

Available online at www.sciencedirect.com



<u>computer</u> communications

Computer Communications 31 (2008) 58-72

www.elsevier.com/locate/comcom

Processing of massive audit data streams for real-time anomaly intrusion detection

Wei Wang ^{a,*}, Xiaohong Guan ^{a,b}, Xiangliang Zhang ^a

^a State Key Laboratory for Manufacturing Systems (SKLMS) and MOE Key Lab for Intelligent Networks and Network Security (KLINNS), Xi'an Jiaotong University, Xi'an 710049, China

^b Center for Intelligent and Networked Systems, Tsinghua University, Beijing 100080, China

Received 26 October 2006; accepted 2 October 2007 Available online 13 October 2007

Abstract

Intrusion detection is an important technique in the defense-in-depth network security framework. Most current intrusion detection models lack the ability to process massive audit data streams for real-time anomaly detection. In this paper, we present an effective anomaly intrusion detection model based on Principal Component Analysis (PCA). The model is more suitable for high speed processing of massive data streams in real-time from various data sources by considering the frequency property of audit events than by use of the transition property or the correlation property. It can serve as a general framework that a practical Intrusion Detection Systems (IDS) can be implemented in various computing environments. In this method, a multi-pronged anomaly detection model is used to monitor various computer system and network behaviors. Three sources of data, system call data from the University of New Mexico (*lpr*) and from KLINNS Lab of Xi'an Jiaotong University (*ftp*), shell command data from AT&T Research laboratory, and network data from MIT Lincoln Lab, are used to validate the model and the method. The frequencies of individual system calls generated by one process and of individual commands embedded in one command block as well as features extracted in one network connection are transformed into an input data vector. Our method is employed to reduce the high dimensional data vectors and thus the detection is handled in a lower dimension with high efficiency and low use of system resources. The distance between a vector and its reconstruction in the reduced subspace is used for anomaly detection. Empirical results show that our model is promising in terms of detection accuracy and computational efficiency, and thus amenable for real-time intrusion detection.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Intrusion detection; Principal Component Analysis; Hidden Markov models; Network security; Data streams

1. Introduction

Network-borne attacks are currently major threats to information security. With the rapid growth of unauthorized activities on the network, Intrusion Detection Systems (IDS) have become very important. Intrusion detection is a technology for detecting hostile attacks against computer network systems, both from outside and inside. In general, the techniques for intrusion detection fall into two major categories depending on the modeling methods used: signature-based detection and anomaly detection. Signature-based detection identifies malicious behavior by matching it against predefined description of attacks, or signatures. Although signature-based detection is effective against known intrusion types, it cannot detect new attacks that were not predefined. Anomaly detection, on the other hand, defines a normal profile of a subject's normal activities (a normal profile) and attempts to identify any unacceptable deviation as possibly the result of an attack. Anomaly detection may be able to detect new attacks. However, it may also cause a significant number of false alarms because the normal behavior varies widely

^{*} Corresponding author. Tel.: +33 2 99127045.

E-mail addresses: wei.wang.email@gmail.com (W. Wang), xhguan@ tsinghua.edu.cn (X. Guan), xlzhang@lri.fr (X. Zhang).

^{0140-3664/\$ -} see front matter @ 2007 Elsevier B.V. All rights reserved. doi:10.1016/j.comcom.2007.10.010

and obtaining complete descriptions of normal behavior is often difficult.

Anomaly detection has been an active research area for more than a decade since it was originally proposed by Denning [1]. Many methods have been developed for anomaly detection, such as machine learning, data mining, neural networks, statistical methodology. There are multiple prongs that anomaly detection models can be built upon in real computer network systems. Many sources of data are then used for anomaly detection, including shell commands, audit events, keystroke records, system calls and network packets. Early studies [2–4] on anomaly detection mainly focused on modeling system or user behavior from monitored system log or accounting log data, including CPU usage, time of login, duration of user sessions and names of files accessed. Schonlau and Theus [5] attempts to detect masquerades by building normal user behavior using truncated command sequences. Experiments with six masquerade detection techniques [6]: Bayes one-step Markov, Markov, Hvbrid multi-step IPAM. Uniqueness, Sequence-Match and Compression, were performed and compared. Maxion and Townsend [7] applied the Naïve Bayes classification algorithm to detect masquerade based on the same data. Lane and Brodley [8] proposed a learning algorithm for analyzing user shell command history to build normal user behavior and detect anomalies. It attempts to address the "concept drift" problem when the normal user behavior changes. Recently Oka et al. [9] used layered networks for masquerade detection based on Eigen Co-occurrence Matrix (ECM).

In recent years, a lot of research activities focused on learning program behavior and building profiles with system call sequences as data sources. In 1996, Forrest et al. [10] introduced a simple anomaly detection method called time-delay embedding (tide), based on monitoring system calls invoked by active and privileged processes. Profiles of normal behavior were built by enumerating all fixed length of distinct and contiguous system calls that occur in the training data sets and unmatched sequences in actual detection are considered anomalous. In subsequent research, the approach is extended by various methods. For example, Lee and Stolfo [11] used data mining approach to study a sample of system call data and characterize the sequences contained in normal data by a small set of rules. The sequences violating those rules were then treated as anomalies for monitoring and detection purpose. Warrender et al. [12] proposed a Hidden Markov Model (HMM) based method for modeling and evaluating invisible events. This method was further studied by many other researchers [13-16]. Wespi et al. [17] extended Forrest's idea and proposed a variable length approach. Asaka et al. [18] developed an approach based on the discriminant method in which an optimal classification surface was first learned from samples of properly labeled normal and abnormal system call sequences. The surface was then used as a basis for deciding the normality of a new system call sequence. Yeung and Ding [14] and Lee et al. [19] used information-theoretic measures for anomaly detection. Liao et al. [20] used K-Nearest Neighbor (K-NN) classifier and Wu et al. [21] applied robust Support Vector Machines (SVM) for intrusion detection based on system call data to model program behavior and classified each process as normal or abnormal when it terminated. Cho [22] applied soft computing techniques for anomaly detection. Our research group [23] employed Rough Set Theory (RST) to learn and model normal behavior with improved detection accuracy while using much smaller size of training data sets. Our research group has also developed another two intrusion detection methods. One [24] is based on plan recognition for predicting intrusion intentions by observing system calls and the other [25] is based on Non-negative Matrix Factorization (NMF) to profile program and user behavior. Other methods or techniques, such as first-order Markov Chain Models [26], high-order Markov Models [27], EWMA [28], Decision tree [29], Chi-Square [30] and Neural Networks [31] are also used for system call based intrusion detection.

In a multi-layered or multi-pronged IDS, monitoring network traffic behavior is as important as monitoring program behavior and user behavior. EMERALD [32] used statistical anomaly detection modules to monitor network traffic. Lee et al. [33,34] extracted features from network data and built signature-based detection models. The detection models generalized rules that classify the data with the values of the extracted features. Liu et al. [35] proposed a new Genetic Clustering (GC) based anomaly detection algorithm. The method can establish clusters and detect network intrusions by labeling normal and abnormal groups. Eskin et al. [36] proposed a geometric framework for unsupervised anomaly detection and three algorithms, Cluster, K-Nearest Neighbor (K-NN) and Support Vector Machine (SVM), were used for classification. Shyu et al. [37] proposed a Principal Component Classifier (PCC) for network intrusion detection. They measured the Mahalanobis distance of each observation from the center of the data for anomaly detection. The Mahalanobis distance is computed based on the sum of squares of the standardized principal component scores. Heywood et al. [38] used a hierarchical neural network approach based on Self Organizing Maps (SOM) and potential function clustering for network intrusion detection. Sarasamma et al. [39] proposed a multilevel Hierarchical Kohonenen Net (HKN) for anomaly network intrusion detection. Each level of the hierarchical map is modeled as a simple winner-take-all Kohonenen net. Other methods, such as neural networks [40] and fusion of multiple neural network classifiers [41], have also been applied for network intrusion detection.

Although existing efforts in anomaly detection have made impressive progress, there are still many issues to be resolved. First, a computer system in daily operation can produce massive data streams from various data sources. Moreover, these sources of data are typically high dimensional. For example, in collecting system calls of sendmail on a host machine, only 112 messages produced a combined trace with the length of over 1.5 million system calls [10]. Each trace of the data may contain about more than 40 distinct system calls, resulting a high dimensional data set. And in experiments carried out by MIT Lincoln Lab for the 1998 DARPA evaluation [42], network traffic over 7 weeks contains 5 GB of compressed binary *tcpdump* data which were processed into about five million connection records. Similarly, the network data is high dimensional as each network connection contains 41 features. Given these figures, high speed processing of high dimensional massive audit data in most cases is essential for a practical IDS so that actions for response can be taken as soon as possible.

However, many current anomaly detection models make the implicit assumption that data is relatively low dimensional, or only a small amount of data is used. This oversimplification limits the effectiveness of these models. On the other hand, many intrusion detection models require too much time to train the models by processing a large amount of data. For example, it took Hidden Markov Models (HMM) approximately two months to train an anomaly detection model with a large data set [12]. Since computing environments change rapidly, an effective intrusion detection model should be periodically retrained to achieve real-time self-adaptive intrusion detection. Spending too much time for training is clearly not adequate for this purpose. Applicability on various data sources is another issue. Many IDSs can only handle one particular audit data source [33,34]. For example, system call based intrusion detection methods may not be applied to other sources of data, such as shell command data or network data. Since activities at different penetration points are normally recorded in different audit data sources, an IDS often needs to be extended to incorporate additional modules that specialize on certain components (e.g., hosts, subnets, etc.) of the network system [33]. Therefore, it is crucial to develop and build a general framework that practical IDSs can be implemented on various data sources.

An IDS is a recognition system in nature. To achieve effective real-time anomaly intrusion detection, it is crucial to choose and extract suitable features and to design effective classification algorithms. In practice, it is always a challenge to choose features that best characterize behavioral patterns of a subject (e.g., a program, a user or a network element, etc.), so that abnormality can be clearly distinguished from normal activities. In general, there are three categories of attributes of activities in computer systems: the transition property, frequency property and correlation property of audit events. These three properties of audit events have been widely studied for intrusion detection.

The intrusion detection methods considering the transition property of audit events extract the transition information between the elements in the audit data. These methods often use sliding windows to divide the data into short sequences for data preparation. These methods include [8,10–17,23,24] and [26,27] in the literature. The intrusion detection methods taking into account the frequency property of audit events compute the distribution of the audit data. These methods do not focus on temporal variations in the data. Some methods also use sliding windows to partition the data into short sequences while other methods do not for data preparation. These methods include [3,5,7,14,20,21,25] in the literature. There are also intrusion detection methods using the correlation property of audit events. These methods capture the correlation information embedded in the audit data. Ref. [9] is an example for capturing the user behaviors by correlating not only connected events but also events that are not adjacent to each other.

In this paper, we firstly conduct experiments to evaluate the relationship between the performance of intrusion detection and the properties of audit events which are considered. Comparative studies are then used as references to choose suitable features of audit events for real-time intrusion detection. Hidden Markov Models (HMM) are usually used for modeling temporal variation in the data. We then use this method to consider the transition property of audit events for intrusion detection. In this paper, we propose a Principal Component Analysis (PCA) method to take into account the frequency property of audit events for intrusion detection. We also compare the testing results of PCA method with the results obtained by using HMM method and ECM method considering the correlation information of audit events.

The novelty of our work lies in the following three aspects. First, the relationships between the performance of intrusion detection and the properties of audit events considered are evaluated. The evaluation results can be used as an important reference for effective real-time intrusion detection. Second, the proposed PCA based intrusion detection method can achieve real-time intrusion detection based on dimensionality reduction and on a simple classifier. Third, the proposed method can also serve as a general framework that practical IDSs can be implemented in various environments based on its flexibility for processing various kinds of data streams. The testing results show that considering the frequency property of audit events is more suitable for real-time intrusion detection, providing adequate detection performance at low computational overhead, comparing to using the transition and correlation property of audit events. The PCA-based intrusion detection method is effective and suitable for high speed processing of massive data streams from various data sources. In this method, a multi-pronged intrusion detection model is used to monitor various computer network behaviors and three sources of data are used to validate the model and the method. Empirical results show that the method is promising in terms of detection accuracy and computational efficiency, and thus amenable for real-time intrusion detection.

The remainder of this paper is organized as follows: Section 2 describes the HMM method considering the transition properties and the related method using the correlation properties of audit events for intrusion detection. Section 3 provides a brief introduction to PCA and describes the proposed intrusion detection model using the frequencies of audit events. Empirical results on three sources of data are shown and analyzed in Section 4 to illustrate the effectiveness and efficiency of the proposed method. The concluding remarks follow in Section 5.

2. Intrusion detection methods based on the transition properties and correlation properties of audit events

2.1. HMM-based intrusion detection method considering the transition properties of audit events

HMMs are dynamic models widely used for considering the transition property of events. An HMM describes a doubly stochastic process. Each HMM contains a finite number of unobservable (or hidden) states. Transitions among the states are governed by a set of probabilities called transition probabilities. An HMM defines two concurrent stochastic processes: the sequence of the HMM states and a set of state output processes. Given an input sequence of observations $O(O_1, \dots, O_t)$, an HMM can model this sequence by three parameters – state transition probability distribution A, observation symbol probability distribution B and initial state distribution π [43,44]. Therefore, a sequence can be modeled as $\lambda = (A, B, \pi)$ using its characteristic parameters.

There are three central issues in HMMs including the evaluation problem, the decoding problem, and the learning problem. Given an HMM model λ and a sequence of observations $O(O_1, \dots, O_t)$, the evaluation problem is to evaluate $P(O|\lambda)$, the probability that the observations are generated by the model. The decoding problem, on the other hand, is to decide the most likely state sequence that produced the observations. The learning problem is a training process and therefore very important. It is to maximize $P(O|\lambda)$ by adjusting the parameters of the model λ .

HMMs learning can be conducted by the *Baum-Welch* (BW) or *forward-backward algorithm* – an example of a generalized *Expectation-Maximization* (EM) algorithm [45].

Standard HMMs have a fixed number of states, so we must decide the size of the model before training. Previous research indicates that a good choice for the application is to choose a number of states roughly corresponding to the number of distinct elements in the audit data [12]. The number of the states therefore depends on the data set used in experiments.

After the HMMs were learned on the training set of the data, normal behavior is thus profiled by the parameters of the HMMs $\lambda = (A, B, \pi)$.

Given a test sequence of the data (length S), we use a sliding window of length L to move along the trace and get (S - L + 1) short sequences of the data $O_i(1 \le i \le S - L + 1)$.

Using the normal model $\lambda = (A, B, \pi)$ which was built by the learning method described above, the probability that a given observation sequence O is generated from the model can be evaluated using the *forward algorithm* [45]. In the experiments, we use log-probability log- $P(O|\lambda)$ instead of $P(O|\lambda)$ to increase the scale of the probability.

Ideally, a well-trained HMM can give sufficiently high likelihood only for sequences that correspond to normal behavior. Sequences that correspond to abnormal behavior, on the other hand, should give significantly lower likelihood values [14]. The HMM based anomaly detection method in this paper is based on this property.

Given a predetermined threshold ε_a with which we compare the probability of a sequence O in a test trace, if the probability is below the threshold, the sequence O is then flagged as a *mismatch*. We sum up the mismatches and define the *anomaly index* as the ratio between the numbers of the mismatches and of all the sequences in the test trace. The classification rule is thus assigned as following equation:

Anomaly index

$$=\frac{\text{Numbers of the mismatches}}{\text{Numbers of all the sequences in the test trace}} > \varepsilon_{\text{h}} \quad (1)$$

If (1) is met, then the trace embedding the test sequences is considered as a possible intrusion. Otherwise it is considered as normal.

2.1.1. Data sets

In the experiments, we use system call data to evaluate the HMM-based intrusion detection model. In this paper, we do not consider the arguments to system calls that would supply additional information (e.g., [46,47]). In order to assess the model, we used two data sets in the experiments for profiling program behaviors. One is *lpr* data collected in MIT Lincoln Lab and also by Warrender et al. [12]. The data set can be downloaded at http:// www.cs.unm.edu/~immsec and the procedures for generating the data are also described on the website. The data set includes 2703 traces of normal data and 1001 traces of intrusion data. We used the first 600 traces of normal data and the first 300 traces of intrusion data in the experiments.

The other data set was collected in the actual system in our KLINNS lab of Xi'an Jiaotong University. We collected live *ftp* system call sequences on a Red Hat Linux system with kernel 2.4.7-10, spanning a time period of about 2 weeks. The normal data are traces of system calls generated by authentic users with various conditions. Intrusion traces are associated with exploitations against a widely known Wu-Ftpd vulnerability [48], which allows remote attackers to execute arbitrary commands on the victim host. The *ftp* system call data generated in the experiments includes 549 normal traces and six intrusion traces. The statistics of the system call data used in the experiments are shown in Table 1.

6	2
υ	4

Table 1 Descriptions of system call data in the experiments

Data set	Number of system calls	Number of distinct system calls	Number of normal traces (processes)	Number of intrusion traces (processes)
MIT <i>lpr</i> data	842,279	41	600	300
Live <i>ftp</i> data from KLINNS Lab	5,699,277	58	549	6

2.1.2. Testing results based on the HMM method

In the experiments, we group the system calls that are invoked by the same process into one trace and classify whether the process behavior is normal or not.

For *lpr* data, 200 traces of the normal data are randomly selected for training and other data, 400 traces of the normal data and 300 traces of the intrusion data are used for detection. For live *ftp* data, 70 traces of the normal data are randomly selected for training and other data, 479 traces of normal data and six traces of intrusion data are used for detection. In the training set of the normal data, there are 41 distinct system calls in *lpr* data and 58 distinct system calls in live *ftp* data, respectively. We therefore use 41 states and 58 states of HMMs in the experiments for profiling the program behavior of *lpr* and *ftp* accordingly.

In our experiments, each trace of the system call data is first converted into short sequences of fixed lengths. We use window sizes as 3 and 6, respectively, in the experiments for comparison of the testing results. The average anomaly indexes of the normal and intrusion traces of these two data sets are calculated respectively and summarized in Table 2 and the False Alarm Rates (FAR) and Detection Rates (DR) are summarized in Table 3. The CPU time required for training and detection of *lpr* data is also summarized in next section for clear comparison with other methods.

From Tables 2 and 3, it is observed that: (1) the anomaly indexes of abnormal traces are significantly higher than those of the normal data for both of the two data sets and the HMM based method is thus an effective method for intrusion detection. (2) Different window sizes result in different anomaly indexes of each trace of the data. As window size increases, the anomaly index corresponding to the trace of the data tends to increase too. This is not strange because a large sliding window contains more tran-

Table 2

The average anomaly indexes of normal and abnormal traces of the two sets of system call data

System call sequences	Anomaly indexes (%)				
	Window size $= 3$	Window size $= 6$			
Lpr data					
Intrusion	5.9524	10.3659			
Normal	1.7736	1.8665			
Live <i>ftp</i> data					
Intrusion	11.42	16.25			
Normal	0.1021	0.1978			

Та	ble	3	
1 a	oic	5	

The	detection	rates and	false alarm	rates of t	the two	sets of	system	call c	lata
1 110	detection	rates and	Taise alarm	Tates of	the two	sets of	system	can	iata

Data set	Window size	ε_{a}	$\varepsilon_{\rm h}~(\%)$	FAR (%)	DR (%)
Lpr data	3	$-32 \\ -32$	2.48 5.85	5.5	100 99 7
	6	$-40 \\ -40$	3.16 6.55	2.3 0.5	100 99.7
Live <i>ftp</i> data	3 6	-40 -70	2.03 4.07	0 0	100 100

sition information between the system calls and this is valuable for classification. The performance of the intrusion detection is related to the window sizes and thus the choice of window size should be addressed in practical IDSs.

2.2. The method considering the correlation properties of audit events

There are few intrusion detection methods that consider the correlation properties of audit events. Recently, Oka et al. [9] proposed an Eigen Co-occurrence Matrix (ECM) method correlating not only connected events but also events that are not adjacent to each other while appearing within a certain distance. Their method created a so-called "co-occurrence matrix" by correlating a command in a sequence with any following commands that appear within a certain distance. User behavior was then built based on the "eigen co-occurrence matrix" created by extracting principal features of the "co-occurrence". The ECM method is typically an intrusion detection method that considers the correlation property of audit events. In this paper, we compare the testing results of this method with those of our method.

3. The proposed intrusion detection method based on Principal Component Analysis

3.1. Principal Component Analysis

Principal Component Analysis (PCA, also called Karhunen-Loève transform) is one of the most widely used dimension reduction techniques for data analysis and compression. It is based on transforming a relatively large number of variables into a smaller number of uncorrelated variables by finding a few orthogonal linear combinations of the original variables with the largest variance. The first principal component of the transformation is the linear combination of the original variables with the largest variance; the second principal component is the linear combination of the original variables with the second largest variance and orthogonal to the first principal component and so on. In many data sets, the first several principal components contribute most of the variance in the original data set, so that the rest can be disregarded with minimal loss of the variance for dimension reduction of the data [45,51]. PCA has been successfully applied in many areas, such as face recognition [50], image processing, text categorization, gene expression analysis and so on. The transformation works as follows.

Given a set of observations be $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, suppose each observation is represented by a row vector of length m. The data set is thus represented by a matrix $X_{n \times m}$

$$X_{n \times m} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} = [\mathbf{x}_{1}, \mathbf{x}_{2}, \cdots, \mathbf{x}_{n}]$$
(2)

The average observation is defined as

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \tag{3}$$

Observation deviation from the average is defined as

$$\mathbf{\Phi}_i = \mathbf{x}_i - \boldsymbol{\mu} \tag{4}$$

The sample covariance matrix of the data set is defined as

$$C = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu}) (\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}} = \frac{1}{n} sum_{i=1}^{n} \boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^{\mathrm{T}} = \frac{1}{n} A A^{\mathrm{T}}$$
(5)

where $A = [\mathbf{\Phi}_1, \mathbf{\Phi}_2, \cdots, \mathbf{\Phi}_n].$

To apply PCA, eigenvalues and corresponding eigenvectors of the sample covariance matrix C are usually computed by the Singular Value Decomposition (SVD) theorem [51]. Suppose $(\lambda_1, \mathbf{u}_1), (\lambda_2, \mathbf{u}_2), \dots, (\lambda_m, \mathbf{u}_m)$ are meigenvalue-eigenvector pairs of the sample covariance matrix C. We choose k eigenvectors having the largest eigenvalues. Often there will be just a few large eigenvalues, and this implies that k is the inherent dimensionality of the subspace governing the "signal" while the remaining (m - k) dimensions generally contain noise [45]. The dimensionality of the subspace k can be determined by [49].

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \ge \alpha \tag{6}$$

where α is the ratio of variation in the subspace to the total variation in the original space. If α is chosen as 99.9%, then variation in the subspace spanned by the former k eigenvectors has only 0.1% loss of variation in the original space. We form a $m \times k$ matrix U whose columns consist of the k eigenvectors. The representation of the data by principal components consists of projecting the data onto the k-dimensional subspace according to the following rules [45].

$$\mathbf{y}_i = U^{\mathrm{T}}(\mathbf{x}_i - \boldsymbol{\mu}) = U^{\mathrm{T}} \boldsymbol{\Phi}_i \tag{7}$$

3.2. Intrusion detection model based on PCA

In order to detect intrusions across-the-board, a typical multi-pronged IDS is proposed and shown in Fig. 1.

In the multi-pronged IDS, the behaviors of a networked computer system are monitored according to the impact order of the attacks and divided into three prongs including network behavior, user behavior and system behavior. Usually various methods are required to process network



Fig. 1. A multi-pronged IDS.

packets, keystroke records, file system, command sequences, system calls, etc., of audit data streams obtained in the three prongs for intrusion detection.

In this paper, we propose a general framework for building intrusion detection models for a multi-pronged IDS. This model is then tested in four experiments using three sources of data: system call data, command data and network connection data. Building the intrusion detection model includes three steps: data preparation, dimension reduction and feature extraction, and classification.

3.2.1. Data preparation

In data preparation, each source of the observation data set is divided into smaller data blocks using a specified scheme. For example, the system call data is divided by processes, network data by connections and shell command data is divided into consecutive blocks with a fixed length. Instead of using the transition information of the data, we use the frequencies of system call data to characterize program behavior and shell command data to charaacterize user behavior. For network data, we use the features extracted from a network connection to characterize network traffic behavior.

To clearly show the detailed data preparation step, we give an example of the data preparation method for system call data. The system calls invoked by the same process are firstly grouped into one trace representing each process in the data. For example, the system call sequence invoked by the Process 7174 in*lpr* data are shown below.

Process ID: 7174

In this sequence, each system call is represented by a number. The mapping between a system call number and the actual system call name is given by a separate table. For example, the number 5 represents system call "open", the number 3 represents system call "read". Instead of using short sequences of system calls used by most intrusion detection methods [10–18,23,24] and by the HMM method proposed in Section 2.1, we use each trace of the data as observation. In the Linux/Unix environment, execution of a program can generate one or more processes. Each process produces a single trace of system calls from the beginning of its execution to the end. Therefore, by treating each trace of the data as observation, program behavior can be profiled for anomaly detection. In each trace of the data, the frequencies of individual system calls are calculated. For example, the frequency of number 5 in the process 7174 is 0.086. Each trace of system call data is thus transformed into a data vector and the matrix representing a system call data set is shown below.

Distinct						
system call	1	2		156	168	
Trace1	0.006	0.005	•••	0.006	0.056	
Trace2	0.110	0.111	•••	0.112	0.108	
<	÷	•	÷	÷	÷	, <i>П</i>
Trace n-1	0.012	0.013	•••	0.014	0	
Trace n	0.006	0.005	•••	0.005	0.051	

Suppose an observation data set is divided into *n* blocks, and there are a total of *m* distinct elements (e.g., system call data and command data) or features (e.g., network data) in the data set. The observed data can be expressed by *n* vectors with each vector containing *m* distinct observations. A $n \times m$ Matrix X, where each element X_{ij} stands for the frequency of *j*th distinct element (e.g., system call data and command data) or feature (e.g., network data) occurs in the *i*th block, is then constructed. The observed data set that is represented by a matrix $X_{n \times m}$ can be written as Eq. (2), where row vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ represent the corresponding blocks of the original data.

3.2.2. Dimension reduction and feature extraction

Given a training set of data vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, the average vector $\boldsymbol{\mu}$ and each mean-adjusted vector can be computed by (3) and (4). *m* eigenvalue-eigenvector pairs $(\lambda_1, \mathbf{u}_1), (\lambda_2, \mathbf{u}_2), \dots, (\lambda_m, \mathbf{u}_m)$ of the sample covariance matrix *C* are then calculated.

The size of principal eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k (k \ll m)$, used to represent the distribution of the original data, is often determined by (6). Any data vector of the training set can be represented by a linear combination of k eigenvectors so that the dimensionality of the data is reduced and the features of the data are extracted.

3.2.3. Classification

A test data vectortwhich represents a test block of data can be projected onto the k-dimensional subspace according to the rules defined by (7). The distance between the test data vector and its reconstruction in the subspace is simply the distance between the mean-adjusted input data vector $\mathbf{\Phi} = \mathbf{t} - \boldsymbol{\mu}$ and

$$\mathbf{\Phi}_f = U U^{\mathrm{T}}(\mathbf{t} - \boldsymbol{\mu}) = U \mathbf{y} \tag{8}$$

If the test data vector is normal, that is, if the test data vector is very similar to the training vectors corresponding to normal behavior, the test data vector and its reconstruction will be very similar and the distance between them will be very small [49,50]. Based on this property, normal program, user and network behaviors can all be profiled for anomaly detection [52,53]. In the experiments presented here, three measures, squared Euclidean distance ε_e , Cosine distance ε_c and Signal-to-Noise Ratio (SNR) ε_s , are used to map the distance or similarity of these two vectors in order to compare the testing results:

$$\varepsilon_{\rm e} = \|\mathbf{\Phi} - \mathbf{\Phi}_f\|^2 \tag{9}$$

$$\varepsilon_{\rm c} = \frac{\Psi \Phi_f}{\|\Phi\| \|\Phi_f\|} \tag{10}$$

$$\varepsilon_{\rm s} = 10 \log \left(\frac{\|\mathbf{\Phi}\|^2}{\|\mathbf{\Phi} - \mathbf{\Phi}_f\|^2} \right) \tag{11}$$

In anomaly detection, ε_e , ε_c and ε_s are characterized as *anomaly indexes*. If ε_e and ε_c are below or ε_s is above a predetermined threshold, then the test data **t** is classified as normal. Otherwise it is treated as anomalous.

4. Experiments and testing

In our experiments, we used four data sets (which include three data sources), *lpr* system call data from the University of New Mexico and *ftp* system call data from the KLINNS lab of Xi'an Jiaotong University, shell command data from AT&T Research lab, and network connection data from MIT Lincoln Lab, to test the anomaly detection model.

4.1. Experiments on system call data

4.1.1. Data sets

In order to facilitate the comparison, we use the same system call data used for evaluating the HMM-based intrusion detection method in Section 2.1.

4.1.2. Testing results and analysis

The model gives good testing results. Fig. 2 shows the detection results on the *lpr* data using the squared Euclidean distance measure with 200 traces of data randomly selected from normal data for training and another 700 traces for detection. It is clear that abnormal data can be easily distinguished from normal data based on the *anomaly index*.

In the experiments, the ratio α , as defined in (6), is selected as 99.9% and the testing results including Detec-



Fig. 2. Testing results on the *lpr* system call data. The *y*-axis represents the *anomaly index* and *x*-axis represents the system call trace number. The stars (*) in the gray shading stand for abnormal traces and the dots (\bullet) with no shading stand for normal traces. The *y*-axis is expanded for readability.

tion Rates (DR) and False Alarm Rates (FAR) are summarized in Table 4 for comparison. It is observed that the detection results are the best in terms of DR and FAR with squared Euclidean distance measure. This is because PCA in nature seeks a projection that best represents the original data in a least-square sense. The results remain similar with Cosine distance and SNR.

To evaluate the impact of the rate α defined in (6) on the performance of the detection model, we conduct the testing with 200 normal data traces for training and squared Euclidean distance as anomaly index for anomaly detection. The testing results are shown in Table 5. It is seen that the false alarm rate is the lowest when $\alpha = 99.92\%$. It decreases first and then tends to increase with increase of ratioa. When ratioais relatively small, the variance in the data set cannot be adequately represented by the reduced subspace. Some valuable information in the original data may be discarded and this leads to relatively high false alarm rates. On the other hand, when ratio α is large enough and near to 100%, the reduced subspace contains noise [45,51] that reduces the effectiveness of the intrusion detection. Also, the threshold tends to be smaller and smaller with increase of ratio and this makes it difficult for detection. With these testing results, we suggest to use $\alpha = 99.9\%$ for feature extraction and squared Euclidean distance as *anomaly index* for anomaly detection in real environments. It can reduce the data largely with good testing result. We will also verify this suggestion with network data in Section 4.3.

The model is also effective for system call data from our KLINNS lab. Fig. 3 shows the testing results of our model on the *ftp* system call data with 70 normal data traces for training and another 485 data traces for testing, using $\alpha = 99.9\%$ and squared Euclidean distance measure. It is seen that six intrusion traces are all detected without any false alarms.

Comparing the testing results obtained by using PCA method with those of the HMM method (shown in Tables 2 and 3), it is observed that HMM is a better method than PCA in terms of detection accuracy on lpr data with window size 6. However, the detection accuracy of HMM on Live ftp data is the same as that of PCA. The HMM method detected all the intrusions with 2.3% false alarms rate on *lpr* data and without false alarms on the Live *ftp* data. On the other hand, while the PCA method successfully detected all the intrusions in the Live *ftp* data, it reached at 2.8% false alarms rate on the lpr data. These results show that focusing on the transition property of audit events can achieve better detection accuracy than focusing on that of the frequency property. However, using frequency property of audit events can also yield satisfactory results in intrusion detection.

Our method is computationally efficient. During the detection stage, the squared Euclidean distance between a test vector and its reconstruction onto the subspace is used for detection. Calculations for each test block of data take O(mk), where *m* is the dimension of each vector represent-

Table 4						
Detection 1	Rates (DR) and False	e Alarm Rates (FAR)	with different	distance or similarity measures	
a 1 E	1. 1	11.		a .		

Squared Euclidean distance measure		Cosine measure			SNR			
$\varepsilon_{\rm e}~(\times 10^{-3})$	FAR (%)	DR (%)	ε _c	FAR (%)	DR (%)	ε _s	FAR (%)	DR (%)
0.273	2.8	100	0.983	10.3	100	49.830	10.3	100
1.200	0.5	99.7	0.989	4.5	99.7	38.791	4.5	99.7

Table 5

Detection Rates (DR) and False Alarm Rates (FAR) using different dimensionality of reduced subspace

Dimensionality	Ratea (%)	$\varepsilon_{\rm e}~(\times 10^{-3})$	FAR (%)	DR (%)
of subspace k				
1	98.34	0.795	13.5	100
		5.200	1.3	99.7
2	99.22	0.606	10.3	100
		1.600	5.3	99.7
5	99.87	0.279	3.8	100
		1.300	0.8	99.7
6	99.92	0.273	2.8	100
		1.200	0.5	99.7
7	99.95	0.205	3	100
		1.100	0.5	99.7
10	99.99	0.135	3.3	100
		0.213	1.5	99.7
15	99.99	0.048	3.5	100
		0.123	1	99.7
32	99.99 ^a	6.3×10^{-6}	3.75	100
35	99.99 ^a	2.9×10^{-7}	3.75	100
36	100	3.21×10^{-29}	35	100
41	100	3.20×10^{-29}	35	100

^a Rate α here is very close to 100%.



Fig. 3. Testing results on the *ftp* system call data from our KLINNS Lab. The *y*-axis represents the *anomaly index* and *x*-axis represents the system call trace number. The stars (*) in the gray shading stand for abnormal traces and the dots (\bullet) with no shading stand for normal traces. The *y*-axis is expended for readability. The *y*-axis is expanded for readability.

ing each block of data and k is the number of principal components used in the model. Experimental results show that after the high-dimensional data is reduced, the original

data can be represented by the linear combination of only a very small number of principle components without sacrificing valuable information. In the experiments, for example, the use of only six principle components out of 41 dimensions can represent the original data with less than 0.1% loss of the total variation. Therefore, the original data can be largely reduced for intrusion detection and k is very small. Because the subspace is low dimensional and the classifier is simple, little computational effort is required for the detection. Moreover, system resources could be largely saved for low dimensional data which are conveniently stored and transmitted.

In the experiments, we evaluate the computational performance of our PCA model in comparison with the HMM method described in Section 2.1 and the tide method reported in [10] in terms of training time for building the models as well as test time for detection. The experiments are conducted on a 2.4-GHz Pentium computer with 512 MB DDR memory and the testing results are shown in Table 6. It is observed that only 5 s are required for PCA method versus up to 4632 s for training the same size of the data for HMM method. Tide is usually regarded as an efficient method for real-time intrusion detection [12]. However, it takes about 33 s for training, requiring more time than our PCA model. The results on detection times are consistent with those on the training time. The HMM method required about 949 s for detecting about 600 thousands system calls with window size as 3. By comparison, our PCA model only requires about 14 s versus about 356 s in the tide method for the same size of data.

Based on the comparative studies of these two intrusion detection methods discussed above, it shows that utilizing the transition property of audit events can produce a good detection performance only at high computational expense. Relying on the frequency property of events, on the other hand, is very suitable for real-time intrusion detection, providing adequate detection performance at very low compu-

Table 6							
Training and	detection	times	with	system	call	data	

Method	Number of system calls for training	CPU Time for training (s)	Number of system calls for detection	CPU Time for detection (s)
НММ	159,642	4632	682,637	949 (window size = 3) 1662 (window size = 6)
Tide	159,642	33	682,637	356
PCA	159,642	5	682,637	14

tational overhead. PCA method, considering the frequency property of the audit events, is thus very suitable for processing massive audit data streams for real-time intrusion detection. Most current intrusion detection methods considering the transition property of events firstly divide sequences of system calls by a fixed length of sliding window for data preparation. Detection performance is shown to be sensitive to window size [54]. As window size increases, the detection performance improves, but only at considerable computational expense. Moreover, the question of how to choose the window size has not been sufficiently addressed. The PCA method takes into account the frequency property of system calls. Processes are considered as observation. It then avoids deciding the size of sliding windows which are often chosen by experience. In the PCA method, each process is represented by a data vector, where each entry is the frequency of a distinct system call during the execution of a process. In this way, the anomaly intrusion detection problem is transformed into the simpler problem of classifying these vectors as normal or abnormal. PCA method uses a simple classifier and can achieve a good real-time detection performance.

4.2. Experiments on shell command data

4.2.1. Data sets

The shell command data that comes from a UNIX server at AT&T's Shannon Research Laboratory are used for testing. User names and associated command sequences (without arguments) make up the testing data available at http://www.schonlau.net/intrusion.html. Fifty users are included with 15,000 consecutive commands for each user divided into 150 blocks of 100 commands. The first 50 blocks are uncontaminated and used as training data. The masquerading command blocks, randomly drawn from outside of the 50 users, are inserted into the command sequences of the 50 users in the rest 100 blocks. The details of the contamination procedure can also be found on the website.

The goal of the experiments is to correctly detect masquerading blocks. Each data block of a user is transformed into a vector which contains the frequencies of individual commands embedded in the block. A test data vector representing a data block of a user is then used as data input for anomaly detection by (8) and (9).

4.2.2. Testing results and analysis

We conduct the experiments on the 50 users and the testing results of most users are promising. The testing results on User 24 are shown in Fig. 4 as an example. It is observed that simulated masquerade data are located at Blocks 69–89 with gray shading in Fig. 4 and our model can easily catch them all.

To use much data for evaluating the detection and computational performance of the model, we reconstruct the data for profiling one user behavior for anomaly detection in the experiments. We randomly select two data sets of



Fig. 4. Testing result of User 24. The *y*-axis represents the *anomaly index* and *x*-axis represents command block number. The stars (*) in the gray shading indicate simulated masquerades data and dots (\bullet) with no shading stand for normal data.

two users. The first 50 data blocks of the first user are used for training and rest 100 data blocks of the first user are considered as normal and 150 blocks of the second user as abnormal. User 5 and user 32 are selected in the experiments.

We use $\alpha = 99.9\%$ and squared Euclidean distance as anomaly index for anomaly detection in the experiments and the testing results are shown as Fig. 5. Table 7 shows the CPU times required for the training and detection for Fully- Connected HMMs (FC-HMM), Left-to-Right HMMs (LR-HMM), and CE (Cross Entropy) method reported in [14] and ECM method reported in [9] and our PCA method. FC-HMM and LR-HMM methods are based on the transition property of audit events and ECM method is based on the correlation information of audit events while CE method and our PCA method are based on the frequency property of audit events. Our method is tested on a computer with 2.4 GHz Pentium



Fig. 5. Testing results of the combined data of user 5 and user 32. The *y*-axis represents the *anomaly index* and *x*-axis represents command block number. All the data blocks of user 5 and 32 are uncontaminated, therefore the first 100 data blocks from user 5 are treated as normal (\bullet) and blocks 101–250 from user 32 are considered as abnormal (*) with gray shading.

Training and detection times with shell command data							
Method	Number of shell commands for training	CPU time for training (s)	Number of shell commands for detection	CPU time for detection (s)			
FC-HMM	10,826	32,696	10,981	20			
LR-HMM	10,826	33,532	10,981	12			
CE	10,826	0	10,981	14			
ECM	250,000	50,444	100	22.15			
PCA	10,000	6	11,000	5			

Table 7 Training and detection times with shell command data

CPU and 512 DDR MB memory and ECM method was tested on a workstation with 3.2 GHz CUP and 4 GB memory [9] while the other two methods were tested on an UltraSPARC 30 workstation [14].

From Fig. 6, it is seen that the abnormal data can be 100% distinguished from the normal data without any false alarms by using our model. It is also observed from Table 7 that our method is much faster than the FC-HMM, LR-HMM and ECM methods in training, while the CE method does not require training time. The detection time of our method is also faster than other four methods. This shows that considering correlation information of audit events is as much computational costly as using those of the transition property. Taking account of the frequency property of audit events, on the other hand, require low overload not only for training but also for detection. It is thus suitable for processing of massive data streams for real-time intrusion detection.

4.3. Experiments on network data

4.3.1. Data sets

The network data used for testing is distributed by MIT Lincoln Lab for 1998 DARPA evaluation [42]. The data contains traffic in a simulated military network that



Fig. 6. ROC curves for different ratios α used in the network intrusion detection experiments.

consists of hundreds of hosts. The data includes 7 weeks of training set and 2 weeks of test set that were not from the same probability distribution as the training set. Since the probability distribution is not the same, in our experiments, we only use the training set and sample one part of the data for training and another different part of the data for testing. The raw training set of the data contains about 4 GB of compressed binary tcpdump data of network traffic and it was pre-processed into about 5 million connection records by Lee et al. [33,34] as part of the UCI KDD archive [55]. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows from a source IP address to a target IP address under some well defined protocol [55]. In the 10% subset data, each network connection is labeled as either normal, or as an exactly one specific kind of attack.

There are 22 types of attacks in total in the subset. These attacks fall in one of the following four categories:

- DOS: denial-of-service (e.g., teardrop).
- R2L: unauthorized access from a remote machine (e.g., password guessing).
- U2R: unauthorized access to local superuser (root) privileges by a local unprivileged user (e.g., buffer overflow attacks).
- PROBE: surveillance and other probing (e.g., port scanning).

A connection of the network data contains 41 features. These features were extracted by Lee et al. from the raw data divided into three groups: basic features of individual TCP connections, traffic features and content features within a connection suggested by domain knowledge [33,34]. Among these 41 features, 34 are numeric and 7 are symbolic. Only the 34 numeric features were used in the experiments. Each connection in the data set is thus transformed into a 34-dimensional vector as data input for detection. There are 494,021 connection records in the training set in which 97,277 are normal and 396,744 are attacks. In the normal data, we randomly selected 7000 connections for training the normal model and 10,000 for detection. All the attack data are used for detection. The data descriptions in the experiments are shown in Table 8.

 Table 8

 Descriptions of network data in the experiments

Data category	Total number of network connections	Number of network connections used		
Normal	97,277	7000 for training 1000 for testing		
Attack				
DOS	391,458	391,458		
R2L	1126	1126		
U2R	52	52		
PROBE	4107	4107		

4.3.2. Testing results and analysis

In the experiments, we use Receiver Operating Characteristic (ROC) curves to evaluate the network intrusion detection performance of our model. The ROC curve is the plot of DR against FAR. There is a tradeoff between the DR and FPR and the ROC curve is obtained by setting different threshold on the *anomaly index* defined by (8). To further investigate the impact of the ratio α defined in (6) on the performance of the intrusion detection model, we also use different number of principal components in the experiments and the testing results of overall data are shown in Fig. 6. In the experiments, only one principal component can account for 90.079% of the total variation and two principal components can account for that of 99.953%. From the ROC curve shown in Fig. 6, it is observed that the testing results are related to the ratio α . The testing results are the best when the ratio is about 99.9% and this consists with our previous results on system call data discussed in Section 4.1.

Testing results on all the network data are shown in Fig. 7. To investigate the performance of our model on different categories of attack data, we conduct the experiments on each category of the attack data. Fig. 8 shows the ROC curves of the detection performance of our model on four categories of attack data as well as overall data.

From Fig. 7 and the ROC curves shown in Fig. 8, it is observed that our model is able to detect most attacks with



Fig. 7. Testing results of all the network data. The *y*-axis represents the *anomaly index* and *x*-axis represents the network connection number. The stars (*) in the gray shading stands for attack connections and the dots (\bullet) stand for normal connections. The *y*-axis is compressed for readability.



Fig. 8. ROC curves for four categories of attack data and overall data.

low false alarm rates. In more details, our model can detect very high percentage of DOS and U2R attacks with a small number of R2L and PROBE attacks missed.

To evaluate the performance of our method and compare with different methods, we summarize the DR and FAR of the four attack categories as well as the overall in Table 9 in comparison with other five methods reported in [35–37]. In the experiments, we also measure the CPU times for training and detection on a computer with 2.4 GHz Pentium CPU and 512 DDR MB memory shown in Table 10.

In the experiments, we used randomly selected normal data of the training set for establishing normal behavior and used nearly all the other data in the same set for detection. The GC method [36] also used the training set both for training and detection but only with a very small part of the data. In the Cluster, K-NN and SVM methods [37], many attack data are filtered so that the resulting data set consisted of 1% to 1.5% attack and 98.5-99% normal instances for unsupervised anomaly detection. The PCC method [35] used the same data set as ours both for training and detection but also with a smaller data size. From Table 9, it can be seen that our model is better than the other first four methods in terms of detection rates and false alarm rates. The DM method [33,34], HKN method [39] and SOM method [38] used both the normal data and attack data of the training set for defining attack signatures or for building detection models and used the test set for detection. The DM method achieved an 80.2% detection rate and HKN method achieved 93.46% detection rate at 3.99% false alarm rate while SOM method obtained 89% detection rate at 4.6% false alarm rate.

The detection performance of the PCC method is almost the same as our PCA model. The PCC method measured the Mahalanobis distance of each observation from the center of the data for anomaly detection. Any observation that has distance larger than a threshold is considered as an anomaly. The Mahalanobis distance is then computed based on the sum of squares of the standardized principal component scores. The PCC method used both principal

Table 9	
The Detection Rates (DR) and False Alarm Rates (FAR) in comparison with other	methods

Methods	Overall		DoS		R2L		U2R		Probe	
	DR(%)	FAR(%)	DR(%)	FAR(%)	DR(%)	FAR(%)	DR(%)	FAR(%)	DR(%)	FAR(%)
GC (Liu) [35]	59.4	0.4	56	_	66	_	78	_	44	_
Cluster (Eskin) [36]	93	10	_	_	_	_	_	_	_	_
K-NN (Eskin) [36]	91	8	_	_	_	_	_	_	_	_
SVM (Eskin) [36]	98	10	_	_	_	_	_	_	_	_
PCC (Shyu) [37]	97.89	0.92	_	_	_	_	_	_	_	_
PCA	98.8	0.4	99.2	0.2	94.5	4	88.5	0.6	80.7	4

Table 10

Training and detection times with network data

Data category	Training Testing (Note that each category of attack data includes 10,000 normal network connection for anomaly detection)						
	Normal data	DOS attack	R2L attack	U2R attack	PROBE attack		
Number of network connections	7000	401,458	11,126	10,052	14,107		
CPU time (s)	36	658	0.33	0.25	0.45		

components and minor components of the sample in the detection stage. Our PCA method directly reduces the high dimensional data into low dimensional space and use the distance between each observation and its reconstruction in the reduced subspace for anomaly detection. Only principal components are required to form the subspace and the detection scheme is straightforward and easy to handle. The PCC method used five principal components and 6–7 minor components in the experiments while our PCA method only used two principal components and achieved better detection results. The PCC method assumes that the sum of squares of several standardized principal component follows a χ^2 distribution. Our model avoids any data distributional assumption and can be more practical for application.

In the experiments, our model is evaluated on 10,000 normal network connections and all the attack connections. Thus over 400,000 network connections are included. It can be observed from Table 10 that training and detection are very efficient. For example, less than 1 second is required for detecting about 15,000 network connections. This shows that our model is suitable for real-time anomaly detection on network data.

5. Concluding remarks

In this paper, we present an effective anomaly intrusion detection model based on Principal Component Analysis (PCA). The model is more suitable for high speed processing of massive data streams in real-time than use of transition property or correlation property. In our model, the data block for a process, command or network connection is associated with a data vector representing the frequencies or other extracted features of individual elements in the data block. Large amounts of data are thus significantly reduced. The anomaly intrusion detection problem is converted into classifying these vectors as normal or abnormal. The detection model provides a general framework for establishing a practical IDS in various environments. It can process many sources of audit data such as system call, UNIX command, network data, etc., of large size, applicable to a broad range of anomaly intrusion detection.

Data used in intrusion detection problems are high dimensional in nature. Our model applies PCA to reduce the high dimensionality of the data. The *anomaly index* of a data block is represented as a single number as the Euclidean distance between the data vector and its reconstruction in the reduced subspace so that normal behavior is easily profiled and anomaly detection is easily implemented without any additional classifier. It is thus an effective model to process a mass of audit data in real-time with low overhead and is suitable for real-time intrusion detection.

It is possible for a hacker to escape detection by not letting the process terminate. However, the model can still be made effective for real-time anomaly detection. An attack usually produces one or more programs and each program produces one or more processes. If one process is detected as anomalous, the program containing the process is then classified as anomalous and an intrusion alarm is thus reported. Besides, to avoid this situation, one can specify a maximum length of system calls in each process, for example, length of 1500 [25], and then only use the limited length of the system call sequence for detection without reaching the end of the process.

There are also disadvantages for the models based on the frequency property of the system behavior such as PCA. If frequencies of system calls or commands generated by a hostile program or an unauthentic user are very similar to those produced by normal programs or authentic users although the sequences are quite different, PCA can hardly detect the anomalies. The other methods based on the transition or correlation analysis may detect such anomaly without difficulty.

Four data sets, the system call data from UNM and our KLINNS lab, the shell command from AT&T Research lab and network data from MIT Lincoln Lab, are used to validate the model. Extensive experiments are conducted to test our model and to compare with the results of many other methods. Testing results show that the model is promising in terms of detection accuracy, computational efficiency and implementation for real-time intrusion detection. For further work, we are investigating approaches to combining the frequencies properties with the transition properties and correlation information of the system and network behavior in order to achieve lower false alarm rates and higher detection rates.

Acknowledgements

We thank Dr. Weixiang Liu, Graduate School at Shenzhen, Tsinghua University, for the fruitful suggestions and comments. We thank Ms. Mizuki Oka, Department of Computer Science of University of Tsukuba, Japan, for the valuable discussions. The research presented in this paper was supported in part by the NSFC (60736027, 60574087), 863 High Tech Development Plan (2007AA01Z475, 2007AA04Z154, 2007AA01Z480, 2007AA01Z464) and 111 International Collaboration Program, of China.

References

- D.E. Denning, "An intrusion-detection model", IEEE Transactions on Software Engineering 13 (2) (1987) 222–232.
- [2] S.E. Smaha, Haystack: An intrusion detection system, in: Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference, 1988.
- [3] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, T. Garvey, A real-time intrusion detection expert system (IDES) final technical report, Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.
- [4] D. Anderson, T. Frivold, A. Valdes, Next-generation intrusion detection expert system (NIDES): a summary. Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, California, May 1995.
- [5] M. Schonlau, M. Theus, Detecting masquerades in intrusion detection based on unpopular commands, Information Processing Letters 76 (2000) 33–38.
- [6] M. Schonlau, W. Dumouchel, W.-H. Ju, A.F. Karr, M. Theus, Y. Vardi, Computer intrusion: detecting masquerades, Statistical Science 16 (1) (2001) 58–74.
- [7] R.A. Maxion, T.N. Townsend, Masquerade detection using truncated command lines, Proceedings of the International Conference on Dependable Systems and Networks (DSN'02), IEEE Computer Society Press, Washington, D.C., Los Alamitos, California, 2002, pp. 219–228.
- [8] T. Lane, C.E. Brodley, Temporal sequence learning and data reduction for anomaly detection, in: Proceedings of Fifth ACM Conference on Computer and Communication Security, 1998.
- [9] M.Oka, Y. Oyama, H. Abe, K. Kato, Anomaly detection using layered networks based on eigen co-occurrence matrix, in: Proceed-

ings of Seventh International Symposium on Recent Advances in Intrusion Detection (RAID'2004), Springer, LNCS-3224, 2004, pp. 223–237.

- [10] S. Forrest, S.A. Hofmeyr, A. Somayaji, T.A. Longstaff, A sense of self for Unix processes, in: Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, Los Alamos, CA, 1996, pp. 120– 128.
- [11] W. Lee, S. Stolfo, Data mining approaches for intrusion detection, in: Proceedings of the Seventh USENIX Security Symposium, Usenix Association, 1998, pp.79–94.
- [12] C. Warrender, S. Forrest, B. Pearlmutter, Detecting intrusions using system calls: alternative data models, in: Proceedings of 1999 IEEE Symposium on Security and Privacy, 1999, pp.133–145.
- [13] Q. Yan, W. Xie, B. Yan, G. Song, An anomaly intrusion detection method based on HMM, Electronics Letters 38 (13) (2002) 663–664.
- [14] D.Y. Yeung, Y. Ding, Host-based intrusion detection using dynamic and static behavioral models, Pattern Recognition 36 (1) (2003) 229– 243.
- [15] S.B. Cho, H.J. Park, Efficient anomaly detection by modeling privilege flows using hidden Markov model, Computers and Security 22 (1) (2003) 5–55.
- [16] W. Wang, X. Guan, X. Zhang, Modeling program behaviors by hidden markov models for intrusion detection, in: Proceedings of the Third International Conference on Machine Learning and Cybernetics (ICMLC'2004), 2004, pp. 2830–2835.
- [17] A. Wespi, M. Dacier, H. Debar, Intrusion detection using variablelength audit trail patterns, in: Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000), LNCS-1907, 2000.
- [18] M. Asaka, T. Onabuta, T. Inoue, S. Okazawa, S. Goto, A new intrusion detection method based on discriminant analysis, IEICE Transactions on Information and Systems E84D (5) (2001) 570–577.
- [19] W. Lee, D. Xiang, Information-theoretic measures for anomaly detection, in: Proceedings of the 2001 IEEE Symposium on Security and Privacy, Oakland, CA, May 2001.
- [20] Y.H. Liao, V.R. Vemuri, Use of k-nearest neighbor classifier for intrusion detection, Computers and Security 21 (5) (2002) 439–448.
- [21] W. Hu, Y. Liao, V.R. Vemuri, Robust support vector machines for anomaly detection in computer security, in: Proceeding of the 2003 International Conference on Machine Learning and Applications (ICMLA'03), Los Angeles, California, 2003.
- [22] S.B. Cho, Incorporating soft computing techniques into a probabilistic intrusion detection system, IEEE Transactions on Systems, Man, and Cybernetics – Part C 32 (2) (2002) 154–160.
- [23] Z. Cai, X. Guan, P. Shao, Q. Peng, G. Sun, A rough set theory based method for anomaly intrusion detection in computer networks, Expert Systems 18 (5) (2003) 251–259.
- [24] L. Feng, X. Guan, S. Guo, Y. Gao, P. Liu, Predicting the intrusion intentions by observing system call sequences, Computers and Security 23 (5) (2004) 241–252.
- [25] W. Wang, X. Guan, X. Zhang, Profiling program and user behaviors for anomaly intrusion detection based on non-negative matrix factorization, in: Proceedings of 43rd IEEE Conference on Control and Decision (CDC'2004), Atlantis, Paradise Island, Bahamas, 2004, pp. 99–104.
- [26] N. Ye, Y. Zhang, C.M. Borror, Robustness of the Markov chain model for cyber attack detection, IEEE Transactions on Reliability 53 (1) (2004) 116–121.
- [27] W.-H. Ju, Y. Vardi, A hybrid high-order Markov chain model for computer intrusion detection, Journal of Computational and Graphical Statistics 10 (2) (2001) 277–295.
- [28] N. Ye, Q. Chen, Computer intrusion detection through EWMA for auto-correlated and uncorrelated data, IEEE Transactions on Reliability 52 (1) (2003) 73–82.
- [29] N. Ye, X. Li, Q. Chen, S.M. Emran, M. Xu, Probabilistic techniques for intrusion detection based on computer audit data, IEEE Transactions on Systems, Man, and Cybernetics – Part A 31 (4) (2001) 266– 274.

- [30] N. Ye, Q. Chen, An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems, Quality and Reliability Engineering International 17 (2) (2001) 105–112.
- [31] A.K. Ghosh, A. Schwartzbard, M. Schatz, Learning program behavior profiles for intrusion detection, in: Proceedings of the First USENIX Workshop on Intrusion Detection and Network Monitoring, 1999, pp. 51–62.
- [32] P.A. Porras, P.G. Neumann, EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances, in: Proceedings of National Information Systems Security Conference, Baltimore, MD, 1997.
- [33] W. Lee, S. Stolfo, K. Mok, A data mining framework for adaptive intrusion detection, in: Proceedings of the 1999 IEEE Symposium on Security and Privacy, Los Alamos, CA, 1999, pp. 120–132.
- [34] W. Lee, S. Stolfo, A Framework for constructing features and models for intrusion detection systems, ACM Transactions on Information and System Security 3 (4) (2000) 227–261.
- [35] Y. Liu, K. Chen, X. Liao, et al., "A genetic clustering method for intrusion detection", Pattern Recognition 37 (5) (2004) 927–942.
- [36] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A Geometric framework for unsupervised anomaly detection, Applications of Data Mining in Computer Security, Kluwer Academics, Dordrecht, 2002.
- [37] M. Shyu, S. Chen, K. Sarinnapakorn, L. Chang, A novel anomaly detection scheme based on principal component classifier, in: Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM'2003), 2003, pp. 172–179.
- [38] H. Kayacik, A. Zincir-Heywood, M. Heywood, On the capability of an SOM based intrusion detection system, in: Proceedings of the IEEE International Joint Conference Neural Networks (IJCNN'2003), 2003, pp. 1808–1813.
- [39] S.T. Sarasamma, Q.A. Zhu, J. Huff, Hierarchical Kohonenen net for anomaly detection in network security, IEEE Transactions on Systems, Man and Cybernetics, Part B 35 (2) (2005) 302–312.
- [40] S.C. Lee, D.V. Heinbuch, Training a neural-network based intrusion detector, IEEE Transactions on Systems man and Cybernetics 31 (4) (2001) 294–299.
- [41] G. Giacinto, F. Roli, L. Didaci, Fusion of multiple classifiers for intrusion detection in computer networks, Pattern Recognition Letters 24 (5) (2003) 1795–1803.
- [42] MIT Lincoln Laboratory-DARPA Intrusion Detection Evaluation Documentation, http://www.ll.mit.edu/IST/ideval/docs/docs_index.html, 1999.
- [43] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989).
- [44] L.R. Rabiner, B.H. Juang, An introduction to hidden Markov models, IEEE ASSP Magazine (1986).
- [45] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., China Machine Press, Beijing, 2004, Feb.
- [46] C. Kruegel, D. Mutz, F. Valeur and G. Vigna, On the detection of anomalous system call arguments, in: Eighth European Symposium on Research in Computer Security (ESORICS'2003), LNCS, Norway, 2003, pp. 101–118.
- [47] D. Mutz, F. Valeur, C. Kruegel, G. Vigna, Anomalous system call detection, ACM Transactions on Information and System Security 9 (1) (2006) 61–93.
- [48] CERT Advisory CA-2001-07 File, Globbing Vulnerabilities in Various FTP Servers, http://www.cert.org/advisories/CA-2001-07.html>, 2001.
- [49] I.T. Jolliffe, Principal Component Analysis, second ed., Springer-Verlag, NY, 2002.
- [50] M. Turk, A. Pentland, Eigenfaces for recognition, Journal of Neuroscience 3 (1) (1991) 71–86.
- [51] G.H. Golub, C.F. van Loan, Matrix Computation, Johns Hopkins University Press, Baltimore, 1996.
- [52] W. Wang, X. Guan, X. Zhang, A novel intrusion detection method based on principal component analysis in computer security, in:

Advances in Neural Networks-ISNN2004. International IEEE Symposium on Neural Networks, Dalian, China. LNCS-3174, August 2004, pp. 657–662.

- [53] W. Wang, R. Battiti, Identifying intrusions in computer networks with principal component analysis, in: Proceedings of the First International Conference on Availability, Reliability and Security (ARES 2006), IEEE Press Society, Vienna, Austria, April 2006, pp. 270–277.
- [54] K.M.C. Tan, R.A. Maxion, Why 6? Defining the operational limits of stide, an anomaly-based intrusion detector, in: Proceedings of 2002 IEEE Symposium on Security and Privacy, 2002, pp. 188–201.
- [55] KDD Cup 1999 Data, http://www.kdd.ics.uci.edu/databases/kdd-cup99/kddcup99.html, 1999.



Wei Wang (wei.wang.email@gmail.com) received his B.S. degree in process equipment and control engineering and M.S. degree in mechanical and electronic engineering from Xi'an Shiyou University, Xi'an, China, in 1997 and 2000, respectively, and his Ph.D. Degree in Control Science and Engineering from Xi'an Jiaotong University, Xi'an, China, in 2005. He was a research fellow from July 2005 to February 2006 and a postdoctoral research fellow from February 2006 to July 2006 in Department of Information and Commu-

nication, University of Trento, Italy. He is a postdoctoral research fellow in RSM department at GET-ENST Bretagne - Campus Rennes, France in 2007 and will join INRIA (French National Institute Research in Computer Science and Control), France, in 2008. His research interests currently focus on computer networked systems and computer network security.



Xiaohong Guan (xhguan@tsinghua.edu.cn) received his B.S. and M.S. degrees in Control Engineering from Tsinghua University, Beijing, China, in 1982 and 1985, respectively, and his Ph.D. Degree in Electrical Engineering from the University of Connecticut in 1993. He was a senior consulting engineer with PG&E from 1993 to 1995. From 1985 to 1988 and since 1995 he has been with the Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, China, and currently he is the Cheung Kong Professor of Systems Engineering

and Director of the National Lab for Manufacturing Systems. He is also the Chair of Department of Automation and Director of the Center for Intelligent and Networked Systems, Tsinghua University, China. He visited the Division of Engineering and Applied Science, Harvard University from January 1999 to February 2000. He is an IEEE fellow. His research interests include computer network security, wireless sensor networks and economics and security of complex networked systems.



Xiangliang Zhang (xlzhang@lri.fr) received her B.S. Degree in Information and Communication Engineering and M.S. Degree in Electronic Engineering from Xi'an Jiaotong University, Xi'an, China, in 2003 and 2006, respectively. She was an internship student in Department of Information and Communication, University of Trento, Italy, from February 2006 to May 2006. She is currently a Ph.D. student in Laboratoire de Recherche en Informatique, mixed with French National Institute for Research in Computer

Science and Control (INRIA), National Center for Scientific Research (CNRS) and University of Paris-sud 11, France. Her research interests include network security, machine learning, data mining and their applications, e.g., computer security, complex system modeling and grid management.