Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 32 (2005) 2617–2634
computers & operations research
www.elsevier.com/locate/dsw

Application of SVM and ANN for intrusion detection

Wun-Hwa Chen, Sheng-Hsun Hsu*, Hwang-Pin Shen

9 fl. Graduate Institute of Business Administration, 1st Building of College of Management, National Taiwan University, No. 1, Section 4, Roosevelt Road, Taipei 106, Taiwan

Accepted 29 March 2004

Abstract

The popularization of shared networks and Internet usage demands increases attention on information system security, particularly on intrusion detection. Two data mining methodologies—Artificial Neural Networks (ANNs) and Support Vector Machine (SVM) and two encoding methods—simple frequency-based scheme and $tf \times idf$ scheme are used to detect potential system intrusions in this study. Our results show that SVM with $tf \times idf$ scheme achieved the best performance, while ANN with simple frequency-based scheme achieved the worst. The data used in experiments are BSM audit data from the DARPA 1998 Intrusion Detection Evaluation Program at MIT's Lincoln Labs.

Keywords: Intrusion detection; Artificial neural networks; Support vector machine

1. Introduction

Electric commerce and the recent online consumer boom have forced a change in the basic computer security design for systems on a shared network. Systems are now designed with more flexibility and less barrier security. Furthermore, as computers become more financially available to the masses, they also become increasingly consumer-oriented. The combination of user friendliness and public accessibility, although advantageous for the average person, inevitably renders any exchanged information vulnerable to criminals. Consumer information, employee data or intellectual property stored in internal data warehouses are all at risk, from external attackers and disgruntled employees who might abuse their access privileges for personal gain.

Security policies or firewalls have difficulty in preventing such attacks because of the hidden weaknesses and bugs contained in software applications. Moreover, hackers constantly invent new

* Corresponding author. Tel.: +886-2-23397759; fax.: +886-2-23654914.
E-mail addresses: spolo@handel.mba.ntu.edu.tw, spolo@ms15.hinet.net (S.-H. Hsu).

0305-0548/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2004.03.019

Table 1
Techniques for intrusion detection systems

Research	Input data format	Methods	Levels
[4]	Sequence of system calls	RIPPER Hidden Markov models	Hosts, program
[5,6]	Sequence of system calls (sendmail)	RIPPER	Hosts, program
[7]	Sequence of system calls (sendmail and lpr)		Hosts, program
[8]	Sequence of system calls (sendmail)	RIPPER	Hosts, program
[9]	Sequence of system calls	ANN	Hosts, program
[10]	Sequence of system calls (sendmail and lpr)	ANN	Hosts, program
[11]	Frequency of system calls	k-nearest neighbor	Hosts, program
[12]	TCP/IP data	ANN SVM	Network, program
[13]	User logs	SVM	Hosts, user

attacks and disseminate them over the Internet. Disgruntled employees, bribery and coercion also make networks vulnerable to attacks from the inside [1]. Mere dependence on the stringent rules set by security personnel is not sufficient. Intrusion detection systems, which can detect, identify and respond to unauthorized or abnormal activities, have the potential to mitigate or prevent such attacks. Accordingly, the security breech is an area with increasing concern to the Internet community [2].

There are proposed techniques that could identify attacks (see Table 1); however, the very success of these techniques may also lead to their own downfall. Attackers will study the rationales behind these techniques and subsequently change their behavior to evade these techniques [3]. New ways to identify attacks are now necessary.

Most researchers (see Table 1) used short sequences of systems calls to characterize program behavior. A few used the frequency distribution of the system calls (e.g., [11]). Frequency-based encoding techniques incur less overhead than sequence-based encoding techniques, which require building a profile for each program (e.g., an individual profile must be built for sendmail or lpr[1]) and checking for attacks at every time frame. Frequency-based encoding techniques build a profile only for each process and not for each program (a process might contain several programs) and check for attack instances at the end of the process. This greatly reduces the system overhead.

In this study, we explore the feasibility of applying an Artificial Neural Network (ANN) and Support Vector Machine (SVM) to predict attacks based on frequency-based encoding techniques. The goal of using ANN and SVM for attack detection is to develop a generalization capability from limited training data. In addition to comparing the ANN and SVM performances, we demonstrate other encoding methods in predicting attacks. The test bed used here is 1998 DARPA data from MIT's Lincoln Labs.

[1] Sendmail and lpr are two commonly programs used in Unix system.

This paper is organized as follows: Section 2 presents an overview of the intrusion detection systems. Section 3 introduces the ANN and SVM. Section 4 presents the results followed by conclusions and limitations.

2. Overview of intrusion detection systems

An intrusion is unauthorized access or use of computer system resources. Intrusion detection systems are software that detects, identifies and responds to unauthorized or abnormal activities on a target system. The major functions performed by intrusion detection systems are: (1) monitor and analyze user and system activities, (2) assess the integrity of critical system and data files, (3) recognize activity patterns reflecting known attacks, (4) respond automatically to detected activities, and (5) report the outcome of the detection process [14].

Intrusion detection techniques can be categorized into misuse detection and anomaly detection. Misuse detection uses the patterns of well-known attacks or vulnerable spots in the system to identify intrusions. However, only known attacks that leave characteristic traces can be detected this way. Anomaly detection, on the other hand, attempts to determine whether deviations from the established normal usage patterns can be flagged as intrusions [5]. Although misuse detection can achieve a low false positive rate (the rate of misclassified normal behavior), minor variations of a known attack occasionally cannot be detected [9]. Anomaly detection can detect novel attacks, yet it suffers a higher false positive rate.

An ideal intrusion detection system is one that has a high attack detection rate along with a 0% false positive rate. However, such a low rate of false positives is only achieved at the expense of ignoring minor malicious activity detection. This provides an attacker with a small window of opportunity to perform arbitrary behaviors, giving them insight regarding the type of the intrusion detection system in use [15]. Because of the complementary nature of these two approaches, many systems attempt to combine both techniques. The misuse detection techniques can be implemented as a first line of defence, while the anomaly detection techniques can be used as a second line [14,16].

Intrusion detection systems are categorized according to the kind of audit source location they analyze. Most intrusion detection systems are classified as either a network-based or a host-based approach to recognize and detect attacks. A network-based intrusion detection system performs traffic analysis on a local area network. A host-based intrusion detection system places its reference monitor in the kernel/user layer and watches for anomalies in the system call patterns. The advantages of using network-based intrusion detection systems are no processing impact on the monitored hosts, the ability to observe network-level events, and monitoring an entire segment at once. However, as the complexity and capacity of networks increase, the performance requirements for probes can become prohibitive [14]. Host-based intrusion detection systems can analyze all activities on the host, including its own network activities. Unfortunately, this approach implies a performance impact on every monitored system.

Additionally, there are at least two levels on which an intrusion detection system could monitor activities. One is at the user level and the other is at the program level. Recently, learning program behavior and building program profiles, especially those privileged programs, have become alternative methods used in intrusion detection. In a dynamic environment, it is nearly impossible to create user profiles that determine normal behavior [11]. In these cases, it would be better to utilize an

intrusion detection system that observes the behavior of processes rather than users [17]. Compared to user behavior profiles, program profiles are more stable over time due to limited range of program behavior [11].

2.1. Anomaly modeling techniques

2.1.1. Statistical models

In Denning's ground laying paper on intrusion detection systems [18], she described several statistical characterizations of events and event counters. These, and more refined techniques, have been implemented in anomaly detection systems. These techniques include (1) Threshold measures: a common example is logging and disabling user accounts after a set number of failed login attempts, (2) Mean and standard deviation: by comparing event measures to a profile mean and standard deviation, a confidence interval for abnormality can be established, (3) Multivariate models: calculating the correlation between multiple event measures relative to the profile expectations.

2.1.2. Immune system approach

Application implementations inherently provide a model of normal behavior in the form of application code paths. In the immune system approach, applications are modeled in terms of the system call sequences. One of the first works analyzing system call sequences for intrusion detection is described in [7]. Forrest et al. [7,19] discovered that the short sequences of system calls made by a program during its normal executions are very consistent and deviations from these short sequences of system calls could be used to identify security violations of an executing process. An alarm is fired when the number of anomalies counted exceeds a threshold. This is known as the stide algorithm. The principle behind this scheme is that when an intrusion actually occurs, the majority of the adjacent system call sequences become abnormal.

2.1.3. Markov process model

Markov processes are widely used to model systems in terms of state transitions. Some intrusion detection algorithms exploit the Markov process model. These methods do not use system call sequences, but instead analyze the state transitions for each system call. In state transition analysis, an event is considered anomalous if its probability, given the previous state and associated value in the state transition matrix, is too low [4].

2.1.4. Rule-based algorithm

One of the most used rule-based algorithms in the intrusion detection field is Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which is a rule learning system developed by William Cohen [20]. This algorithm performs classifications by creating a list of rules from a set of labeled training examples.

2.1.5. Data mining techniques

Many recent approaches to intrusion detection systems utilize data mining techniques [21]. These approaches build detection models by applying data mining techniques to large data sets of an audit trail collected by a system [22].

3. Methodology

3.1. Artificial neural network

ANN is a biologically inspired form of distributed computation. It is composed of simple processing units, or nodes, and connections between them. The connection between any two units has some weight, which is used to determine how much one unit will affect the other. A subset of the units acts as input nodes and another subset acts as output nodes, which perform summation and thresholding. An early stopping strategy is usually used to overcome the over-fitting problem. The early stopping method tracks the network performance using a separate validation set. Typically, the errors in the validation set will decrease as the network fits the data, and then increase as the network fits idiosyncrasies in the training data. The ANN has successfully been applied in different settings, including network reliability, sports winning prediction, medical, marketing, retail, banking and finance, insurance, telecommunications, operations management and other industries [23–26]. In this study, we will employ a classic feed-forward neural network trained with the back-propagation algorithm to predict intrusions.

A feed-forward neural network has two stages: a forward pass and a backward pass. The forward pass involves presenting a sample input to the network and letting activations flow until they reach the output layer. The linear sum, sigmoid function and Gaussian function are three often used activation functions. During the backward pass, the network's actual output (from the forward pass) is compared with the target output and error estimates are computed for the output units. The weights connected to the output units can be adjusted to reduce those errors. The detailed algorithm is presented in Fig. 1.

3.2. Support vector machine

SVM has recently been introduced as a new technique for solving a variety of learning, classification and prediction problems. SVM originated as an implementation of Vapnik's [27] structural risk minimization (SRM) principle, which minimizes the generalization error, i.e., true error on unseen examples. The basic SVM deals with two-class problems—in which the data are separated by a hyperplane defined by a number of *support vectors*. Support vectors are a subset of training data used to define the boundary between the two classes. In situations where SVM cannot separate two classes, it solves this problem by mapping input data into high-dimensional feature spaces using a kernel function. In high-dimensional space it is possible to create a hyperplane that allows linear separation (which corresponds to a curved surface in the lower-dimensional input space). Accordingly, the kernel function plays an important role in SVM. In practice, various kernel functions can be used, such as linear, polynomial or Gaussian.

In the two-dimensional case, the SVM action can be illustrated using Fig. 2. In Fig. 2, a series of points for two different classes of data are shown, circles (class A) and squares (class B). The SVM attempts to place a linear boundary (solid line) between the two different classes and orients this line in such a way that the margin (space between dotted lines) is maximized. The nearest data points used to define the margin are known as support vectors (gray circles and square). Support vectors, not the number of input features, contain all of the information needed to define the classifier. One remarkable property of SVM is its ability to learn can be independent of the feature space

Gradient-Decent
Each training example is a pair of the form $<\vec{x},t>$, *where* $\vec{x}$ *is the vector of input values, and t is the target output value,* η *is the learning rate*
Begin
Initialize each w_{ij} to some small random value
While
For each $<\vec{x},t>$ in training data set
Input the instance to the unit $\vec{x}$ and compute the output *o*
For each weight w_{ij}, computes and updates
$\Delta w_{ij}(t+1) = \Delta w_{ij}(t) + \eta \delta_j' x_i'$
if node j is an output node, then
$\delta_j = o_j(1-o_j)(t_j - o_j)$
if node j is an internal hidden node, then
$\delta_j = x_j'(1-x_j')\sum_k \delta_k w_{jk}$
Until convergence
End

Fig. 1. ANNs algorithm.

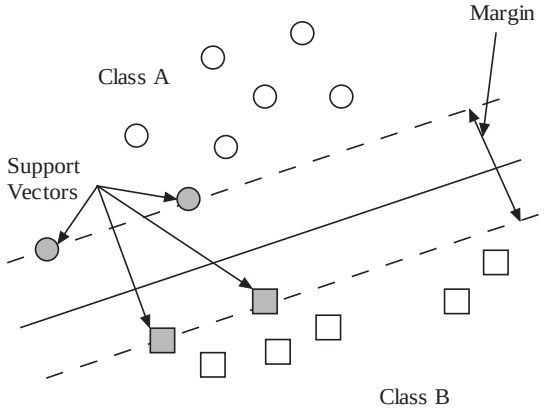

Fig. 2. Separation of two classes by SVM.

dimensionality. This means that SVM can generalize well in the presence of many features. Fig. 2 presents the simplest model of SVM called the maximal margin classifier. It works only for data that are linearly separable in the feature space. Though it is the easiest algorithm and not very useful in real-world situations, it forms the building block for understanding the complex SVM models. Space here prevents a detailed explanation of the SVM principles; however, a good introduction to the SVM theory can be found in Refs. [27–29].

Mathematically, the linear boundary can be expressed in terms of

$$w^{\mathrm{T}}x + b = 0. \tag{1}$$

In a classification problem, we try to estimate a function $f : \Re^n \mapsto \{\pm 1\}$ using training data. Let us denote the class A with $x \in A$, $y = 1$ and class B with $x \in B$, $y = -1$; $(x_i, y_i) \in \Re^n \times \{\pm 1\}$. If the

Support vector machine

Problem def:

For a classification problem, we try to estimate a function $f:\Re^n \mapsto \{\pm 1\}$ using training data. So, let us denote two classes – A and B. The class A with $x \in A$, y = 1 and the class B with $x \in B$, y = -1 $(x_i, y_i) \in \Re^n \times \{\pm 1\}$. If the training data are linearly separable then there exists a pair $(w,b) \in \Re^n \times R$ such that $y(w^T x + b) \geq 1$, for all $x \in A \cup B$.

1. SVM belongs to the type of maximal margin classifier, in which the classification problem can be represented as an optimization problem

$$\underset{w,b}{\text{Minimize}}\ \Phi(w)=\frac{1}{2}\|w\|^2 \qquad s.t.\ y(w^T x + b) \geq 1$$

2. The dual of the optimization problem: find multipliers λ_i which maxmize

$$W(\Lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2}\|w\|^2 = \sum_{i=1}^{l} \lambda_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \lambda_i \lambda_j y_i y_j x_i^T x_j,$$

and by the Karush-Kuhn-Tuker (KKT) complementary conditions

$$\lambda_i [y_i (w^T x_i + b) - 1] = 0, \qquad i = 1, \ldots, l,$$

Fig. 3. SVM algorithm.

training data are linearly separable then there exists a pair $(w, b) \in \Re^n \times R$ such that

$$w^{\mathrm{T}} x + b \geqslant +1 \quad \text{for all } x \in A, \tag{2}$$

$$w^{\mathrm{T}} x + b \leqslant -1 \quad \text{for all } x \in B \tag{3}$$

with the decision function given by

$$f_{w,b}(x) = \operatorname{sign}(w^{\mathrm{T}} x + b), \tag{4}$$

w is termed the weight vector and b the bias. The inequality constraints (2) and (3) can be combined to give

$$y(w^{\mathrm{T}} x + b) \geqslant 1 \quad \text{for all } x \in A \cup B. \tag{5}$$

The maximal margin classifier optimizes this by separating the data with the maximal margin hyperplane. The learning problem is hence formulated as: minimize $1/2^*\|w\|^2$ subject to the constraints of linear separability (5). The optimization is a quadratic programming (QP) problem:

$$\underset{w,b}{\text{Minimize}}\ \Phi(w) = \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y(w^{\mathrm{T}} x + b) \geqslant 1.$$

In applications, two parameters must be decided beforehand—the kernel function and the margin that separates the data. Fig. 3 presents the main concepts of the maximal margin classifier.

Compared with the ANN, the SVM has two advantages. Firstly, the global optimum can be derived. Secondly, the over-fitting problem can be easily controlled by the choice of a suitable margin (i.e., support vectors) that separates the data. Empirical testing has shown that the SVN performance is better than that for the ANN in classification (e.g., [30,31]) and regression problems (e.g., [32]).

4. Results and discussion

4.1. Data set

The data used here originated from MIT's Lincoln Labs. It was developed for KDD competition by DARPA and is considered a standard benchmark for intrusion detection evaluations. We used BSM audit data from the DARPA 1998 intrusion detection evaluation program.[2] The data produced by BSM are compact binary representation of system calls made by a program to the Solaris kernel. A tool called praudit is distributed with Solaris to convert the BSM data into human readable ASCII form. Individual sessions, processes and system calls are extracted from the converted data. Each session consists of at least one or more processes, and each process is composed of a set of system calls.

Instead of looking at the local sequence of the system calls, we followed the techniques employed by Liao and Vemuri [11]. They used a frequency-based encoding method by aggregating system call information over the entire execution of a process to characterize program behavior. Using the text processing metaphor, each system call is treated as a "word" in a document and the set of system calls generated by a process is treated as the "document". This analogy makes it possible to bring the full spectrum of well-developed text processing methods to apply to the intrusion detection problem. Analogous to text categorization, each process is first represented as a vector, where each entry represents the occurrence of a specific system call during the process execution.

4.2. Procedure of our test

Our test procedure is outlined in Fig. 4. First, the BSM audit data are converted and represented by the frequency distribution of system calls, which is similar to vector space representation developed by Slaton [33]. The training data set is then separated into attack data sets and normal data sets, which are then subsequently fed into the ANN and SVM algorithms. Through the training process, both ANN and SVM predictive models can be built. We then feed the test data set into the ANN and SVM predictive models. By varying the detection threshold, the Receiver Operating Characteristic (ROC) curve for both models is produced.

4.3. Data processing

From the 7-week (35 days) DARPA training data, 21 days had attacks. We selected 10 days on which most attacks appeared for inclusion in our data. The data were then arbitrarily divided into two parts: one 5-day part was treated as the training dataset and the other 5-day part was treated as the test dataset. As described in Section 4.1, the BSM data include sessions, processes and system calls. Each session consists of at least one or more processes, and each process is composed of a set of system calls. Some sessions are attack sessions and some are normal sessions. A sample of a session and a sample of a process with its system call list are shown in Figs. 5 and 6, respectively. A session contains a sequential index, start time, duration, service name and IP addresses and ports of the source and the destination. A process is composed of system calls.

[2] http://www.ll.mit.edu/IST/ideval/index.html

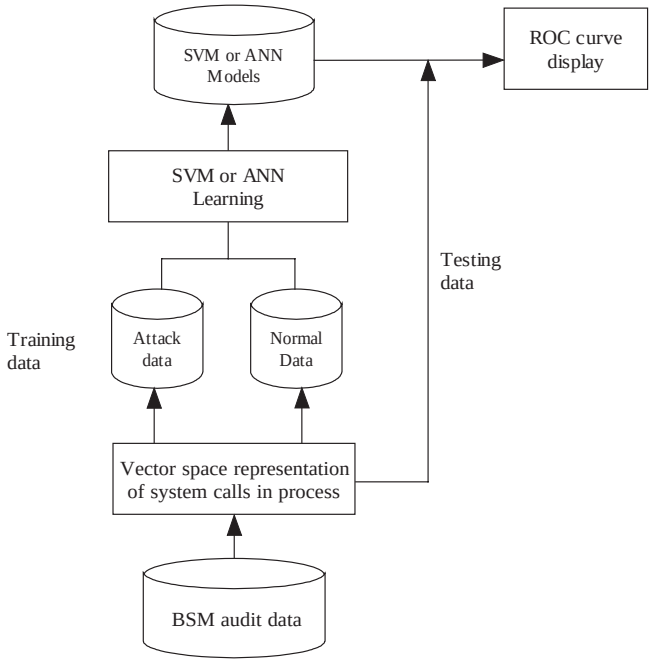

Fig. 4. Procedure of our test.

873 06/11/1998 08:36:57 00:00:01 smtp 3631 25 192.168.001.010 172.016.112.050

The log means:

Session index: 873

Start time: 1998/06/11 08:36:57 Duration: 00:00:01

Service name: smtp

Source port: 3631 IP: 192.168.001.010

Destination port: 25 IP: 172.016.112.050

Fig. 5. A sample of some session.

By converting BSM data, we identified 50 commonly used system calls (see Fig. 7). For each process ID, we collected system calls belonging to the process and put them into a system call list on the basis of their appearance order. We then counted the numbers of each system call in the system call list, which constitutes the frequency distribution of system calls for the process. A sample of the frequency distribution for system calls belonging to a process ($ID = 6400$) is shown in Fig. 8. This vector represents that the frequencies for "access," "audit," and "auditon" are zero, and the frequency for "close" is 19, and so on.

The aforementioned method is the simplest frequency-based encoding method. There are other methods for representing the vector space of system calls within a process. One of these is the

Process ID: 6400,

Process name: date,

Time interval: 1998/06/12 02:00:26 – 1998/06/12 02:00:26

System calls:

close, close, open, close, close, close, close, close,
close, execve, open, mmap, open, mmap, mmap, munmap,
mmap, close, open, mmap, mmap, munmap, mmap, mmap,
close, open, mmap, mmap, munmap, mmap, close, open,
mmap, close, open, mmap, mmap, munmap, mmap, close,
close, munmap, open, close, ioctl, ioctl, close, close,
close, close, exit,

Fig. 6. A sample of some process.

access, audit, auditon, chdir, chmod, chown, close, creat,
execve, exit, fchdir, fchown, fcntl, fork, fork1, getaudit,
getmsg, ioctl, kill, link, login, logout, lstat, memcntl,
mkdir, mmap, munmap, nice, open, pathdonf, pipe, putmsg,
readlink, rename, rmdir, setaudit, setegid, seteuid, setgid, setgroups,
setpgrp, setrlimit, setuid, stat, statvfs, su, sysinfo, unlink,
utime, vfork.

Fig. 7. A list of 50 common system calls.

(0, 0, 0, 0, 0, 0, 19, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
15, 5, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Fig. 8. A sample of frequencies system calls of the process above.

$tf \times idf$ (term frequency $\times$ inverse document frequency) method, which is a representation scheme commonly used in text mining approaches. The basic assumption is that term frequency (tf) in the given document shows how important the term is in this document and the document frequency of the term (df, the percentage of documents that contain this term) shows how important the term is to the document. A low df value indicates that the term does not appear in many documents, which shows the uniqueness of the term in the corpse of documents. Therefore, instead of adopting df, we chose idf (inverse df) as our weighting scheme. A high weight in a $tf \times idf$ scheme is therefore reached by a high-term frequency in the given document and a low document frequency of the term in the whole database.

(0, 0, 0, 0, 0, 0, -1.99281634142461, 0, 0.00269061492298033,
0.00112889962801342, 0, 0, 0, 0, 0, 0, 0, -0.197807703315683, 0, 0, 0, 0, 0, 0,
0, -0.703017747607287, -0.152961536926822, 0, -0.790558274822214, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Fig. 9. A sample of $tf \times idf$ vectors of the process above.

We utilized a 50-dimension vector to represent a process, with each dimension corresponding to a system call. The components of the vector were calculated using the following formula:

$$c_i = \log\left(\frac{N}{d_i}\right) \times \frac{f_i}{\sqrt{\sum_i f_i^2}}.$$

The meanings of the symbols are:

c_i: component of the process vector corresponding to system call i
d_i: number of times system call i appear in the whole data set
f_i: number of times system call i appear in the process
N: number of processes in the whole data set
A sample of $tf \times idf$ vectors belonging to the process is shown in Fig. 9.

To train the ANN and SVM to recognize attack behaviors, we must have training data containing information about "normal" and "abnormal" processes. Because this information is only contained at the session level, we must develop a relationship between sessions and processes to derive this information. The detailed procedure is as follows: first, we collected system calls belonging to each process in the training data set. To distinguish normal and abnormal processes, we connected session and process information using their timing relations. For example, if some process is executed during an attack session, we considered it an abnormal process; otherwise, we considered it a normal process. We then prepared a frequency distribution of system calls for normal and abnormal processes. To make sure that there was no identical frequency distribution in both normal and abnormal processes, we deleted some processes in normal processes that have the same frequency distribution as those in abnormal processes.

The DARPA data contain hundreds of megabytes of process information. To convert these data into usable information, 1 h 35 min was required (with hardware: Pentium III 1.06 GHz CPU, 256 MB SDRAM). Most of the time was spent on building relations between sessions and processes (83.1% of time in building relations between sessions and processes, 8.2% of time in building frequency-based vectors and 8.7% of time in distinguishing normal and abnormal processes).

4.4. Parameter of ANN and SVM

4.4.1. ANN

The ANN used in this experiment was implemented using Matlab 6.1 Neural Network Toolbox software. There are various learning techniques, such as the quasi-Newton method, back-propagation, Levenberg–Marquardt algorithm, conjugate gradients methods from which to choose. For the sake of efficiency, the Levenberg–Marquardt algorithm was used.

A standard three-layer network is used as a benchmark. In the input layer, there were 50 nodes, equal to the number of system calls. The number of output nodes was equal to 1. The number of hidden nodes was determined using the formula, $W \leqslant M/3$, where W is the number of interconnection weights that satisfies the following equality:

$$W = (I + O)*H, \tag{6}$$

where M is the number of training examples, I the number of input nodes, O the number of output nodes, and H the number of hidden nodes. The network size was controlled by ensuring the ratio of the number of training samples to the number of weights was equal to or larger than 3. Based on Eq. (6), the maximal number of hidden nodes was 10. By varying the number of hidden nodes from 4 to 10, we determined that ANN with 8 (for frequency-based encoding method) and 6 (for $tf \times idf$-based encoding method) hidden nodes achieved the best performance. The hidden nodes used the sigmoid transfer function and the output node used the linear transfer function.

The computational complexity for training the ANN is V^2N, where V is the number of input features (or the number of input nodes) and N is the number of classifiers [34]. This suggests that the computational complexity of ANN depends on the number of input features.

4.4.2. SVM

The typical kernel functions are the polynomial kernel $k(x, y) = (x \times y + 1)^d$ and the Gaussian kernel $k(x, y) = \exp(-(x - y)^2/\delta^2)$, where d is the degree of the polynomial kernel and δ^2 is the bandwidth of the Gaussian kernel. In our experiment, we chose the Gaussian kernel as our kernel function because it tends to achieve better performance.

The parameters that must be determined are the kernel bandwidth δ^2 and the margin C. To determine these two parameters, 10-fold cross validation was used in the training data set to choose parameters that yield the best result. Subsequently, this set of parameters was applied to the test data set. The parameters tried in the 10-fold cross-validation process were $\delta^2 \in \{2, 1, 0.5, 0.1, 0.01, 0.001, 0.0001\}$ and $C \in \{1000, 750, 500, 100, 50, 2\}$. The two sets of chosen parameters were $\delta^2 = 2$ and $C = 1000$, and $\delta^2 = 1$ and $C = 2$ for frequency-based and $tf \times idf$-based encoding methods. A SVM implementation called LIBSVM [3] was used in this work.

The computational complexity for training the SVM was Nm^2, where N is the number of classifiers and m the number of training examples [34]. Compared with the ANN, the SVM algorithm is less sensitive to the size of the input features.

4.5. Anomaly detection

In the intrusion detection, the Receiver Operating Characteristic (ROC) curve is typically used to measure intrusion detection performance. The ROC curve indicates how the detection rate changes as the internal thresholds are varied to generate more or fewer false alarms. The ROC curve is a plot of intrusion detection accuracy against the false-positive probability that tradeoffs detection accuracy against the analyst workload. Here, we define the attack detection rate and false-positive rate as

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Table 2
The results of ANN model with the frequency method (threshold: 0.16–0.40)

Threshold	Attacks		Normal sessions		Attack detection rate (%)	False-positive rate (%)
	Discovered	Real	Misclassified	Real		
0.160	250	250	16,872	41,426	100.00	40.73
0.170	248	250	14,698	41,426	99.20	35.48
0.180	248	250	3178	41,426	99.20	7.67
0.200	248	250	2784	41,426	99.20	6.72
0.230	248	250	2046	41,426	99.20	4.94
0.240	111	250	1934	41,426	44.40	4.67
0.300	108	250	757	41,426	43.20	1.83
0.400	98	250	227	41,426	39.20	0.55

Table 3
The results of SVM model with the frequency method (threshold: 0.50–1.00)

Threshold	Attacks		Normal sessions		Attack detection rate (%)	False-positive rate (%)
	Discovered	Real	Misclassified	Real		
0.500	250	250	4288	41,426	100.00	10.35
0.600	249	250	1729	41,426	99.60	4.17
0.650	248	250	1388	41,426	99.20	3.35
0.675	248	250	1284	41,426	99.20	3.10
0.680	113	250	1247	41,426	45.20	3.01
0.700	113	250	938	41,426	45.20	2.26
0.800	111	250	877	41,426	44.40	2.12
1.000	108	250	739	41,426	43.20	1.78

Liao and Vemuri [11]:

$$\text{Attack detection rate} = \frac{\textit{Number of detected attacks}}{\textit{Number of attacks}} * 100\%,$$

$$\text{False-positive rate} = \frac{\textit{Number of misclassified processes}}{\textit{Number of normal processes}} * 100\%.$$

In the chosen DARPA data set, 250 attacks and 41,426 normal sessions occurred over the 5 days selected.

4.5.1. Simple frequency-based encoding method results

The ROC curves for the ANN and SVM models with simple frequency-based encoding method are shown in Tables 2, 3 and Fig. 10. The SVM model outperformed the ANN model because the ROC curve for the SVM model was located higher than that for the ANN model. Tables 2 and 3 show that in the SVM result, the attack detection rate reached 100.00% (250 of 250) with a false-positive rate of about 10.00% (4288 of 41,426). The ANN attack detection rate reached 100.00 percent with a false-positive rate of about 40.72% (16,872 of 41,426).

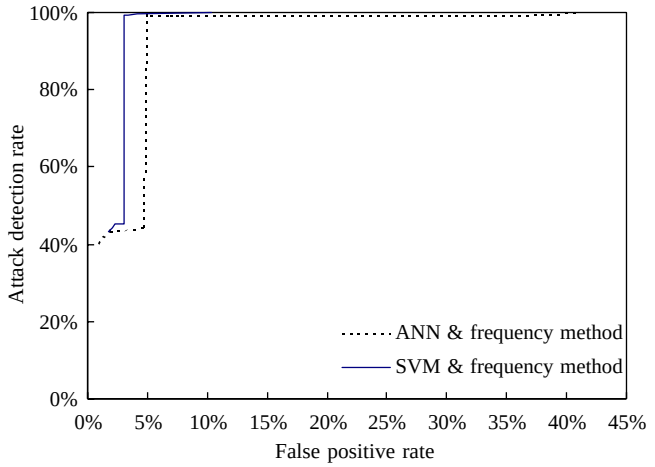

Fig. 10. ROC curves of SVM and ANN models with frequency method.

Table 4
The results of ANN model with the *tf*×*idf* method (threshold: 0.135–0.22)

Threshold	Attacks		Normal sessions		Attack detection rate (%)	False-positive rate (%)
	Discovered	Real	Misclassified	Real		
0.135	250	250	16,260	41,426	100.00	39.25
0.140	249	250	6351	41,426	99.60	15.33
0.150	249	250	2543	41,426	99.60	6.14
0.160	247	250	1828	41,426	98.80	4.41
0.170	247	250	1637	41,426	98.80	3.95
0.180	113	250	1423	41,426	45.20	3.44
0.210	113	250	513	41,426	45.20	1.24
0.220	109	250	168	41,426	43.60	0.41

4.5.2. *tf*×*idf* encoding method results

The ROC curves for the ANN and SVM models with the *tf*×*idf* encoding method are shown in Tables 4, 5 and Fig. 11. Similar to the results of simple frequency-based encoding method, the SVM model outperformed the ANN model, as shown in Fig. 11. Tables 4 and 5 show that in the SVM result, the attack detection rate could reach 100.00% (250 of 250) with a false-positive rate of about 8.53% (3533 of 41,426). The ANN attack detection result reached 100.00% with a false-positive rate of about 39.25%.

4.5.3. Comparison and discussion

The ROC curves for the SVM and ANN with different encoding schemes are shown together in Fig. 12. In summary, the SVM with *tf*×*idf* encoding method performed better than SVM with the simple frequency-based method, which in turn, outperformed the ANN with *tf*×*idf* encoding method and the ANN with simple frequency-based method. A series of paired *t*-tests were conducted to

Table 5
The results of SVM model with the $tf \times idf$ method (threshold: 0.35–0.54)

Threshold	Attacks		Normal sessions		Attack detection rate (%)	False-positive rate (%)
	Discovered	Real	Misclassified	Real		
0.350	250	250	3533	41,426	100.00	8.53
0.360	249	250	1474	41,426	99.60	3.56
0.370	249	250	1188	41,426	99.60	2.87
0.385	247	250	1097	41,426	98.80	2.65
0.400	247	250	828	41,426	98.80	2.00
0.410	113	250	695	41,426	45.20	1.68
0.430	113	250	476	41,426	45.20	1.15
0.450	109	250	136	41,426	43.60	0.33
0.540	109	250	111	41,426	43.60	0.27

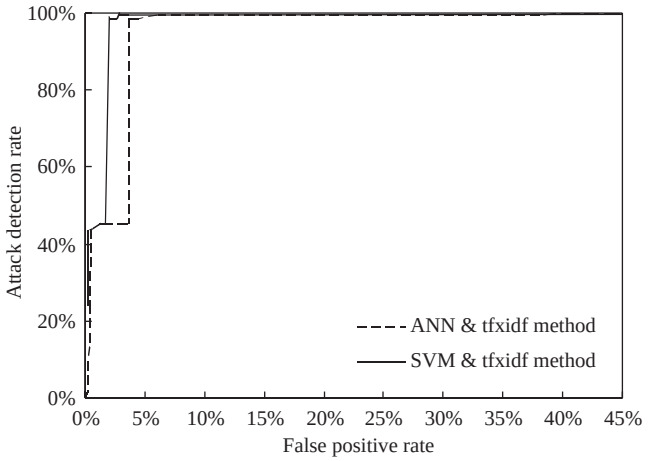

Fig. 11. ROC curves of SVM and ANN models with $tf \times idf$ method.

test the statistical difference between these different algorithms and encoding schemes. The results shown in Table 6 reach the same conclusion that SVM with $tf \times idf$ encoding method performed the best, while ANN with simple frequency-based encoding method performed the worst. This further supports that SVM is better than ANN for intrusion detections and $tf \times idf$ encoding method is better than simple frequency-based encoding method.

Jumps existed (from nearly 45% to nearly 100% for each model) in the attack detection rate as the threshold decreased. This may have been the result of insufficient training datasets, which made the models incomplete, which causes model outputs to be close, and consequently could not be classified with any threshold. Besides, although we recognized that most attacks would reveal sufficient differences by presenting in frequency of system calls, it is still likely that some attacks might be similar to some normal process. Indeed, the jumps indicated above might also be the results of such an occurrence.

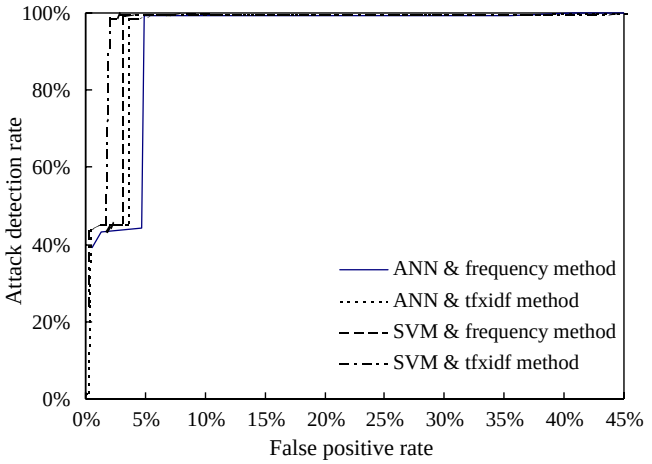

Fig. 12. ROC curves of four models.

Table 6
Statistic paired t-tests between four models

	SVMT–SVMF	SVMF–ANNT	ANNT–ANNF
Means	0.11	0.07	0.12
Standard deviation	0.21	0.18	0.22
t-value	3.63	2.69	3.75
p-value	<0.001	<0.01	<0.001

SVMT denotes SVM with the $tf \times idf$ encoding method; SVMF denotes SVM with the frequency encoding method; ANNT denotes ANN with the $tf \times idf$ encoding method; ANNF denotes ANN with the frequency encoding method.

5. Conclusion

Electronic commerce and the recent online consumer boom have forced a change in the basic computer security design for systems on a shared network. Systems are now designed with more flexibility and less barrier security. The policies that balance convenience versus strict system control and information access also make it impossible for an operational system to be completely secure. Thus, intrusion detection systems have become another essential component of computer security in detecting attacks before they inflict widespread damage [35]. Intrusion detection systems offer the potential advantages of reducing the manpower needed in monitoring, increasing detection efficiency, providing data that would otherwise not be available, helping the information security community learn about new vulnerabilities and providing legal evidence.

Some approaches that could identify anomaly attacks already exist, such as using short sequences of system calls [7], Markov process [4], rule-based algorithms (RIPPER) [20], etc. However, the very success of any approach for intrusion detection usually leads to its own downfall because attackers will change their behavior to evade its detection [3]. Thus, there is still a need for new ways to identify anomaly attacks. In this study, we applied the ANN and SVM algorithms for

intrusion detection. Our preliminary experiments with the 1998 DARPA BSM audit data showed that both approaches are able to effectively detect intrusive program behavior. Instead of using a conventional sequence-based encoding scheme, we employed a frequency-based encoding method. The main strength of the frequency-based scheme over the sequence-based method is a reduction in overhead by checking attacks at the end of the process. There is no need to build a profile for each program and check every sequence of the system calls.

Our result indicated that SVM performance was superior to that of ANN and the $tf \times idf$ encoding method is better than the simple frequency-based method. The superior performance of SVMs over ANNs is due to the following reasons: (1) SVMs implement the structural risk minimization principle which minimizes an upper bound for the generalization error rather than minimizing the training error. However, ANNs implement the empirical risk minimization principle, which might lead to worse generalization than SVM. (2) An ANN may not converge to global solutions due to its inherent algorithm design. In contrast, finding solutions in SVMs is equivalent to solving a linearly constrained quadratic programming problem, which leads to a global optimal solution. (3) In choosing parameters, SVMs are less complex than ANNs. The parameters that must be determined in SVMs are the kernel bandwidth δ^2 and the margin C. However, in ANNs, the number of hidden layers, number of hidden nodes, transfer functions and so on must be determined. Improper parameter selection might cause the over-fitting problem.

The reason that the $tf \times idf$ encoding method is better than the simple frequency-based encoding method is that the $tf \times idf$ encoding method not only stresses the system call frequency (i.e., tf), but also the uniqueness of a system call (i.e., idf). By giving higher weight to the uniqueness of a system call, intrusion detection systems can better distinguish normal and abnormal processes.

In future research, other data mining techniques such as genetic algorithms, case-based reasoning, decision tree and inductive learning may be applied to intrusion detection systems. Comparisons of various data mining techniques will provide clues for selecting appropriate models for detecting intrusions.

References

[1] Durst R, Champion T, Witten B, Miller E, Spagnuolo L. Testing and evaluating computer intrusion detection systems. Communications of the ACM 1999;42:53–61.

[2] Joo D, Hong T, Han I. The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors. Expert Systems with Applications 2003;25:69–75.

[3] Wagner D, Soto P. Mimicry attacks on host-based intrusion detection systems. Proceedings of the ninth ACM Conference on Computer and Communications Security, 2002.

[4] Warrender C, Forrest S, Pearlmutter B. Detecting intrusions using system calls: alternative data models. Proceedings of 1999 IEEE Symposium on Security and Privacy, 1999.

[5] Lee W, Stolfo SJ. Data mining approaches for intrusion detection. Proceedings of the seventh USENIX Security Symposium, 1998.

[6] Lee W, Stolfo SJ, Chan PK. Learning patterns from Unix process execution traces for intrusion detection. Proceedings of AAAI97 Workshop on AI Methods in Fraud and Risk Management, 1997.

[7] Forrest S, Hofmeyr SA, Somayaji A, Longstaff TA. A sense of self for UNIX processes. Proceedings of the 1996 IEEE Symposium on Security and Privacy, 1996.

[8] Helmer G, Wong JSK, Honavar V, Miller L. Automated discovery of concise predictive rules for intrusion detection. The Journal of Systems and Software 2002;60:165–75.

[9] Ghosh AK, Schwartzbard A, Schatz M. Learning program behavior profiles for intrusion detection. First Workshop on Intrusion Detection and Network Monitoring, 1999.
[10] Liu Z, Florez G, Bridges SM. A comparison of input representations in neural networks: a case study in intrusion detection. Proceedings of the 2002 International Joint Conference on Neural Networks, 2002.
[11] Liao Y, Vemuri VR. Use of K-nearest neighbor classifier for intrusion detection. Computers Security 2002;21: 439–48.
[12] Mukkamala S, Janoski G, Sung A. Intrusion detection using neural networks and support vector machines. Proceedings of IEEE International Joint Conference on Neural Networks, 2002.
[13] Mukkamala S, Janoski G, Sung A. Intrusion detection using support vector machines. Proceedings of the High Performance Computing Symposium—HPC 2002.
[14] Verwoerd T, Hunt R. Intrusion detection techniques and approaches. Computer Communications 2002;25:1356–65.
[15] Soto P. The new economy needs new security solutions. http://www.xcf.berkeley.edu/~paolo/ids.html 2001.
[16] Ning P, Jajodia S, Want XS. Design and implementation of a decentralized prototype system for detecting distributed attacks. Computer Communications 2002;25:1374–91.
[17] Biermann E, Cloete E, Venter LM. A comparison of intrusion detection systems. Computers Security 2001;20: 676–83.
[18] Denning DE. An intrusion-detection model. IEEE Transactions on Software Engineering 1987;13:222–32.
[19] Forrest S, Hofmeyr SA, Somayaji A. Computer immunology. Communications of the ACM 1997;40:88–96.
[20] Cohen WW. Fast effective rule induction. Proceedings of the 12th International Conference on Machine Learning, 1995.
[21] Lam KY, Hui L, Chung SL. A data reduction method for intrusion detection. Systems Software 1996;33:101–8.
[22] Helmer G, Liepins G. Statistical foundations of audit trail analysis for the detection of computer misuse. IEEE Transactions on Software Engineering 1993;19:866–901.
[23] Smith KA, Gupta JND. Neural networks in business: techniques and applications for the operations research. Computers & Operations Research 2000;27:1023–44.
[24] Srivarre-Ratana C, Konak A, Smith AE. Estimation of all-terminal network reliability using an artificial neural network. Computers & Operations Research 2002;29:849–68.
[25] Condon EM, Golden BL, Wasil EA. Predicting the success of nations at the Summer Olympics using neural networks. Computers & Operations Research 1999;26:1243–65.
[26] Salchenberger L, Venta ER, Venta LA. Using neural networks to aid the diagnosis of breast implant rupture. Computers & Operations Research 1997;24:435–44.
[27] Vapnik VN. The nature of statistical learning theory. New York: Springer; 1995.
[28] Burges CJC. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 1998;2:1–47.
[29] Cristianini N, Shawe-Taylor J. An introduction to support vector machines and other kernel-based learning methods. Cambridge, UK: Cambridge University Press; 2000.
[30] Cai Y-D, Lin X-J, Xu X-B, Chou K-C. Prediction of protein structural classes by support vector machines. Computers and Chemistry 2002;26:293–6.
[31] Morris CW, Autret A, Boddy L. Support vector machines for identifying organisms—a comparison with strongly partitioned radial basis function networks. Ecological Modeling 2001;146:57–67.
[32] Tay FEH, Cao L. Application of support vector machines in financial time series forecasting. Omega 2001;29: 309–17.
[33] Salton G. Automatic text processing. Reading, MA: Addison-Wesley; 1989.
[34] Basu A, Watters C, Shepherd M. Support vector machines for text categorization. IEEE Proceedings of the 36th Hawaii International Conference on System Sciences, 2002.
[35] Lippmann RP, Fried DJ, Graf I, Haines JW, Kendall KR, McClung D, Weber D, Webster SE, Wyschogrod D, Cunningham RK, Zissman MA. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. Proceedings of the 2000 DARPA Information Survivability Conference and Exposition, 2000.