusters will be made in accordance with the attributes of packets. At this point in the process, if we label the generated clusters using a statistical method or a verification procedure (distinguishing normal from abnormal by off-line analysis), then some of the clusters would be used as a *normal* packet profile, or a profile of packets exhibiting normal packet features. Not only that, an unsupervised clustering using SOFM does not require any other measures to make clusters because it only relies on the original characteristics of the given inputs. Therefore, SOFM can act as a simple and efficient method for defining normal packet classes.

3.2. Packet filtering using Passive TCP/IP Fingerprinting (PTF)

TCP/IP Fingerprinting is the process of determining the identity of a remote host's operating system by using unique quirks found in the TCP/IP stack of each OS [18]. This method can be divided into active schemes and passive schemes depending on whether probe packets are sent by the host. Active schemes such as Xprobe [3] and Nmap [16] are the predominant OS fingerprinting tools used today. They operate by sending a sequence of packets to the target and analyzing the returned packets for behavior or values known to occur on certain systems. As with any heuristic approach, this method requires updating, however, a huge amount of information can be gleaned from carefully constructed packets. The passive schemes are similar to active fingerprinting methods in that they analyze and correlate packets according to the OS. However, these schemes operate in a different way. First, passive fingerprinting utilizes a packet sniffer. This means that it is set up on a network and passively collects/analyses packets between hosts. Thus, the passive fingerprinting obviously has limited function in a switched network; however, it may be useful if it is attached in a shared network space, with a router or well-utilized network host. PTF uses some of the same techniques as active fingerprinting, but waits for the information to return. Tools like p0f [47] have multiple modes that can be used to determine the OS for outgoing connections, incoming connections, or established connections.

The packet filter is used to confirm the structure of received packets. In other words, if a packet is recognized as not using the constructions of well-known platforms, (some of which follow the appropriate RFCs), then it is discarded. Filtering based on a strict standard compliance is not an option for systems that often bend or break the rules, such as Microsoft Windows. As mentioned before, fingerprinting uses heuristics, and thus can be expected to correctly identify normal systems as long as they are common enough, or a fingerprint about the system is submitted. Allowing a packet through this initial checking procedure does not mean that it is safe, but it means only more likely to be a normally generated packet. By using this initial determination of normal behavior, we expect to decrease many crafted packet types, such as covert channels and some active scanning probes.

3.3. Field selection using genetic algorithm (GA)

Genetic Algorithm (GA) is a model used to mimic the behavior of the evolutionary processes in nature [19,29,45]. It is known to be an ideal technique for finding solutions for optimization problems. GA uses three operators to produce the next generation from the current: reproduction, crossover, and mutation. The reproduction operator determines which individuals are chosen for crossover and how many offspring each selected individual produces. The selection uses a probabilistic survival of the fittest mechanism based on a problem-specific evaluation of the individuals. The crossover operator then generates new chromosomes within the population, by exchanging randomly selected segments from existing chromosomes. Finally, the mutation operator allows the random mutating of existing chromosomes so that new chromosomes may contain parts not found in any existing chromosomes. This whole process is repeated probabilistically, moving from generation to generation with the expectation that eventually we are able to choose the individual that closely matches the desired conditions. When the process terminates, the best chromosome selected from among the final generation is the solution. To apply this "process of evolution" to the problem domain, we have to first define the following three components: individual gene presentation and initialization, evaluation function modeling, and a specific function of genetic operators and their parameters.

For the first operation, we transform TCP/IP packets into binary gene strings. We convert each TCP and IP header field into a bit binary gene value, '0' or '1'. '1' means that the corresponding field exists and '0' means it does not. The initial population consists of a set of randomly generated 24-bit strings, including 13 bits for IP fields and 11 bits for TCP fields. The total number of individuals in the population should be carefully considered. If the population size is too small, then all gene chromosomes soon converge into the same gene string, making it impossible for the genetic model to generate new individuals. In contrast, if the population size is too large, then the model spends too much time calculating gene strings, negatively affecting the overall effectiveness of the method.

In addressing the second process, we create a fitness function for evaluating individuals. The fitness function consists of an object function $f(X)$ and its transformation function $g(f(X))$

$$F(X) = g(f(X)). \tag{1}$$

In Eq. (1), the object function's values are converted into a measure of relative fitness by fitness function $F(X)$, with transformation function $g(x)$. To create the own objective function, we use the anomaly and communication scores shown in Table 1.

The anomaly score refers to the attack packets of MIT Lincoln Lab datasets, covert channels, and other anomaly attacks [1,9,27,35,38]. The scores increase in proportion to the frequency of a field being used for anomaly attacks. In the case of communication scores, they are divided into three kinds of scores according to their importance during communication. The three kinds of score are as following. 'S' fields have static values, "De" field values are dependent on connection status, and field values for "Dy" can change dynamically. From these scores, we can derive a polynomial equation which uses the above-mentioned considerations as a coefficient. For the combination of an anomaly score and a communication score, the coefficients of this derived polynomial equation have the property of being weighted sums. The objective function $f(X)$ consists of two polynomial functions, $A(X)$ and $N(X)$, as shown in (2).

$$f(X) = A(X) + N(X), \tag{2}$$

$A(X)$ is the anomaly scoring function and $N(X)$ is the communication scoring function. $X(x)$ is an individual with 24 attributes and x_i is one of the individual's attributes with an attribute's value a_i (as illustrated in Table 1). To prevent generating too many features from (2)'s fitness function, a bias term μ is used as follows:

$$f'(X(x)) = f(X(x)) - \mu = A(X(x)) + N(X(x)) - \mu. \tag{3}$$

Here, μ is the bias term of the new objective function $f'(X(x))$, and the boundary is $0 < \mu < \mathrm{Max}(f(X))$. In the case of $A(X(x))$, we can derive the proper equation as follows:

$$A(X) = A(X(x)) = a_i x_i + \cdots + a_2 x_2 + a_1 x_1, \quad i = \{1, \ldots, 24\}, \tag{4}$$

where, x_i denotes the ith attribute. $\{a_i, \ldots, a_2, a_1\}$ is a set of coefficients in the polynomial equation (as illustrated in Table 1), and each coefficients represents anomaly scores. From Eq. (4), we use the bias term to

Table 1
TCP/IP anomaly and communication score

Index number	Name of coefficients	Anomaly score[a]	Communication score[b]
01	x_{01} (Version)	0	S
02	x_{02} (Header length)	0	De
03	x_{03} (Type of service)	0	S
04	x_{04} (Total length)	0	De
05	x_{05} (Identification)	2	Dy
06	x_{06} (Flags)	5	Dy
07	x_{07} (Fragment offset)	5	Dy
08	x_{08} (Time to live)	1	Dy
09	x_{09} (Protocol)	1	S
10	x_{10} (Header checksum)	0	De
11	x_{11} (Source address)	2	S
12	x_{12} (Destination address)	1	S
13	x_{13} (Options)	1	S
14	x_{14} (Source port)	1	S
15	x_{15} (Destination port)	1	S
16	x_{16} (Sequence number)	2	Dy
17	x_{17} (Acknowledge number)	2	Dy
18	x_{18} (Offset)	1	Dy
19	x_{19} (Reserved)	1	S
20	x_{20} (Flags)	2	Dy
21	x_{21} (Window)	0	S
22	x_{22} (Checksum)	0	De
23	x_{23} (Urgent pointer)	1	S
24	x_{24} (Options)	1	S

[a] By anomaly analysis in [1,9,28,36,39].
[b] S: static, De: dependent, Dy: dynamic.

satisfy the condition (5). Thus, we can choose a reasonable number of features without over-fitting and can derive the new anomaly scoring function (6) with the bias term μ_A as follows:

$$\begin{aligned} A(X) &= a_i x_i + \cdots + a_2 x_2 + a_1 x_1 \\ &< \mathrm{Max}(A(X)) \end{aligned} \tag{5}$$

$$\begin{aligned} &A'(X) = (a_i x_i + \cdots + a_2 x_2 + a_1 x_1) - \mu_A, \quad 0 < \mu_A < \mathrm{Max}(A(X)) \\ &\therefore \quad 0 < A'(X) < \mathrm{Max}(A(X)). \end{aligned} \tag{6}$$

For $N(X(x))$, we also develop an appropriate function with the same derivation as Eq. (6).

$$\begin{aligned} N(X) &= N(X(x)), \quad \alpha = 1,\ \beta = 2,\ \gamma = 3,\ i = \{1, \ldots, 24\} \\ &= a_i x_i + \cdots + a_2 x_2 + a_1 x_1 \\ &= (a_1 x_1 + a_3 x_3 + a_9 x_9 + a_{11} x_{11} + a_{12} x_{12} + a_{13} x_{13} \\ &\quad + a_{14} x_{14} + a_{15} x_{15} + a_{19} x_{19} + a_{21} x_{21} + a_{23} x_{23} + a_{24} x_{24}) \\ &\quad + (a_2 x_2 + a_4 x_4 + a_{10} x_{10} + a_{22} x_{22}) + (a_5 x_5 + a_6 x_6 + a_7 x_7 \\ &\quad + a_8 x_8 + a_{16} x_{16} + a_{17} x_{17} + a_{18} x_{18} + a_{20} x_{20}) \\ &= \alpha(x_1 + x_3 + x_9 + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{19} + x_{21} + x_{23} + x_{24}) \\ &\quad + \beta(x_2 + x_4 + x_{10} + x_{22}) + \gamma(x_5 + x_6 + x_7 + x_8 + x_{16} + x_{17} + x_{18} + x_{20}), \end{aligned} \tag{7}$$

where N is a set of communication scores, and the coefficients α, β, γ mean static (S), dependent (De), and dynamic (Dy), respectively (as illustrated in Table 1). In Eq. (7), we give the bias term using the same method as (5) and (6), as follows:

$$\begin{aligned} N(X) &= \alpha(x_\alpha) + \beta(x_\beta) + \gamma(x_\gamma) \\ &< \mathrm{Max}(N(X)) \end{aligned} \tag{8}$$

$$\begin{aligned} &N'(X) = \alpha(x_\alpha) + \beta(x_\beta) + \gamma(x_\gamma) - \mu_N, 0 < \mu_n < \mathrm{Max}(N(X)) \\ &\therefore \quad 0 < N'(X) < \mathrm{Max}(N(X)), \end{aligned} \tag{9}$$

where x_α, x_β and x_γ are a set of elements with the coefficient α, β and γ, respectively. From Eqs. (6) and (9), we can derive the entire objective Eq. (10) as follows:

$$\begin{aligned} f'(X(x)) = A'(X) + N'(X) &= (a_i x_i + \cdots + a_2 x_2 + a_1 x_1) - \mu_A + \alpha(x_\alpha) + \beta(x_\beta) + \gamma(x_\gamma) - \mu_N \\ &= (a_i x_i + \cdots + a_2 x_2 + a_1 x_1) + \alpha(x_\alpha) + \beta(x_\beta) + \gamma(x_\gamma) - (\mu_A + \mu_N) \\ &= (a_i x_i + \cdots + a_2 x_2 + a_1 x_1) + \alpha(x_\alpha) + \beta(x_\beta) + \gamma(x_\gamma) - \mu, 0 < f'(X(x)) < \mathrm{Max}(f(X(x))). \end{aligned} \tag{10}$$

The relative fitness is calculated using the proposed objective function (10) and is converted into normalized fitness function $F(x_k)$ using a rank-based transformation function. This rank-based transformation overcomes the scaling problems of the proportional fitness assignment. The reproductive range is limited so that no small group of individuals generates too many offspring. The ranking method introduces a uniform scaling across the population.

The last step for genetic modeling is to decide a specific function of genetic operators and their related parameters. In the reproduction operator, a roulette wheel method is used. Each individual has their own selection probability, by means of n roulette. A roulette wheel contains one sector per member of the population, which is proportional to the value $P_{\mathrm{sel}}(i)$. If the selection probability is high, this indicates that more gene strings are inherited for the next generation. For crossover, a single crossover point method is used. This method has just one crossover point, so a binary string from the beginning of the chromosome to the crossover point is copied from the first parent; the rest is copied from the second parent. If we use a low crossover probability, this prevents convergence to an optimized solution. Conversely, if the probability is too high, it increases the possibility of destroying the best solution because of the frequent gene exchange. In mutation, we use a general discrete mutation operator. If the mutation probability is too small, then new characteristics will be accepted too late. If the probability is too high, new mutated generations do not have a close relationship with former generations. In Section 6, we construct preliminary tests to determine the best parameters for the problem domain.

4. An enhanced support vector machine approach

We introduce two existing SVM methods: soft margin SVM and one-class SVM. The SVM is generally used as a supervised learning method. In order to decrease misclassified data, a supervised SVM approach with a slack variable is called soft margin SVM. Additionally, single class learning for classifying outliers can be used as an unsupervised SVM. After considering both SVM learning schemes, we propose an Enhanced SVM approach.

4.1. Existing approaches

4.1.1. Soft Margin SVM: Supervised SVM

We begin by discussing a soft margin SVM learning algorithm written by Cortes [12], which is sometimes called c-SVM. This SVM classifier has a slack variable and penalty function for solving non-separable problems. First, given a set of points $x_i \in R^d, i = 1, \ldots, l$ and each point x_i belongs to either of two classes with the label $y_i \in \{-1, 1\}$. These two classes can be applied to anomaly attack detection with the positive class representing normal and negative class representing abnormal. Suppose there exists a hyper-plane $w^T x_i + b = 0$ that separates the positive examples from the negative examples. That is, all the training examples satisfy:

$$\begin{aligned} w^T x_i + b &\geqslant +1 \quad \text{for all } x_i \in P, \\ w^T x_i + b &\leqslant -1 \quad \text{for all } x_i \in N. \end{aligned} \tag{11}$$

w is an adjustable weight vector, x_i is the input vector and b is the bias term.

Equivalently:

$$y_i(w^T x_i + b) \geqslant 1 \quad \text{for all } i = 1, \ldots, N. \tag{12}$$

In this case, we say the set is linearly separable.

In Fig. 2, the distance between the hyper-plane and $f(x)$ is $\frac{1}{\|w\|}$. The margin of the separating hyper-plane is defined to be $\frac{2}{\|w\|}$. Hence, the learning problem is reformulated as minimizing $\|w\|^2 = w^T w$ is subject to the constraints of linear separation shown in Eq. (13). This is equivalent to maximizing the hyper-plane distance between the two classes, of which the maximum distance is called the support vector. The optimization is now a convex quadratic programming problem.

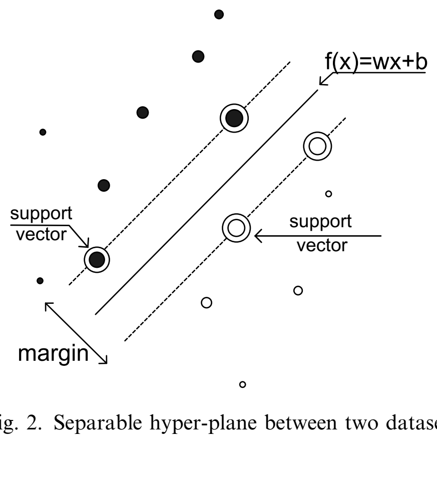

Fig. 2. Separable hyper-plane between two datasets.

$$\begin{aligned} &\text{Minimize}_{w,b} \quad \Phi(w) = \frac{1}{2}\|w\|^2 \\ &\text{subject to} \quad y_i(w^{\mathrm{T}}x_i + b) \geqslant 1, \quad i = 1, \ldots, l. \end{aligned} \tag{13}$$

Since $\Phi(w) = \frac{1}{2}\|w\|^2$ is convex in w and the constraints are linear in w and b, we can be sure that this problem has a global optimum solution. This has the advantage that parameters in quadratic programming (QP) affect only the training time, and not the quality of the solution. This problem is tractable, but anomalies in internet traffic show characteristics of non-linearity, and as a result they are more difficult to classify. In order to proceed to such non-separable and non-linear cases, it is useful to consider the dual problem as outlined below.

The Lagrange for this problem is

$$L(w, b, \Lambda) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \lambda_i \left[y_i(w^{\mathrm{T}}x_i + b) - 1\right] \tag{14}$$

where $\Lambda = (\lambda_1, \ldots, \lambda_l)^{\mathrm{T}}$ are the Lagrange multipliers, one for each data point. The solution to this quadratic programming problem is given by maximizing L with respect to $\Lambda \geqslant 0$ and minimizing with $\frac{1}{2}\|w\|^2$ respect to w and b. Note that the Lagrange multipliers are only non-zero when $y_i(w^{\mathrm{T}}x_i + b) = 1$, and the vectors for this case are called *support vectors,* since they lie closest to the separating hyper-plane. However, in the non-separable case, forcing zero training error leads to poor generalization. We introduce the soft margin SVM using a vector of slack variables $\Xi = (\xi_1, \ldots, \xi_l)^{\mathrm{T}}$ that measure the amount of violation of the constraints (15), taking into account the fact that some data points may be misclassified.

The equation is now

$$\begin{aligned} &\text{Minimize}_{w,b,\Xi} \quad \Phi(w, b, \Xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i^k \\ &\text{subject to} \quad y_i(w^{\mathrm{T}}\phi(x_i) + b) \geqslant 1 - \xi_i, \xi_i \geqslant 0, \quad i = 1, \ldots, l, \end{aligned} \tag{15}$$

where C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error. The effects of C are crucial. If C is too small, insufficient stress is placed on fitting the training data. If C is too large, the algorithm will overfit the dataset.

In practice, a typical SVM approach, such as the soft margin SVM, showed excellent performance more often than other machine learning methods [13,23]. For intrusion detection applications, supervised machine learning approaches based on SVM methods proved superior to intrusion detection approaches using artificial neural networks [13,30,40]. Therefore, the high classification capability and processing performance of the soft margin SVM approach is useful for anomaly detection. However, because the soft margin SVM is a supervised learning approach, it requires datasets to be labeled.

4.2. One-class SVM: Unsupervised SVM

SVM algorithms can also be adapted into an unsupervised learning algorithm called one-class SVM, which identifies outliers among positive examples and uses them as negative examples [37]. In anomaly detection, if we consider anomalies as outliers, then the one-class SVM approach can be applied to classify anomalous packets as outliers.

Fig. 3 shows the relation between a hyper-plane of one-class SVM and outliers. Suppose that a dataset has a probability distribution P in the feature space, and we want to estimate a subset S of the feature space so the probability that a test point drawn from P lies outside of S, which is bounded by some a priori specified value $v \in (0, 1)$. The solution is obtained by estimating the function f, which is a positive function taking the value $+1$ in a small region where most of the data lies and -1 elsewhere (16)

$$f(x) = \begin{cases} +1, & \text{if } x \in S, \\ -1, & \text{if } x \in \overline{S}. \end{cases} \tag{16}$$

The main idea is that the algorithm maps the data into a feature space H using an appropriate kernel function, and then attempts to find the hyper-plane that separates the mapped vectors from the origin with maximum

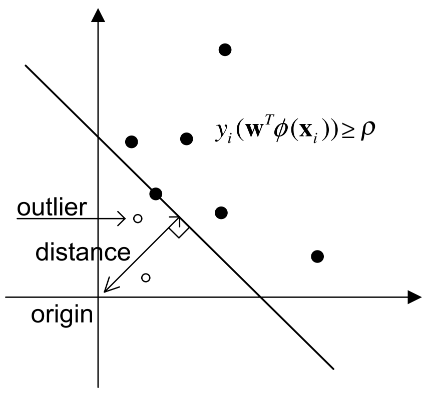

Fig. 3. One-class SVM.

margin. Given a training dataset $(x_1, y_1), \ldots, (x_1, y_1) \in \Re^N \times \{\pm 1\}$, let $\Phi : \Re^N \to H$ be a kernel map that transforms the training examples into the feature space H. Then, to separate the dataset from the origin, we need to solve the following quadratic programming problem (17):

$$\begin{aligned} &\text{Minimize}_{w,b,\Xi} && \Phi(w,b,\Xi) = \frac{1}{2}\|w\|^2 + \frac{1}{vl}\sum_{i=1}^{l}\xi_i^k - \rho, \\ &\text{subject to} && y_i(w^{\mathrm{T}}\phi(x_i)) \geqslant \rho - \xi_i, \xi_i \geqslant 0, \quad i = 1,\ldots,l, \end{aligned} \tag{17}$$

where v is a parameter that controls the trade-off between maximizing the distance from the origin and containing most of the data in the region related to the hyper-plane, and corresponding to the ratio of outliers in the training set. Then the decision function $f(x) = sgn((w \cdot \Phi(x) + b) - \rho)$ will be positive for most examples x_i contained in the training set.

In practice, even though the one-class SVM has the capability of outlier detection, the approach is more sensitive to a given dataset than other machine learning schemes [30,37]. This means that it is very important to decide on an appropriate hyper-plane for classifying outliers.

4.2.1. Proposed enhanced SVM: supervised and unsupervised SVM

In the problem domain, most of the Internet traffic exhibits normal features, and the amount of anomalous traffic is relatively small. To put it simple, we can see that the number of outliers we hope to detect is relatively small in comparison with normal traffic. Therefore, from this point of view, the one-class SVM need not always make the single classifier to maximize the distance between the origin and outliers. We propose an enhanced SVM method with the capability of one-class SVM through modifying soft margin SVM. The enhanced SVM approach can be run on the soft margin SVM, and the proposed SVM approach will not only have higher detection rate and faster processing performance than the soft-margin SVM but also possess the unsupervised learning feature of one-class SVM.

In one-class SVM learning (17), the important parameters for deciding the hyper-plane are $\|w\|, \rho, \frac{1}{vl}\sum_{i=1}^{l}\xi_i^k$. Each parameter has the following meaning: $\|w\|$ has to be decreased and ρ has to be increased in order to obtain maximum margin between the origin and hyper-plane. In the case of $\frac{1}{vl}\sum_{i=1}^{l}\xi_i^k$, this parameter is related to the violation of outliers and has to be decreased. Moreover, in soft margin SVM learning (15), the bias term is generally related to the distance between the origin and hyper-plane. As the bias term is decreases, the hyper-plane approaches the origin similar to the behavior of the one-class SVM classifier. The bias term b of (15) is deleted, and then we derived Eq. (18) as follows:

$$\begin{aligned} &\text{Minimize}_{w,b,\Xi} && \Phi(w,b,\Xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i^k, \\ &\text{subject to} && y_i(w^{\mathrm{T}}\phi(x_i)) \geqslant 1 - \xi_i, \xi_i \geqslant 0, \quad i = 1,\ldots,l. \end{aligned} \tag{18}$$

Therefore, by considering these parameters of one-class SVM and soft margin SVM, we derive an *Enhanced* SVM that provides the unlabeled classification capability of one-class SVM as well as the high detection performance of supervised soft margin SVM.

If we compare (18) with (17),

$$\text{soft margin SVM without a bias } \cong \text{one-class SVM}$$

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i^k \cong \frac{1}{2}\|w\|^2 + \frac{1}{vl}\sum_{i=1}^{l}\xi_i^k - \rho \tag{19}$$

$$y_i(w^{\mathrm{T}}\phi(x_i)) \geqslant 1 - \xi_i \cong y_i(w^{\mathrm{T}}\phi(x_i)) \geqslant \rho - \xi, \quad 0 < v < 1, 1 < l, 0 \leqslant \rho \tag{20}$$

In the minimization condition (19), $C\sum_{i=1}^{l}\xi_i^k$ of soft margin SVM is the trade-off value to adjust the training error to obtain maximum margin between the origin and the hyper-plane. $\frac{1}{vl}\sum_{i=1}^{l}\xi_i^k$ of one-class SVM is also related to the extent of violation of outliers. Thus, we can approximate each term as a similar value by controlling C of soft margin SVM. In the subject condition (20), the ρ of the one-class SVM is a parameter for maximizing the margin between the origin and the hyper-plane. However, we do not need to consider maximizing the ρ value, because anomalous data of the domain can be classified by the hyper-plane near the origin. In the end, we can regard the value of ρ as a very small number like ‘1’. We finally derive an Enhanced SVM, by combining all these parameter approximations in Eq. (21).

$$\begin{aligned} &\text{Minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i^k - \rho, \quad C \cong \frac{1}{vl}, \\ &\text{subject to} \quad y_i(w^{\mathrm{T}}\phi(x_i)) \geqslant \rho - \xi, \quad 0 \leqslant \rho \cong 1. \end{aligned} \tag{21}$$

The *minimization* condition and *subject* condition of (21) have to satisfy the approximated conditions of $C \cong \frac{1}{vl}$ and $\rho \cong 1$, respectively. Therefore, in the proposed SVM approach (21), we can expect the unlabeled learning feature of one-class SVM, the relatively low false alarm and high detection rate of soft margin SVM.

5. Experiment methods

The experiments use the IDS evaluation dataset of the MIT Lincoln Lab [1] and the known SVM toolkits [38,39]. We concentrate on novel attack detection in TCP/IP traffic because TCP/IP based network traffic occupies about 95% of all network traffic [11,41]. Furthermore, SVM input is basically limited in that they do not take into account temporal event sequences. In order to overcome this limitation, we propose a data preprocessing method based on packet flow. Finally, we describe the MIT Lincoln Lab dataset, SVM setup parameters, and kernel functions.

5.1. Data preprocessing

In the proposed framework, the pre-designated fields in packets filtered by PTF are used as SVM input data. At this stage of processing, time variation between packets is not considered because SVM can not generally distinguish between temporal relationships of the SVM inputs. However, if we pick out their relationships among the sequences of network packets, then the knowledge would deeply influence the performance of SVM learning. In order to consider the relationships between packets, we used the concept of a sliding window, using IP identification number during data preprocessing. Each packet was classified by its IP identification number within the same connection. Moreover, the packets with the same IP identification number were rearranged according to the size of the sliding window. In Fig. 4, Case 1 shows that each packet is preprocessed in order of its arrival within the sequence. In contrast, the packets in Case 2 are rearranged by ID number without considering their original flow.

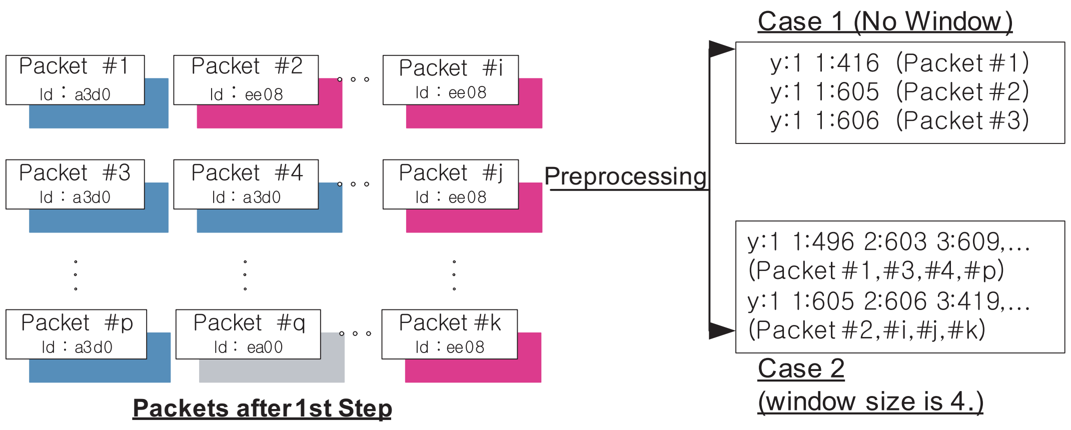

Fig. 4. Data Preprocessing based on packet flow with the same ID field.

5.2. Data description

The 1999 DARPA IDS dataset was collected at the MIT Lincoln Lab to evaluate intrusion detection systems, and the dataset contains a wide variety of intrusions simulated in a military network environment. All the network traffic including the entire payload of each packet was recorded in tcpdump [21] format and was provided for evaluation. The data consists of three weeks of training data and two weeks of test data. For this dataset, we made attack-free training data for normal behavior modeling and attack data for the testing and construction of the anomaly scores shown in Table 1. Real network data was used in comparison tests with real NIDS. Also, the attack datasets include our own generated attacks such as covert channels, malformed packets, and some Denial of Service attacks. The simulated attacks include one of following five categories and consist of DARPA attacks and generated attacks:

- Denial of Service: Apache2, arppoison, Back, Crashiis, dosnuke, Land, Mailbomb, SYN Flood, Smurf, sshprocesstable, Syslogd, tcpreset, Teardrop, Udpstorm.
- Scanning: insidesniffer, Ipsweep, Mscan, Nmap, queso, resetscan, satan, saint.
- Covert Channel: ICMP covert channel, Http covert channel, IP ID covert channel, TCP SEQ and ACK covert channel, DNS tunnel).
- Remote Attacks: Dictionary, Ftpwrite, Guest, Imap, Named, ncftp, netbus, netcat, Phf ppmacro, Sendmail sshtrojan Xlock Xsnoop.
- Forged Packets: Targa3.

In order to make the dataset more realistic, we organized many of the attacks so that the resulting data set consisted of 1–1.5% attacks and 98.5–99% normal traffic. For the supervised learning algorithm, we used a learning dataset with the characteristics described above. This dataset has 100,000 normal data packets and 1000–1500 abnormal packets for training and evaluation. In the case of unsupervised learning methods such as Enhanced SVM and one-class SVM, the dataset contains 100,000 normal packets for training and various normal and abnormal packets with about 1000–1500 packets for evaluation. The training dataset uses only normal traffic because these methods have the capability which can learn without pre-acquired information differently with supervised learning. Then, the combined dataset with normal packets and abnormal packets is used for test experiments. In the SVM experiments, we used the supervised SVM method using the SVM-light package [24] and the unsupervised SVM method using the Libsvm tool [10].

5.3. Parameter setup

The SVM has a variety of kernel functions and associated parameters. In the case of soft margin SVM, we have to set a regularization parameter C. The kernel function transforms a given set of vectors to a possibly higher dimensional space for linear separation. For the parameter selection of the SVM learning, we referred to the experimental results of the SVM [22] which include a range of basic SVM parameters and various kernel functions and performance arguments. In the experiments, we selected parameters as follows: C from 0.9 to 10, d in a polynomial kernel as 1, σ in a radial basis kernel as 0.0001 and κ and θ in a sigmoid kernel as 0.00001, respectively. The SVM kernel functions we consider are linear, polynomial, radial basis, and sigmoid, as follows:

Innerproduct: $K(x,y) = x \cdot y$

Polynomial with deg d: $K(x,y) = (x^{\mathrm{T}}y + 1)^d$

Radial basis with width σ: $K(x,y) = \exp\left(\dfrac{-||x-y||^2}{2\sigma^2}\right)$

Sigmoid with parameter κ and θ: $K(x,y) = \tan h(\kappa x^{\mathrm{T}}y + \theta)$

6. Experimental results and analysis

In this section we detail the overall results of the proposed framework. To evaluate the approaches, we used three performance indicators from intrusion detection research. The *detection rate* is defined as the number of correctly classified normal and abnormal packets divided by the total number of the test data. The *false positive rate* is defined as the total number of normal data, which were incorrectly classified as attacks, divided by the total number of normal data. The *false negative rate* is defined as the total number of attack data that were incorrectly classified as normal traffic divided by the total number of attack data.

6.1. Packet filtering, field selection, and packet profiling

We used the p0f tool for packet filtering since it is one of the most well developed and respected systems available. This tool uses 235 inbound and 103 outbound traffic signatures to identify remote OS intrusions, and some other attacks, such as Nmap scans. The signatures cover all major operating systems, some network devices, and packet filters. The structure of p0f fingerprints as well as detailed descriptions of each field can be found in the fingerprint databases (p0f.fp) included with the p0f tool. A brief description of the fields is shown here:

- Window size (WSS) – maximum amount of data to be sent without ACK.
- Overall packet size – a function of all IP and TCP options and bugs.
- Initial TTL – estimate based on proximity to well known OS defaults.
- Do not fragment flag (DF) – field set by some modern OS's to implement PMTU discovery.
- Maximum segment size (MSS) – field used by p0f to determine link type of the remote host.
- Window scaling (WSCALE) – feature used to scale WSS.
- Timestamp – in some systems, a field set to zero in the initial SYN.
- Selective ACK permitted – a flag set by systems that implement selective ACK functionality.
- NOP option – its presence, count and sequence.
- Other and unrecognized options (TCP-related and such) – implemented by some eccentric or very buggy TCP/IP stacks.
- The sequence of all TCP options as a whole.

In practice, the aim of packet filtering was to exclude malformed packets from raw traffic. However, in this experiment, we used labeled datasets [27] in order to verify the efficiency of the packet filtering approach. In the case of a normal dataset, each dataset consists of collected normal packets from the first Monday, Wednesday, the third Monday, and Wednesday of the MIT dataset. In case of abnormal dataset, abnormal #1 means attack packets using TCP/IP covert channels, abnormal #2 means DoS attacks, and abnormal #3 consists of malformed TCP/IP packets that violate the published TCP/IP standard. In Table 2, the best example of nor-

Table 2
Packet filtering results using passive TCP/IP fingerprinting

	No. of packets	Inbound filtering	Outbound filtering	No. of filtered packets	Ratio (%)
Normal #1	1,369,134	1,249,924	39,804	1,289,728	94.20
Normal #2	1,574,274	1,435,838	46,374	1,482,212	94.20
Normal #3	1,571,598	1,439,880	44,071	1,483,951	94.40
Normal #4	1,783,444	1,589,337	65,035	1,654,372	92.80
Total	6,298,450	5,714,979	195,284	5,910,263	Ave : 93.84 SD : 0.74
Abnormal #1	3145	1691	0	1691	53.80
Abnormal #2	30,018	0	15,008	15,008	50.00
Abnormal #3	836	105	158	263	30.70
Total	34,089	1,796	15,166	16,962	Ave: 49.76 SD: 12.34

Ave: Average, SD: Standard deviation.

mal traffic filtering was the Normal #3 dataset, with a 94.40% filtering rate. About 6% of normal traffic was dropped because the packets did not follow TCP/IP standards or the packet generation methods of the well-known TCP/IP stack. The best case of abnormal datasets was Abnormal #2 with a 30.70% filtering rate; a low filtering rate means a lot of abnormal traffic was dropped. In this case, about 70% packets were dropped, indicating that the majority of the abnormal traffic did not follow typical packet generation schemes of well-know platforms.

Next, we discuss a field selection scheme using GA. In order to find reasonable genetic parameters, we made preliminary tests using the typical genetic parameter values mentioned in the literature [29]. Table 3 describes the four-time preliminary test parameters. The final fitness value refers to the last resultant value calculated by the proposed objective equation.

Fig. 5 shows four graphs of GA feature selection using the fitness function from Eq. (21) and the parameters shown in Table 5. In Case #1 and Case #2, the values converge rapidly because the reproduction rate is too low, and the crossover and mutation rates are too high. In Case #3, the graph values are constant because the reproduction rate is too high. Finally, Case #4 seems to have struck an appropriate balance as we observe the values descend gradually and finally converge. Detailed experimental results of Case #4 are described in Table 4.

Although we found the appropriate GA conditions for the problem domain using the preliminary tests, we attempted to optimize the solution further by selecting the best generation from all the generations in Case #4. The experiment using soft margin SVM learning showed that the final generation was well-optimized. Generations 91–100 showed the best detection rate, with relatively fast processing time. Moreover, a comparison of generations 16–30 with generation 46–60 shows that fewer fields does not always guarantee faster processing, because the processing time is also dependant on the data types of those fields.

We constructed a Self-Organized Feature Map to create various packet clusters and implemented this map using the neural network library of MATLAB [43]. The SOFM has a hexa type of topology, with 20 epoch time, and a dimension size of 10x10. Moreover, we prepared 60,000 raw packets, 30,000 from MIT's Lincoln Lab, and 30,000 from our institute (Korea University). The data used from MIT's Lincoln Lab consisted of only attack-free data, or *normal* packets. The data collected by our institute included various kinds of packets

Table 3
Preliminary test parameters of GA

	No. of population	Reproduction rate (Pr)	Crossover rate (Pc)	Mutation rate (Pm)	Final fitness value
Case #1	100	0.100	0.600	0.001	11.72
Case #2	100	0.900	0.900	0.300	14.76
Case #3	100	0.900	0.600	0.100	25.98
Case #4	100	0.600	0.500	0.001	19.12
					Ave: 17.89 SD: 6.187

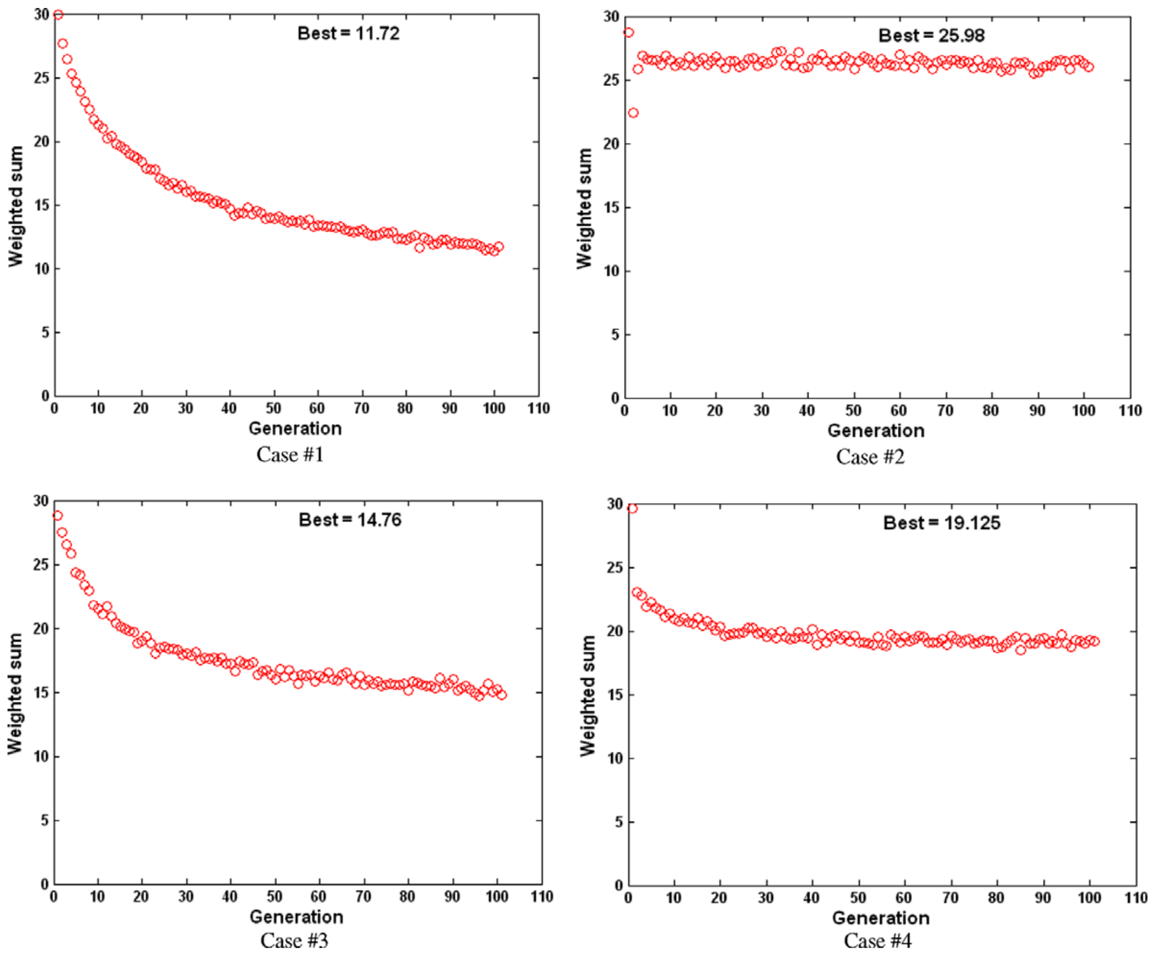

Fig. 5. Evolutionary Process according to preliminary test parameters.

collected from the Internet. The features that SOFM used to cluster the input data are the same as the selected fields of the GA-process. We applied two kinds of clustering schemes for making a more reasonable profile with normal packets, during this packet normalization using SOFM. One was to make clusters according to pre-captured guide-lines such as those set forth by the MIT's Lincoln Lab data. We made clusters using MIT's Lincoln Lab data and used this information to create a normal profile of our institute's data. This meant that the varied data collected by our institute was grouped according to MIT clusters of attack-free packets. The other method used was to generate a statistical ranking of clusters themselves.

In general, most Internet packets are comprised of normal packets intermingled with a relatively small number of anomaly packets. Therefore, when we rank the clusters, the most highly-ranked clusters will be consisted of normal packets. Tables 5 and 6 show the data normalization results. The tables have four kinds of column titles such as class, count, percent, and accumulated. First, the total class number means the dimension size 100 of the SOFM cluster. Some of the total packets will be included in one of the classes between 1 and 100. In other words, if a cluster has more packets than other clusters, then it indicates that the cluster can be a representative cluster for reflecting a characteristic of packets in the packet domain. Moreover, the count means total frequency and the percent is a class per total ratio. Finally, the accumulated means the sum of a class per each total ratio. Each cluster in Tables 5 and 6 comprised greater than 1% of the total packets (30,000). MIT's data as shown in Table 5 contains 40 clusters each greater than 1% of the total, accounting for 66.53% of the total volume. Our institute's data, as shown in Table 6, contains 21 clusters and accounts for 81.05% of the total volume. In other words, the clusters shown in Tables 5 and 6 reflect the characteristics

Table 4
GA field selection results of preliminary test #4

Generation units	Number of selected fields	Number of selected fields	DR (%)	FP (%)	FN (%)	PT (ms)
01–15	19	2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 21, 22, 23, 24	96.68	1.79	7.00	2.27
16–30	15	2, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17, 20, 21, 23, 24	95.00	0.17	16.66	1.90
31–45	15	2, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17, 20, 21, 23, 24	95.00	0.17	16.66	1.90
46–60	18	1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 17, 19, 20, 22, 23, 24	95.12	0.00	16.66	1.84
61–75	17	2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21	73.17	0.00	91.60	0.30
76–90	17	2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20,v21	73.17	0.00	91.60	0.30
91–100	15	3, 5, 6, 7, 9, 12, 13, 16, 17, 18, 19, 21, 22, 23, 24	97.56	0.00	8.33	1.74
			Ave: 89.39 SD: 11.12	Ave: 0.30 SD: 0.66	Ave: 35.50 SD: 38.54	Ave: 1.46 SD: 0.81

DR: detection rate, FP: false positive, FN: false negative, PT: processing time (SVM learning time with 2000 packets).

of normal behaviors. From these results, we made three kinds of training sets to use in the real world experiments, namely the top 40 clusters of MIT's data, the top 21 clusters of our institute's data, and our institute's data set as filtered by MIT's clustering results. For all non-real world experiments, the MIT attack-free dataset was used because the objective of these experiments was not to measure learning efficiency but to compare performance among the machine learning algorithms.

6.2. Comparisons among the Three SVM's and Cross Validation Tests

In this analysis, the three SVM's were tested as follows: soft margin SVM as a supervised method, one-class SVM as an unsupervised method, and the proposed Enhanced SVM. The results are summarized in Table 7. Each SVM approach was tested using four different kinds of SVM kernel functions. The high performance of soft-margin SVM is not a surprising result, since the soft-margin SVM is based on a well-developed supervised learning method. However, we reemphasize that due to its supervised nature it is not suitable for detecting novel attacks. In the case of the one-class SVM, the RBF kernel provided the best performance (94.65%); however, the false positive rate was as high as in the previous discussion. Moreover, we could not see the results of the sigmoid kernel experiment of the one-class SVM, because the data was overfitted. The overfitting of kernel functions also appeared in three of the four Enhanced SVM experiments. It appears that Enhanced SVM is very sensitive to feature mapping using a kernel function, because it employs two kinds of machine learning features. However, in the end, the Enhanced SVM experiment with a sigmoid kernel showed not only similar detection performance as the soft margin SVM, but also showed consistently lower false positive and negative rates than that of the one-class SVM.

We evaluated the proposed approaches using an *m*-fold cross validation test, which is the standard technique used to obtain an estimation of a method's performance over unseen data. Cross validation tests were performed using a 3-fold cross validation method on 3000 normal packets divided into three subsets, with the holdout method [4] repeated three times. We used the Enhanced SVM because this scheme showed the most reasonable performance from overall experiment results of Table 7. Each time we ran a test, one of the three subsets was used as the training set and all subsets were grouped together to form a test set. The results are illustrated in Table 8, and shows that the performance of each method depended on the training set used.

Although all validation sets were attack-free datasets from the MIT Lincoln Lab dataset, there were many differences between Validations sets #1, #2, and #3. The results show that tests were sensitive to how well the collected learning sets incorporated a wide variety of normal and abnormal features. Therefore, from the 3-

Table 5
MIT cluster results

Index	Class	Count	Percent	Accumulated
1	40	1062	3.54	3.54
2	5	841	2.80	6.34
3	21	776	2.59	8.93
4	100	725	2.42	11.35
5	39	694	2.31	13.66
6	31	661	2.20	15.86
7	25	659	2.20	18.06
8	7	657	2.19	20.25
9	3	638	2.13	22.38
10	16	605	2.02	24.39
11	32	597	1.99	26.38
12	10	585	1.95	28.33
13	28	576	1.92	30.25
14	18	569	1.90	32.15
15	8	539	1.80	33.95
16	1	526	1.75	35.7
17	2	516	1.72	37.42
18	38	500	1.67	39.09
19	11	488	1.63	40.71
20	19	472	1.57	42.29
21	24	452	1.51	43.79
22	60	427	1.42	45.22
23	29	397	1.32	46.54
24	27	394	1.31	47.85
25	37	389	1.30	49.15
26	14	382	1.27	50.42
27	23	381	1.27	51.69
28	98	380	1.27	52.96
29	13	379	1.26	54.22
30	4	372	1.24	55.46
31	63	358	1.19	56.66
32	95	357	1.19	57.85
33	69	350	1.17	59.01
34	75	332	1.11	60.12
35	90	331	1.10	61.22
36	67	330	1.10	62.32
37	33	325	1.08	63.41
38	17	321	1.07	64.48
39	20	310	1.03	65.51
40	77	306	1.02	66.53

fold cross validation experiments, the training with Validation Set #2 showed the best overall detection rate and the lowest average false positive rate, suggesting that it contained more well-organized normal features than other validation sets.

6.3. Comparison with Real NIDS

Finally, we compared the proposed framework using the enhanced SVM method with NIDS systems based on the real data. Network traffic was captured from our institute, and treated as a black box in terms of the attack fingerprints. Snort and Bro are some of the most well-known and high performance NIDSs available. Snort is a GPL-licensed network intrusion detection system. It was originally written by Martin Roesch, however, now, it is maintained by Sourcefire. Snort is capable of performing real-time traffic analysis, packet logging on IP networks, protocol analysis, and content searching/matching. In addition, it can be used to detect a variety of attacks and probes, including buffer overflows, stealth port scans, CGI attacks, SMB probes, OS

Table 6
Korea University's cluster results

Index	Class	Count	Percent	Accumulated
1	72	5090	16.97	16.97
2	73	4417	14.72	31.69
3	61	2322	7.74	39.43
4	98	1892	6.31	45.74
5	89	1791	5.97	51.71
6	78	1375	4.58	56.29
7	77	1244	4.15	60.44
8	88	1026	3.42	63.86
9	71	986	3.29	67.14
10	87	978	3.26	70.4
11	51	872	2.91	73.31
12	62	855	2.85	76.16
13	97	651	2.17	78.33
14	94	476	1.59	79.92
15	95	462	1.54	81.46
16	93	418	1.39	82.85
17	64	401	1.34	84.19
18	99	400	1.33	85.52
19	44	362	1.21	86.73
20	96	357	1.19	87.92
21	54	330	1.10	89.02

Table 7
Overall experiment results of SVM's

	Kernels	Detection rate (%)	False positive rate (%)	False negative rate (%)
Soft margin SVM	Inner product	90.13	10.55	4.36
	Polynomial	91.10	5.00	10.45
	RBF	98.65	2.55	11.09
	Sigmoid	95.03	3.90	12.73
	Average	93.73	5.50	9.66
	Standard deviation	3.91	3.51	3.66
One-class SVM	Inner product	53.41	48.00	36.00
	Polynomial	54.06	45.00	46.00
	RBF	94.65	20.45	44.00
	Sigmoid	–	–	–
	Average	67.37	37.82	42.00
	Standard deviation	23.62	15.11	5.29
Enhanced SVM	Sigmoid	87.74	10.20	27.27

fingerprinting attempts. Bro is also an open-source and unix-based NIDS that passively monitors network traffic and looks for suspicious traffic. It detects network intrusions by comparing network traffic against a customizable set of rules describing events that may be malicious. If Bro detects an event of interest, then it can be instructed to either generate a log entry, alert the operator in real-time or initiate the execution of an operating system command [31,34]. In this way, Snort and Bro are respected NIDSs and have capabilities of contents-based signature matching and scan detection. However, Snort has an advantage of analyzing GUIs, configuration GUIs, and a large user community. On the other hand, Bro has flexible site customization, contextual signature, and high-speed network capability.

In the experiment of Real-world Test (referred to as Table 9), Snort and Bro do not require training datasets because they are signature-based NIDSs. However, in the Enhanced SVM, we used three kinds of training

Table 8
3-fold cross validation results

Training	Test	Detection rate (%)	False positive rate (%)
Validation set #1	Validation set #1	79.40	20.60
	Validation set #2	82.50	17.50
	Validation set #3	89.80	10.20
	Average	83.90	16.10
	Standard deviation	5.34	5.34
Validation set #2	Validation set #1	85.40	14.60
	Validation set #2	88.30	12.70
	Validation set #3	92.40	7.60
	Average	88.70	11.63
	Standard deviation	3.52	3.62
Validation set #3	Validation set #1	9.20	90.80
	Validation set #2	17.00	83.00
	Validation set #3	13.70	86.30
	Average	13.30	86.70
	Standard deviation	3.92	3.91

sets as follows: TR1, the high-ranked clusters of MIT; TR2, the data collected by our institute and clustered according to the MIT cluster results; and TR3, the high-ranked packets captured by our institute, independent of the MIT results. In each test experiment, four kinds of test sets were used. Test #1 was an attack-free MIT Lincoln Lab dataset not used during training. Test #2 contained ten kinds of attacks found in the data captured by our institute. Test #3 contained nine kinds of attack data from the DARPA IDS Test. Test #4 comprised of real data collected by our Institute.

From the results shown in Table 9, both signature-based NIDS performed as expected, detecting known attacks in Test #2 and Test #3. Furthermore, although Bro exhibited a lower detection rate, the result does not mean the Bro is in bad shape, because, unlike snort Bro does not have a standard rule configuration file for off-line analysis. This means that Bro users can construct their own Bro rule sets with individual Bro rule files. When we used the rule files suggested by Bro programmer, Vern Paxson [31], it worked as well against specific attacks, similar to the operation of Snort.

In the case of Test #1's attack-free data, Snort and Bro generated many warning messages such as "tiny fragment", "bad UDP checksum," and so on. On the other hand, the framework worked well in TR1, because albeit different data-sets were used for training and testing, the SVM classifier was trained by the attack-free data set of MIT's Lincoln Lab. TR2 and TR3 were generated by the own normal profile and showed better results than TR1. This means that the profile reflects a more robust variety of normal packet characteristics than the normal TR1 profile based on MIT's data.

In the case of Test #2 and #3's known attacks, even though the two NIDS demonstrated better performance than the Enhanced SVM, the result cannot be directly compared because they are the best-known signature-based systems and their detection rates are proportional to the number of rules in their rule-set databases.

In response to Test #4's unknown, real data, even though packet analysis was not done to ensure correct results, Snort and Bro performed much as they did in Test #1. When we compare their results with Test #4 with Test #1, we observe the following detection rates: Test #1 and #4 of Snort were 94.77% and 93.62%, respectively, and Test #1 and #4 in Bro were 96.56% and 97.28%, respectively. This most likely means that the real data of Test #4 was highly similar to the attack free data of Test #1 and seems not to have contained any serious or novel attacks. In the case of the TR results, TR2 and TR3 showed higher detection rate than TR1. We analyzed it because TR2 and TR3 originated from our Institute's raw packets. In other words, TR2

Table 9
Real-world test results

		Test sets	Detection rate (%)	False positive rate (%)	False negative rate (%)
Enhanced SVM	TR1	Test#1 – Normal	92.40	7.60	–
		Test#2 – Attacks	68.70	–	31.30
		Test#3 – Attacks	74.47	–	25.53
		Test#4 – Real	99.80	0.20	–
	TR2	Test#1 – Normal	94.53	5.46	
		Test#2 – Attacks	66.90		33.10
		Test#3 – Attacks	65.60		34.40
		Test#4 – Real	99.93	0.07	
	TR3	Test#1 – Normal	95.63	4.37	
		Test#2 – Attacks	64.20		35.8
		Test#3 – Attacks	57.20		42.80
		Test#4 – Real	99.99	0.01	–
	Analysis	Test#1 – Normal	Ave: 94.19, SD: 1.64	Ave: 5.81, SD: 1.64	Ave: 0.00, SD: 0.00
		Test#2 – Attacks	Ave: 66.60, SD: 1.26	Ave: 0.00, SD: 0.00	Ave: 33.40, SD: 2.26
		Test#3 – Attacks	Ave: 65.76, SD: 8.64	Ave: 0.00, SD: 0.00	Ave: 34.24, SD: 8.63
		Test#4 – Real	Ave: 99.90, SD: 0.10	Ave: 0.09, SD: 0.09	Ave: 0.00, SD: 0.00
Snort		Test#1 – Normal	94.77	5.23	–
		Test#2 – Attacks	80.00	–	20.00
		Test#3 – Attacks	88.88	–	11.12
		Test#4 – Real	93.62	6.38	–
Bro		Test#1 – Normal	96.56	3.44	–
		Test#2 – Attacks	70.00	–	30.00
		Test#3 – Attacks	77.78		22.22
		Test#4 – Real	97.29	2.71	–

and TR3 can reflect more realistic distribution of packets than TR1. Therefore, we can see the test (Test #4) results are similar to Test #1.

7. Conclusion

The overall goal of the *Enhanced SVM* approach was to propose a general framework for detection and classification of novel attacks in network traffic. The overall framework consisted of an Enhanced SVM-based anomaly detection engine and its supplement components, such as packet profiling using SOFM, packet filtering using PTF, field selection using GA, and packet flow-based data preprocessing. SOFM clustering was used for normal profiling. To describe in detail, we made three kinds of training sets using SOFM clustering. The PTF could diminish abnormal packets about 6% of total network traffic based on TCP/IP standards of well-known platforms. The optimized field selection with GA has relatively fast processing time and better detection rate through four preliminary tests. Specifically, we designed an anomaly detector using the Enhanced SVM with the high performance of a supervised scheme and the unlabeled capability of an unsupervised scheme. Through the packet flow-based data preprocessing, inputs of SVM learning could consider time variation of inputs. Finally, the *m*-fold validation method verified the experimental datasets, and the real world tests demonstrated that the approach can be compared with real NIDSs.

The Enhanced SVM approach with combined features from two machine learning techniques demonstrated an aptitude for detecting novel attacks while Snort and Bro, signature based systems in practice, have well-developed detection performance. The SVM approach exhibited a low false positive rate similar to that of real NIDSs. In addition, it does not require labeling using pre-existing knowledge. Given the results of the system's performance, from both experimental and real world data, and the approaches benefits as a hybrid machine learning algorithm, we assert that the Enhanced SVM framework rivals the most current and state-of-the art NIDSs and justifies deployment in real network environments.

Future work will address more advanced profiling of normal packets using methods such as data mining and data clustering. We expect that the approach will be better in classifying novel attacks as we become able to construct more realistic profiles of normal packets. In addition to profiling normal packets, we also need to apply this framework to real world TCP/IP traffic and consider more realistic packet associations based on statistical traffic distribution.

Acknowledgements

This research was partly supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement).

References

[1] K. Ahsan, D. Kundur, Practical Data Hiding in TCP/IP, in: Proc. Workshop on Multimedia Security at ACM Multimedia, French Riviera, 2002, pp. 7–19.
[2] D. Anderson, TF. Lunt, H. Javits, A. Tamaru, A. Valdes, Detecting Unusual Program Behavior Using the Statistical Component of the Next-Generation Intrusion Detection, SRI-CSL-95-06, Computer Science Laboratory, SRI International, Menlo Park, CA, 1995.
[3] O. Arkin., xprobe, 2004, <http://www.sys-security.com/html/projects/X.html>.
[4] C.G. Atkeson, A.W. Moore, S. Schal, Locally weighted learning for control, AI Review 11 (1996) 11–73.
[5] Amit Konar, Uday K. Chakraborty, Paul P. Wang, Supervised learning on a fuzzy Petri net, Information Sciences 172 (2005) 397–416.
[6] H. Byun, S.W. Lee, A survey on pattern recognition applications of support vector machines, International Journal of Pattern Recognition and Artificial Intelligence 17 (2003) 459–486.
[7] B.D.J. Cabrera, B. Ravichandran, Raman K. Mehra, Statistical traffic modeling for network intrusion detection, in: Proc. the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, CA, 2000, pp. 466–473.
[8] S.B. Cho, Ensemble of structure-adaptive self-organizing maps for high performance classification, Information Sciences 123 (2000) 103–114.
[9] CERT Coordination Center, Denial of Service Attacks, Carnegie Mellon University, 2001, Available from: <http://www.cert.org/tech_tips/denial_of_service.html>.
[10] C.C. Chung, LIBSVM: a library for support vector machines, 2004.
[11] K.C. Claffy, Internet Traffic Characterization, University of California at San Diego, La Jolla, CA, 1994.
[12] C. Corinna, V. Vapnik, Support-vector network, Machine Learning 20 (1995) 273–297.
[13] S. Dumais, H. Chen, Hierarchical classification of Web content, in: Proc. The 23rd annual international ACM SIGIR conference on Research and development in information retrieval, Athens, Greece, 2000, pp. 256–263.
[14] L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, P. Dokas, The MINDS – Minnesota Intrusion Detection System, Next Generation Data Mining, MIT Press, 2004.
[15] E. Eskin, A. Arnold, M. Prerau, Leonid Portnoy and Salvatore Stolfo, A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data, Data Mining for Security Applications, Kluwer, 2002.
[16] Fyodor, nmap: network mapper, Available from: <http://www.insecure.org/nmap/index.html>.
[17] K.A. Heller, K.M. Svore, A. Keromytis, S.J. Stolfo, One class support vector machines for detecting anomalous windows registry accesses, in: Proc. The workshop on Data Mining for Computer Security, Melbourne, FL, 2003, pp. 281–289.
[18] T. Glaser, TCP/IP Stack Fingerprinting Principles, SANS, 2000.
[19] J. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1995.
[20] W. Hu, Y. Liao, V.R. Vemuri, Robust Support Vector Machines for Anomaly Detection in Computer Security, in: Proc. International Conference on Machine Learning, Los Angeles, CA, 2003, pp.168–174.
[21] V. Jacobson, tcpdump, 1989, Available from: <http://www.tcpdump.org>.
[22] T. Joachims, Making large-Scale SVM Learning Practical, Advances in Kernel Methods – Support Vector Learning, MIT Press, 1998 (Chapter 11).
[23] T. Joachims, Estimating the Generalization Performance of an SVM efficiently, in: Proc. the Seventeenth International Conference on Machine Learning, San Francisco, CA, 2000, pp. 431–438.
[24] T. Joachims, mySVM – a Support Vector Machine, University Dortmund, 2002.
[25] T. Kohonen, Self-Organizing Map, Springer, Berlin, 1995.
[26] W. Lee, D. Xiang, Information-theoretic measures for anomaly detection, in: Proc. IEEE Symposium on Security and Privacy, 2001, pp. 130–143.
[27] Lincoln Laboratory, DARPA intrusion detection evaluation, MIT, Available from: <http://www.ll.mit.edu/IST/ideval/index.html>.
[28] P. Massimiliano, V. Alessandro, Properties of support vector machines, Neural Computation Archive 10 (1998) 955–974.
[29] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, 1997.
[30] B.V. Nguyen, An Application of Support Vector Machines to Anomaly Detection, CS681 (Research in Computer Science – Support Vector Machine) report, 2002.
[31] V. Paxson, Bro 0.9, 2004, Available from: <http://bro-ids.org>.

[32] L. Portnoy, E. Eskin, Salvatore J. Stolfo, Intrusion detection with unlabeled data using clustering, in: Proc. ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001). Philadelphia, PA, 2001, pp. 5–8.
[33] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: Proc. the ACM SIGMOD Conference, Dallas, TX, 2000, pp. 427–438.
[34] M. Roesch, C. Green, Snort 2.2.0, 2004, Available from: <http://www.snort.org>.
[35] C.H. Rowland, Covert channels in the TCP/IP protocol suite, Tech. Rep. 5, First Monday, Peer Reviewed Journal on the Internet, 1997.
[36] J. Ryan, M-J. Lin, R. Miikkulainen, Intrusion detection with neural networks, in: Proc. Workshop on AI Approaches to Fraud Detection and Risk Management, AAAI Press, 1997, pp. 72–77.
[37] B. Schölkopf, Estimating the support of a high-dimensional distribution, Neural Computation 13 (2001) 1443–1471.
[38] C.L. Schuba, I.V. Krsul, M.G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, Diego Zamboni, Analysis of a denial of service attack on TCP, in: Proc. IEEE Security and Privacy Conference, 1997, pp. 208–223.
[39] S. Staniford, J. Hoagland, J. McAlerney, Practical automated detection of stealthy portscans, Journal of Computer Security 10 (2002) 105–136.
[40] A.H. Sung, S. Mukkamala, Identifying important features for intrusion detection using support vector machines and neural networks, in: Proc. SAINT, 2003, pp. 209–217.
[41] K. Thompson, G.J. Miller, R. Wilder, Wide area internet traffic patterns and characteristics (extended version), IEEE Network 11 (1997) 10–23.
[42] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, NewYork, 1995.
[43] J. Vesanto, J. Himberg, E. Alhonieme, J. Parhankangas, SOFM in Matlab: SOM toolbox, 1999.
[44] G. Vigna, W. Robertson, D. Balzarotti, Testing network-based intrusion detection signatures using mutant exploits, in: Proc. ACM CCS 2004, Washington, DC, 2004, pp. 21–30.
[45] M.R. Wick, A.T. Phillips, Comparing the template method and strategy design patterns in a genetic algorithm application, ACM SIGCSE 34 (2002) 76–80.
[46] Dong-Her Shih, Hsiu-Sen Chiang, C. David Yen, Classification methods in the detection of new malicious emails, Information Sciences 172 (2005) 241–261.
[47] M.Zalewski, William Stearns, p0f, Available from: <http://lcamtuf.coredump.cx/p0f>.