Journal of Network and Computer Applications 32 (2009) 31–44

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Fast intrusion detection based on a non-negative matrix factorization model ☆

Xiaohong Guan [a,b], Wei Wang [a,*], Xiangliang Zhang [a]

[a] *MOE Key Lab for Intelligent Networks and Network Security (KLINNS) and State Key Lab for Manufacturing Systems (SKLMS), Xi'an Jiaotong University, Xi'an 710049, China*
[b] *Center for Intelligent and Networked Systems, TNLIST Lab, Tsinghua University, Beijing 100084, China*

ARTICLE INFO

Article history:
Received 7 November 2007
Received in revised form
2 April 2008
Accepted 23 April 2008

Keywords:
Computer security
Intrusion detection system
Anomaly detection
Non-negative matrix factorization

ABSTRACT

In this paper, we present an efficient fast anomaly intrusion detection model incorporating a large amount of data from various data sources. A novel method based on non-negative matrix factorization (NMF) is presented to profile program and user behaviors of a computer system. A large amount of high-dimensional data is collected in our experiments and divided into smaller data blocks by a specific scheme. The system call data is divided into blocks by processes, while command data is divided into consecutive blocks with a fixed length. The frequencies of individual elements in each block of data are computed and placed column by column as data vectors to construct a matrix representation. NMF is employed to reduce the high-dimensional data vectors and anomaly detection can be realized as a very simple classifier in low dimensions. Experimental results show that the model presented in this paper is promising in terms of detection accuracy, computation efficiency and implementation for fast intrusion detection.

1. Introduction

Computer network security is gaining worldwide attention as attacks on computer network systems have become more and more widespread in recent years. In many cases, sophisticated

☆ 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.

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

1084-8045/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2008.04.006

hackers successfully penetrated many conventional peripheral protection mechanisms such as firewalls and various authentications, and caused enormous security and economic damages (Zou et al., 2002; Moore and Shannon C, 2002; Kruegel et al., 2005; Moore et al.). Moreover, distributed and coordinated attacks launched mostly from mal-codes and spam emails may cease normal functions of a large portion of Internet in less than 15 min (Moore et al.). Therefore, fast-automated and integrated detection mechanisms and control strategies are required, and intrusion detection systems (IDS) together with intelligent control of network infrastructure consisting of routers, switches, firewalls, hosts, etc., constitute the defense-in-depth or layered framework for securing a computer network system. The concepts of system modeling and control can play an important role in defending attacks and preventing intrusions. The framework of the integrated network defense system is shown in Fig. 1, where the information gathered from the IDSs with different data resources (host and network) and the flow analyzer for monitoring bursts of large-scale distributed denial of service attacks caused by worms, etc., is fed into control centers. The decisions are made and control actions are taken at the control centers with coordination and then sent to firewall, switches, routers, etc., to prevent intrusions and attacks in real time. Fast intrusion detection, therefore, is very important so that appropriate response action can be taken as soon as possible for intrusion prevention.

Intrusion detection is a technology for detecting hostile attacks against computer systems from both outside and inside. In terms of detection mechanism, the techniques for intrusion detection can be classified into two categories: signature-based detection and anomaly detection. Signature-based detection looks for evidences of malicious behaviors matched against pre-defined descriptions of attacks or signatures. Although signature-based detection is effective for detecting known attacks, it generally cannot detect the new attacks that are not predefined. Anomaly detection, on the other hand, builds the profile of normal behaviors and attempts to identify the patterns or activities that deviate from the normal profile. Based on the concept of profiling normal behaviors, a salient feature of anomaly detection is that it can detect unknown attacks. However, it may also cause significant number of false alarms since the model assumed to describe complete normal behaviors may not be accurate, and obtaining such a model usually by machine learning is difficult. The research focus on anomaly detection is to find more effective and accurate methods.

Anomaly detection is an active research area and has been widely studied for more than a decade since it was originally proposed by Denning (1987). There are multiple levels that anomaly detection can be built upon in an actual computer network system. A great challenge is to select features that best characterize patterns of a subject (e.g., a program, a user, etc.), so that abnormal activities can be

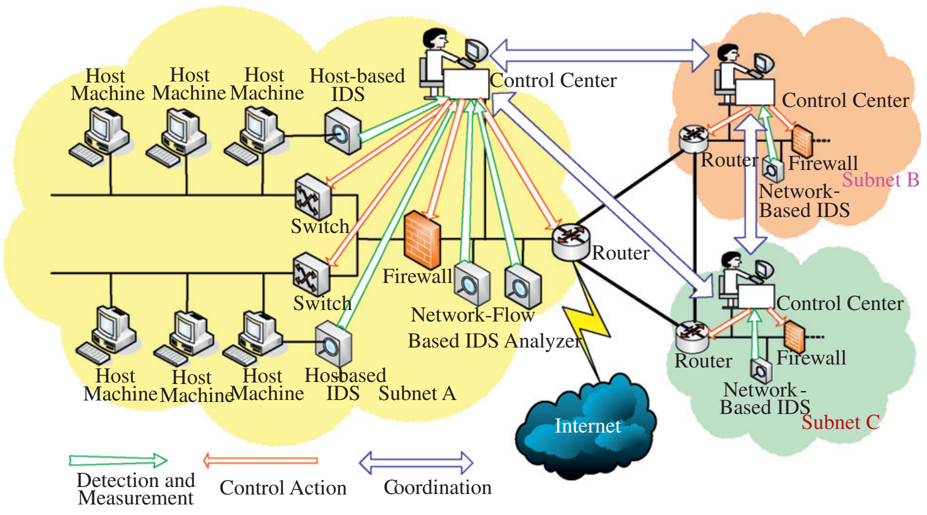

Fig. 1. The framework of the integrated network defense system.

clearly distinguished from normal ones. Many types of data sources can be used including command sequences, audited events, keystroke records, system calls, and network packets. Earlier studies (Smaha, 1988; Lunt et al., 1992; Anderson et al., 1995) on anomaly detection mainly focused on modeling systems or user behaviors from monitored system logs or accounting log data. Examples of these data logs include: CPU usage, login times, durations of user sessions, names of files accessed, etc. More recently, Schonlau and Theus (2000) attempted to detect masquerades by profiling user behaviors using truncated command sequences. Experiments with six masquerade detection techniques including Bayes one-step Markov, hybrid multi-step Markov, IPAM, uniqueness, sequence-match and compression, are performed and compared in Schonlau et al. (2001). Lane and Brodley (1998) propose a learning algorithm for analyzing user command history to profile normal user behaviors and detect anomalies. The problem of "concept drift" is addressed when the normal user behavior changes.

In recent years, a lot of research activities in anomaly detection focused on learning program behaviors and building the profiles with system call sequences as data sources. In 1996, Forrest et al. (1996) introduced a simple anomaly detection method named time-delay embedding (TIDE). The method is based on monitoring system calls issued by active and privileged programs. The profile of normal behaviors is built by enumerating all distinct and contiguous system calls with fixed length that occur in the training data, and unmatched sequences in actual detection are considered abnormal. This approach is extended by various other methods. Lee and Stolfo (1998) used the data mining approach to study a sample of system call data to characterize the sequences by a small set of rules and the sequences violating those rules are treated as anomalies. Warrender et al. (1999) proposed the hidden Markov model (HMM) method for modeling and evaluating invisible events based on system calls. This method was further studied by many researchers (Yeung and Ding, 2003; Cho and Park, 2003; Wang et al., 2004a–c). Wespi et al. (2000) further developed Forrest's idea and proposed a variable length approach. The Rough set theory method is applied to learn a rule set modeling the normal behaviors with small size of training data sets and improved detection accuracy Cai et al. (2003). It is extended by another method based on plan recognition for predicting intrusion intentions (Feng et al., 2004). Liao and Vemuri (2002) used kNN classifier and Hu et al. (2003) applied robust support vector machines for intrusion detection based on system call sequences. In our research group, we also employed principal component analysis (PCA) (Wang et al., 2004a–c, 2008) and self-organizing maps (SOM) Wang et al. (2006) for system call-based anomaly detection.

Although the efforts on anomaly detection have resulted in impressive progress, there are still some open issues to be resolved. An actual computer system may produce a large amount of audit data from various data sources with high dimensionality. In the experiment for collecting live *ftp* system calls on a Linux server in our KLINNS lab, for example, more than 1 million system calls are generated in less than half an hour. There are more than 200 distinct system calls that can be invoked in running a process in a Linux operation system with kernel 2.4.7-10, and a lot of processes would be generated in a short time on an active host. Therefore, fast processing of a large amount of high-dimensional data is crucial to build a real-time intrusion detection model so that an intrusion can be detected before substantial damage to the computer system is done. However, many intrusion detection models make implicit assumptions of processing relatively small amount or low dimension of data to train the models (Feng et al., 2004; Liao and Vemuri, 2002; Hu et al., 2003), since it may be very time consuming to process a large amount of high-dimensional data. For example, it takes approximately 2 months to train an HMM-based intrusion detection model (Warrender et al., 1999). An efficient method able to process a massive amount of data is required for building an intrusion detection model.

In this paper, we present an efficient fast anomaly intrusion detection model incorporating a large amount of data from various sources. A novel method based on non-negative matrix factorization (NMF) is presented to profile program and user behaviors of a computer system. NMF has been successfully applied to image representation such as human face recognition, document classification, and color categorization (Wang et al., 2004a–c). It is suitable for analyzing the frequency characteristics of individual system call or command in a block of audit data. Profiling program behaviors based on NMF reduces the time-consuming training process. A large amount of high-dimensional data is collected in our experiments and divided into smaller data blocks by a specific scheme. The system call data are divided into small blocks by processes, while command data are divided into consecutive blocks with a fixed length. The frequencies of individual elements in each block of data are computed and placed column by column as data vectors to construct a matrix representation. NMF is employed to reduce the high-dimensional data vectors and anomaly detection can be realized as a very simple classifier in low dimensions. Moreover, computational resources could be largely saved since the data with reduced dimension are stored and used. Based on NMF, the data matrix can be factorized into two simple matrices with positive elements, one as the base and the other as encoding coefficients representing normal or abnormal features. Program and user behaviors are then profiled by the features extracted by NMF. If the features contained in any block of data collected in real-time deviates significantly from those of the normal behaviors learned in training data sets, the block of data is considered associated with anomalous behaviors.

Two sources of data, *sendmail* and live *lpr* system call data from the University of New Mexico (UNM), live *ftp* system call data from our lab at MOE Key Lab for Intelligent Networks and Network Security (KLINNS) of Xi'an Jiaotong University, and command data from research lab, are used to validate and evaluate our model. Experimental results show that the model presented in the paper is promising in terms of detection accuracy, computational efficiency, and implementation for fast detection.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction of NMF and describes the new intrusion detection model. In Section 3, experiments are conducted with several system call sequences and command data sets to illustrate the effectiveness of the model and the empirical results are discussed and analyzed. Conclusion remarks are given next.

2. Intrusion detection model based on non-negative matrix factorization

2.1. Non-negative matrix factorization

NMF is a powerful method for reduced representation of data and has been successfully applied to face recognition, document classification, color categories, etc. (Lee and Seung, 1999). The salient feature of this method is its use of non-negativity constraints, leading to the par- based representation and allowing only additive combination. This is specifically suitable for analyzing the frequency characteristics of individual system calls or commands since all the frequencies are positive.

Given an initial data set expressed by an $n \times m$ matrix X, where each column is an n-dimensional non-negative vector of the original data (m vectors). Based on the theory of NMF, it is possible to find two new matrices, W and H, to approximate the original matrix $X \approx WH$ (Lee and Seung, 2000), or

$$X_{i\mu} \approx (WH)_{i\mu} = \sum_{a=1}^{r} W_{ia} H_{a\mu} \tag{1}$$

where $X_{i\mu}$, W_{ia} and $H_{a\mu}$ are the elements of matrices X, W, and H, respectively.

The dimensionalities of the factorized matrices W and H are $n \times r$ and $r \times m$, respectively. Usually, r is chosen as a number smaller than n and m so that W and H are smaller than the original matrix X. Eq. (1) can be rewritten by column vectors as $\mathbf{x} \approx W\mathbf{h}$, where $\mathbf{x}$ and $\mathbf{h}$ are the corresponding column vectors of W and H. In other words, each data vector $\mathbf{x}$ is approximated by a linear combination of the columns of W, weighed by the components of $\mathbf{h}$. Therefore, W is regarded as containing a basis optimized for the linear approximation of X (Lee and Seung, 1999). Each column of matrix W contains basis vectors and each column of H contains encoding coefficients. The data matrix X are considered as original observation vectors, and the encoding matrix H as the feature vectors learned based on the basis W.

In order to find a factorization $X \approx WH$, one of the iterative approaches is given by the following rules (Lee and Seung, 2000):

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^{\mathrm{T}}X)_{a\mu}}{(W^{\mathrm{T}}WH)_{a\mu}} \tag{2}$$

$$W_{ia} \leftarrow W_{ia}\frac{(XH^{\mathrm{T}})_{ia}}{(WHH^{\mathrm{T}})_{ia}}. \tag{3}$$

Initialization is performed using positive random initial conditions for matrices W and H. The convergence of the process is also ensured.

2.2. The intrusion detection model based on NMF

The fast intrusion detection model developed in this paper is based on the NMF method. Instead of using the transition information of the data in hidden Markov models based on short sequences of continuous system calls (Forrest et al., 1996; Lee and Stolfo, 1998; Warrender et al., 1999; Wespi et al., 2000; Cai et al., 2003; Feng et al., 2004) or Markov chain models based on command sequences (Yeung and Ding, 2003; Cho and Park, 2003; Wang et al., 2004a–c, etc.), the frequency properties of system calls and commands are analyzed to characterize program and user behaviors, respectively. The steps of building the intrusion detection model include: data preparation, reduction and feature extraction by NMF and intrusion detection.

2.2.1. Data preparation

The observation data set is divided into smaller data blocks. For system call data, the processes and the associated system calls are grouped into one trace. The frequency of each individual system call in each trace is calculated. For example, the system call sequence invoked by Process 3939 in *sendmail* data are shown below:

Process ID:3939

105	104	104	106	105	104	104	106	105	104	104	106	5	4	5	5	0
40	41	105	104	104	106	61	5	85	50	27	18	50	27	2	3	2
3	3	2	3	3	2	3	2	2	3	3	2	3	3	2	3	2
3	12	112	19	128	9	9	5	9	9	5	112	4	5	5	5	5

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 "105" represents system call "vtimes", the number "5" represents system call "open". Most current intrusion detection methods (Forrest et al., 1996; Lee and Stolfo, 1998; Warrender et al., 1999; Yeung and Ding, 2003; Cho and Park, 2003; Wang et al., 2004a–c; Wespi et al., 2000; Cai et al., 2003; Feng et al., 2004) divide each sequence (trace) of data into short sequences of system calls with sliding windows and treat the short sequence of system calls as observation. In this paper, we use the total sequence (trace) as observation instead. In each trace of data, the frequencies of individual system calls are calculated. For example, the frequency of number 105 in the process ID of 3939 is 0.056. Each trace of system call data, invoked by each process, is thus transformed into a data vector and the matrix representing a system call data set is shown below:

$$\begin{array}{c c}
\text{Distinct System call} & \begin{array}{ccccc}\text{Trace 1} & \text{Trace 2} & & \text{Trace } m-1 & \text{Trace } m\end{array} \\
\begin{array}{c} 1 \\ 2 \\ \\ 155 \\ 167 \end{array} & \left\{\begin{array}{ccccc}
0.051 & 0.055 & \cdots & 0.049 & 0.051 \\
0.122 & 0.125 & \cdots & 0.118 & 0.115 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0.101 & 0.1 & \cdots & 0.102 & 0.102 \\
0.03 & 0.03 & \cdots & 0.02 & 0.03
\end{array}\right\}
\end{array}$$

Similarly, command data are also divided into smaller blocks consecutively with a fixed length. The frequency of each individual command embedded in a block is calculated. Suppose an observation data set is divided into m blocks and there are a total of n distinct observations (e.g., system calls or commands) in the data set, the observed data set can be expressed by m vectors with each vector containing n observations. Suppose each block of data are represented by a vector of length n denoted as $\mathbf{x}_i$. A $n \times m$ matrix X, where each element X_{ij} stands for the frequency of i-th distinct observation occurs in the j-th block, is then constructed. The observed data set that is represented by a matrix $X_{n \times m}$ can be written as

$$X_{n\times m} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} = \begin{bmatrix} \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x}_m \end{bmatrix} \tag{4}$$

where vectors $\mathbf{x}_1$,$\mathbf{x}_2$,…,$\mathbf{x}_m$ represent the corresponding blocks of the original data.

Clearly based on the frequency property, the following normalization condition is satisfied:

$$\sum_{i=1}^{n} X_{ij} = 1, \quad j \in \{1,2,3,\ldots,m\}. \tag{5}$$

2.2.2. *Dimension reduction and feature extraction*

Given m blocks of training data, the data set can be represented by vectors $\mathbf{x}_1$,$\mathbf{x}_2$,…,$\mathbf{x}_m$ or matrix $X_{n \times m}$. With the iterative updating algorithm given by (2) and (3), the training data set represented by $X_{n \times m}$ is factorized into the $n \times r$ bases W and $r \times m$ coefficients H. Matrix H represents the features learned from the original matrix X and the features associated with each block of data are represented by column vector $\mathbf{h}$ in H.

The iterative updating algorithm in (2) and (3) has the following property (Lee and Seung, 1999):

$$\sum_{i=1}^{r} H_{ij} = \sum_{i=1}^{n} X_{ij}, \quad j \in \{1,2,3,\ldots,m\} \tag{6}$$

Based on Eq. (5), Eq. (6) can also be written as

$$\sum_{i=1}^{r} H_{ij} = 1, \quad j \in \{1,2,3,\ldots,m\} \tag{7}$$

Eq. (7) shows that the sum of the elements in each column of H is equal to number 1. In other words, any block of normal training data is characterized by a single value 1 by adding all the elements in each column vector $\mathbf{h}$ of H. This is the normal behavior profiled from the training data set.

2.2.3. *The scheme of intrusion detection*

Given a data vector $\mathbf{t}$ that represents a block of data obtained in real time, the coefficient vector $\mathbf{h}_t$ can be obtained by the iterative rule (2) based on the basis W learned from the training data set. If the data vector $\mathbf{t}$ corresponds to normal behaviors, its associated coefficient vector should have similar characteristics with those coefficient vectors learned from training data vectors. In other words, if the data vector $\mathbf{t}$ corresponds to normal behaviors, the sum of all the elements in $\mathbf{h}_t$ should be approximately equal to number 1. Let $d = \sum_{i=1}^{r} h_{t_i}$ and a simple classifier is defined as the *anomaly index* for intrusion detection

$$|(d-1)| = \varepsilon. \tag{8}$$

If ε is above a threshold, the block of data associated with the data vector $\mathbf{t}$ is then considered as normal. Otherwise, it is treated as anomalous (Wang et al., 2004a–c).

The training and detection step is illustrated in Fig. 2.

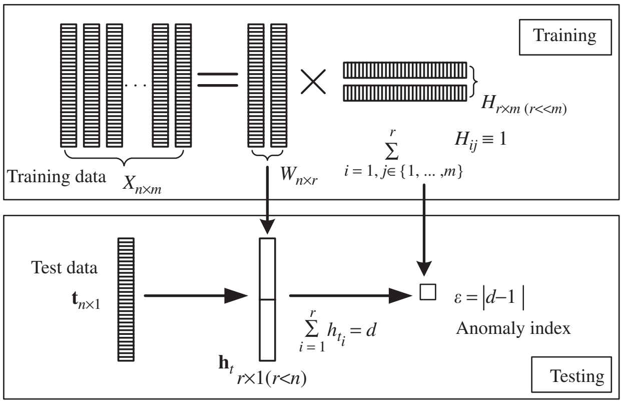
Training
$H_{r\times m\ (r<<m)}$
$\sum_{i=1,j\in\{1,\dots,m\}}^{r} H_{ij}\equiv 1$
Training data $X_{n\times m}$
$W_{n\times r}$
Test data
$\mathbf{t}_{n\times 1}$
$\sum_{i=1}^{r} h_{t_i}=d$
$\varepsilon=|d-1|$
Anomaly index
$\mathbf{h}_t\ r\times 1(r<n)$
Testing

Fig. 2. The training and detection step of the NMF model.

3. Experiments and testing

Four data sets, *sendmail* and live *lpr* system call data from UNM and live *ftp* system call data from our KLINNS lab and command data from AT&T lab, are used to test the intrusion detection model developed in this paper.

3.1. Testing on system call data

3.1.1. Data sets

In order to validate the intrusion detection model on system call data, we used three sets of system call data in the experiments to profile program behaviors for intrusion detection. To facilitate comparison, one data set for testing is *sendmail* system call sequences collected in a UNIX-based host at UNM by Forrest et al. (1996), since they were widely used for testing many other intrusion detection models. In fact one sequence of normal data and 12 sequences of abnormal data including 4 syslog attacks, 3 sunsendmailcp (sscp) attacks and 5 forwarding loop in error conditions are used for testing, including 147 normal traces and 36 abnormal traces. The other data set from UNM is live *lpr* system call data with 2703 traces of normal data and 1001 traces of intrusion data. The first 600 traces of the normal data and the first 300 traces of the intrusion data are used in the experiments. The data sets can be downloaded at http://www.cs.unm.edu/~immsec/, and the procedures of generating the data are also described on the same website.

Another data set used in our experiments is collected in the actual system at our KLINNS laboratory The live *ftp* system call sequences are collected on a Red Hat Linux operation system with kernel 2.4.7-10 including 549 normal traces and 6 intrusion traces. The normal data are the system calls sequences collected during normal usage on an actual system used by actual users. The intrusion data are collected from the exploitations against a widely known Wu-Ftpd vulnerability (CERT), which allows remote attackers to execute arbitrary commands at the victim host.

The description of these three system call data sets used in the experiments are summarized in Table 1.

In the Linux/Unix platform, the execution of a program can generate one or more processes and each process produces a single trace of system calls from the beginning of its execution to the end. Since there are about 221 distinct system calls in the Linux operation system with kernel 2.4.7-10 and below, each trace of the system calls can be converted into a 221 dimensional data column vector. Most elements in the vector are zero since only a limited type of system calls are used in the real environment.

3.1.2. Results and analysis

Good testing results are obtained by employing NMF to profile program behaviors for anomaly detection. Fig. 3 shows the testing results on *sendmail* data with $r = 8$, where 47 traces of normal data are randomly selected for training and another 136 traces for detection. The testing results on live *lpr* data are shown in Fig. 4 by randomly selecting 200 normal traces for training and another 700 traces for detection. It is observed that abnormality indicated by gray shading in the figures can be easily distinguished from normality.

To evaluate the impact of rank r that determines how to factorize the matrix on the performance of the detection model, testing is conducted by varying r with the same data set and the results are summarized in Tables 2 and 3. It is observed that the detection performance of the proposed model is stable when r is kept in a range that is not less than 8. In order to reduce the high-dimensional data at the farthest, it suggests that we use $r = 8$ for dimension reduction and feature extraction for anomaly detection in a practical system call-based intrusion detection system. From Tables 2 and 3, it is observed missing alarm rates and false alarm rates are low with an appropriate r such as $r = 8$. The overall testing results are promising.

The model is also effective with the live *ftp* data from the KLINNS laboratory. Fig. 5 shows the testing results with 35 traces randomly selected from the normal data for training and 520 traces of data for detection. It is observed that there are 6 intrusion traces in the data and the model can detect all of them without false alarm.

It is essential for a practical IDS to process a large amount of audit data in real time. To evaluate the real-time performance of our model, the computational time for training the normal model and real-time detection are studied and compared with other methods including HMM (Wang et al., 2004a–c) and TIDE (Forrest et al., 1996), applied by many research groups. The experiments on HMM method are conducted on a host with P-IV 1.5 GHz processor and 768 MB RAM. The TIDE and our

Table 1
Description of the system call data in the experiments

Data set	Number of system calls	Number of distinct system calls	Number of normal traces (processes)	Number of abnormal traces (processes)
sendmail data from UNM	29,718	56	147	36
live *lpr* data from UNM	842,279	41	600	300
live *ftp* data from the CNSIS Lab	5,699,277	58	549	6

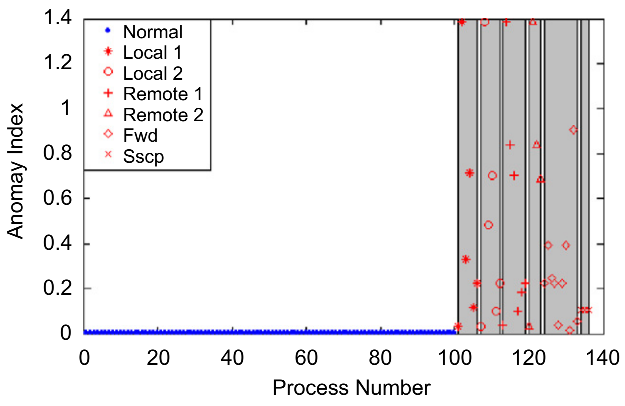

Fig. 3. Testing results on *sendmail* system call data with $r = 8$. The y-axis represents the *anomaly index* and x-axis represents the system call traces (processes). The gray shading indicates abnormality. The y-axis is expanded for readability.

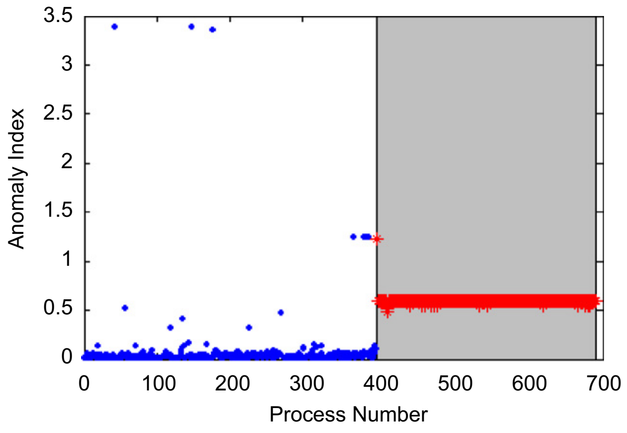

Fig. 4. Testing results on live *lpr* system call data with $r = 8$. The y-axis represents the *anomaly index* and x-axis represents the system call traces (processes). The stars (*) in gray shading stand for abnormal traces and the dots ($\cdot$) with no shading stand for normal traces. The y-axis is expanded for readability.

Table 2
Missing alarm rates and false alarm rates with different r for *sendmail* data

r	No. Missing alarms (36 abnormal traces in total)	No. False alarms (100 normal traces in total)	Missing alarm rate (%)	False alarm rate (%)
20	0	1	0	1
15	0	1	0	1
10	0	1	0	1
8	0	1	0	1
5	0	7	0	7
4	0	30	0	30
3	0	36	0	36
2	0	33	0	33
1	0	97	0	97

Table 3
Missing alarm rates and false alarm rates with different r for live *lpr* data

r	No. Missing alarms (300 abnormal traces in total)	No. False alarms (400 normal traces in total)	Missing alarm rate (%)	False alarm rate (%)
20	0	10	0	2.5
10	0	10	0	2.5
8	0	10	0	2.5
5	0	10	0	2.5
4	0	15	0	3.8
3	0	33	0	8.3
2	0	41	0	10.3
1	0	79	0	19.8

models are tested on a host with Pentium 2.4 GHz processor and 512 MB DDR memory. The testing results are summarized in Table 4.

From Table 4, it is observed that our NMF model has the highest computational efficiency. It takes only 5 s to process hundreds of thousand system calls to train our model. In contrast, training an

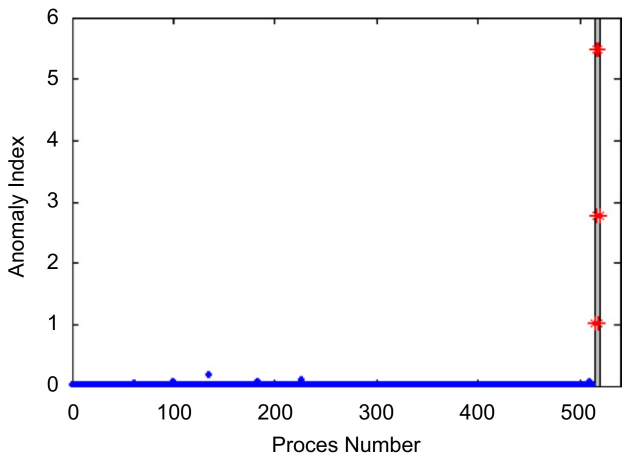

Fig. 5. Testing results on live *ftp* system call data from the CNSIS lab with $r = 8$. The y-axis represents the *detection index* and x-axis represents the system call traces (processes). The stars (*) in the gray shading stand for abnormal trances and the dots ($\cdot$) with no shading stand for normal traces. The y-axis is expanded for readability.

Table 4
Comparison of time for building a normal model and time for intrusion detection on system call data with various methods

Method	Number of system calls for training	CPU time for training (s)	Number of system calls for detection	CPU time for detection (s)
HMM	24,075	About 12000	31,955	59
TIDE	159,642	33	682,637	356
NMF	159,642	5	682,637	13

HMM model is very time consuming. TIDE is regarded as an efficient method for real-time intrusion detection (Warrender et al., 1999). However, it still falls behind our NMF model. It is seen that the NMF model is also very efficient in detection in comparison with other two methods and, therefore, is more suitable for real-time intrusion detection.

3.2. Testing on command data

3.2.1. Data sets

In many cases, the attacks from an insider are more dangerous and intrusion detection based on data sources of a host computer or server should be paid much attention. In this paper, we use command data to profile user behaviors for inside attack detection or masquerade detection. The command data sets collected in AT&T's Shannon Research Laboratory are used in our experiments (Schonlau and Theus, 2000). The command data consist of user names and the associated command sequences (without arguments). 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. Starting at block 51 and onward, some masquerading command blocks, randomly drawn from outside of the 50 users, are inserted into the command sequences of the 50 users. The goal is to correctly detect the masquerading blocks in the user community. The data used in the experiments are available for downloading at http://www.schonlau.net/intrusion.html (see Schonlau and Theus, 2000) for more details of the contamination procedure).

3.2.2. *Results and analysis*

Experiments were performed on command data for different *r*. The testing results show that the performance of detection is very stable, and the rank *r* is not sensitive to the outcomes. This is quite different to the testing results of system call data. The difference between traces of the system call data is small but the difference between blocks of the command data is relatively large. Therefore, it requires more information for representation of system call data and we then choose a relatively large $r = 8$ as mentioned before in a practical system call-based IDS. Meanwhile, we can choose a small $r = 1$ in a practical command-based IDS so that the command data can be largely reduced and anomaly detection is quite easy to be implemented without any additional classifier.

In the experiments on command data, the testing results for most users are promising with some false alarms and missing alarms for some users. The testing results of User 24 and User 9 are shown in Figs. 6 and 7, respectively, as two examples. It is observed that simulated masquerade data are located at blocks 69–89 with gray shading in Fig. 6 and our model can easily catch all of them. Testing result of User 9 is shown as an example of false inference. From Fig. 7, it can be seen that the *anomaly index* of some simulated masquerade data blocks of user 9 is near to zero, resulting in missing alarms. With further investigation, it is found that most commands in the missing blocks of masquerade data of User 9 appeared also in the training data, and anomalies caused by these commands could not be identified in detection.

The false alarm rates and missing alarm rates are summarized in Table 5 in comparison with other 6 methods reported in Schonlau et al. (2001). Principal component analysis (PCA) is another alternative method for anomaly intrusion detection (Wang et al., 2004a–c, 2008). In order to further evaluate our model, we also present the testing results of PCA in Table 5 based on the same command data sets. PCA is one of the most widely used dimension reduction techniques for data analysis and compression (Duda et al., 2004). It transforms 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 datasets, the first several principal components contribute most of the variance in the original dataset, so that the rest can be disregarded with minimal loss of the variance for dimension reduction of the data (Duda et al., 2004; Golub and Van Loan, 1996). Since PCA seeks a projection that best represents the data in a least-square sense, in intrusion detection we use the squared Euclidean distance to measure the distance between a test vector and its reconstruction in the subspace spanned by several eigenvectors of the sample covariance matrix of the training data set.

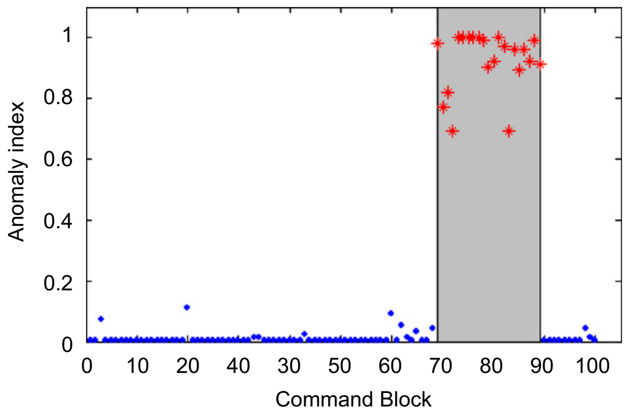

Fig. 6. Testing result of User 24. The gray shading indicates simulated masquerades data.

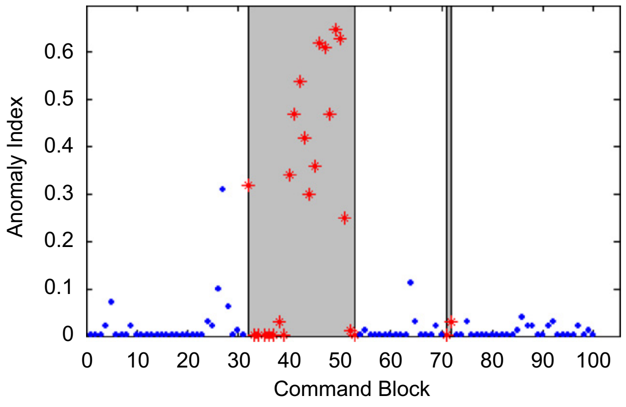

Fig. 7. Testing result for User 9. The gray shading covers simulated masquerades data.

Table 5
The missing alarm rates and false rates in comparison

Method	False alarm rate (%)	Missing alarm rate (%)
Compression	5.0	65.8
Sequence-match	3.7	63.2
IPAM	2.7	58.9
Hybrid multi-step Markov	3.2	50.7
Bayes one-step Markov	6.7	30.7
Uniqueness	1.4	60.6
PCA	1.9	70.6
NMF		
$\varepsilon = 0.2$	3.9	38.9
$\varepsilon = 0.3$	2.7	47.6
$\varepsilon = 0.4$	1.9	57.5

The descriptions of the PCA-based intrusion detection method can be found in details in Wang et al. (2004a–c, 2008).

From Table 5, it is seen that the NMF method is better than the first four methods in terms of both false alarm rates and missing alarm rates. The false alarm rate of NMF is better than Bayes one-step Markov, but the missing alarm rate is worse. The uniqueness approach is based on the idea that commands not previously seen in the training data may indicate attempted masquerade (Schonlau et al., 2001). This method is similar to our method and the false alarm rate is better. However, it considers the test statistics of each individual command in a block of data. If the data set is large, the computational time may increase significantly. The NMF method extracts the features when factorizing the original data matrix. The computation expense does not increase largely with the size of the data set since the NMF model considers the distribution of the data set and also the number of iterations in (2) and (3) does not change much with the matrix size. Our model is also better than the PCA method in terms of missing the alarm rate based on the same false alarm rate.

Table 6 shows the CPU time required for training and detection in comparison with the HMM method, the cross entropy method reported in Yeung and Ding (2003) and our NMF method proposed in this paper. The testing for the NMF method is performed on a host with P-IV 2.4 GHz CPU and 512 MB DDR memory, and that for the other two methods on an UltraSPARC 30 workstation (Yeung and Ding, 2003). From Table 6, it is seen that the NMF method is faster than the FC-HMM and LR-HMM in both the training and detection. Although the cross entropy method needs almost no training the CPU time for detection is much more than the NMF method.

Table 6
Comparison on computational efficiency with command data

Method	Number of commands for training	CPU time for training (s)	Number of 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
Cross entropy	10,826	0	10,981	14
NMF	10,000	1	15,000	2

4. Concluding remarks

In this paper, we presented a new fast anomaly intrusion detection method based on non-negative matrix factorization (NMF). The basic idea of the method is to divide the massive original data into smaller blocks and to calculate the frequency of individual elements in each block of data to construct data vectors so that large amounts of data can be represented and reduced. In this way, the anomaly intrusion detection problem is converted into the simpler problem of classifying normal or abnormal vectors. The NMF method is applied to reduce the high dimensionality of the data. Based on the fact that the summed frequency of each event in a data block is number 1 and the iterative updating rule that the sum of each entry in a new coefficient remains during the process of matrix factorization, the *anomaly index* of each block of data is characterized by a single number so that normal behaviors of computer systems are easily profiled and anomaly detection can be implemented in low dimensions with a very simple classifier. The above three steps reduce the computation and largely improve the efficiency performance of intrusion detection.

The NMF model is based on the frequency attribute of system calls and commands, in contrast to many other methods considering transition property, and there is no need to consider individual system calls or commands. It has wide applicability and can provide a general framework for practical intrusion detection with different sources of data such as audit logs, system calls, commands, etc. of large size. It is thus an effective model to fast process large amounts of data with low overhead.

The NMF model is implemented and tested with the *sendmail* system call data and live *lpr* system call data from UNM, live *ftp* system call data from the KLINNS laboratory of Xi'an Jiaotong University and the command data from the AT&T research lab. The testing results show that the NMF model is promising in terms of detection accuracy and computational efficiency.

There are disadvantages for the methods based on frequency property such as NMF. As mentioned, 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 though the sequences are quite different, NMF can hardly detect the anomaly. Nevertheless, the NMF method can be mixed with other methods based on transition analysis for improving the detection performance.

There is still much that remains to be explored. Our future work focuses on three aspects: (1) integrating the frequency property with the transition properties of system calls and commands to improve the detection accuracy; (2) testing our NMF model in practical systems with various data sources; (3) dealing with the problem of concept drift during model building and intrusion detection.

Acknowledgment

The authors thank Dr. Weixiang Liu, Research Institute of Tsinghua University in Shenzhen, China, for his valuable suggestions and comments.

References

Anderson D, Frivold T, Valdes A. Next-generation intrusion detection expert system (NIDES): a summary. Technical report SRI-CSL-95-07, Menlo Park, CA: Computer Science Laboratory, SRI International; May 1995.

Cai Z, Guan X, Shao P, et al. A rough set theory based method for anomaly intrusion detection in computer networks. Expert Syst 2003;18(5):251–9.

CERT Advisory CA-2001-07 File globbing vulnerabilities in various FTP servers [Online]. Available: ⟨http://www.cert.org/advisories/CA-2001-07.html⟩.

Cho SB, Park HJ. Efficient anomaly detection by modeling privilege flows using hidden markov model. Comput Secur 2003;22(1):45–55.

Denning DE. An intrusion-detection model. IEEE Trans Software Eng 1987;13(2):222–32.

Duda RO, Hart PE, Stork DG. Pattern classification, 2nd ed. Beijing: China Machine Press; 2004.

Feng L, Guan X, Guo S, et al. Predicting the intrusion intentions by observing system call sequences. Comput Secur 2004;23(5):241–52.

Forrest S, Hofmeyr SA, Somayaji A, et al. A sense of self for Unix processes. In: Proceedings of the 1996 IEEE symposium on research in security and privacy, Los Alamos, CA, 1996, p.120–8.

Golub GH, Van Loan CF. Matrix computation. Baltimore: Johns Hopkins University Press; 1996.

Hu W, Liao Y, Vemuri VR. Robust support vector machines for anomaly detection in computer security. In: Proceedings of the 2003 international conference on machine learning and applications (ICMLA'03), Los Angeles, California, June 2003.

Kruegel C, Kirda E, Mutz D, Robertson W, Vigna G. Polymorphic worm detection using structural information of executables. In: Eighth symposium on recent advances in intrusion detection (RAID), 2005.

Lane T, Brodley CE. Temporal sequence learning and data reduction for anomaly detection. In: Proceedings of the fifth ACM conference on computer & communication security, 1998.

Lee DD, Seung HS. Learning the parts of objects with nonnegative matrix factorization. Nature 1999;401:788–91.

Lee DD, Seung HS. Algorithms for nonnegative matrix factorization. Advances in neural information processing systems 13. Cambridge, MA: MIT Press; 2000.

Lee W, Stolfo S. Data mining approaches for intrusion detection. In: Proceedings of the seventh USENIX security symposium, Usenix Association, January 1998, p. 79–94.

Liao Y, Vemuri VR. Use of k-nearest neighbor classifier for intrusion detection. Comput Secur 2002;21(5):439–48.

Lunt T, Tamaru A, Gilham F, et al. A real-time intrusion detection expert system (IDES)–final technical report. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February, 1992.

Moore D, Shannon C. Code-red: a case study on the spread and victims of an internet worm. In: Proceedings of the 2002 ACM SICGOMM internet measurement workshop, Marseille, France, November 2002, p. 273–84.

Moore D, et al. The Spread of the Sapphire/Slammer Worm [Online]. Available: ⟨http://www.caida.org/analysis/security/sapphire⟩, February 2007.

Schonlau M, Theus M. Detecting masquerades in intrusion detection based on unpopular commands. Inf Process Lett 2000;76:33–8.

Schonlau M, Dumouchel W, Ju WH, et al. Computer intrusion: detecting masquerades. Stat Sci 2001;16(1):58–74.

Smaha SE. Haystack: an intrusion detection system. In: Proceedings of the IEEE fourth aerospace computer security applications conference, 1988.

Wang W, Guan X, Zhang X. 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, p. 2830–5.

Wang W, Guan X, Zhang X. A novel intrusion detection method based on principal component analysis in computer security. Advances in neural networks—ISNN 2004. In: International IEEE symposium on neural networks, Dalian, China. Lecture notes in computer science (LNCS), no. 3174. 2004, p. 657–62.

Wang W, Guan X, Zhang X. Profiling program and user behaviors for anomaly intrusion detection based on non-negative matrix factorization. In: Proceedings of 43rd IEEE conference on decision and control (CDC'04), Atlantis, Paradise Island, Bahamas, December 2004, p. 99–104.

Wang W, Guan X, Zhang X, Yang L. Profiling program behavior for anomaly intrusion detection based on the transition and frequency property of computer audit data. Comput Secur Elsevier 2006;25(7):539–50.

Wang W, Guan X, Zhang X. Processing of massive audit data streams for real-time anomaly intrusion detection. Comput Commun Elsevier 2008;31(1):58–72.

Warrender C, Forrest S, Pearlmutter B. Detecting intrusions using system calls: alternative data models. In: Proceedings of the 1999 IEEE symposium on security and privacy, 1999, p. 133–45.

Wespi A, Dacier M, Debar H. Intrusion detection using variable-length audit trail patterns. In: Proceedings of the third international workshop on the recent advances in intrusion detection (RAID'2000), no. 1907, Lecture notes on computer scinece, October 2000.

Yeung DY, Ding Y. Host-based intrusion detection using dynamic and static behavioral models. Pattern Recognition 2003;36(1):229–43.

Zou CC, Gong W, Towsley D. Code red worm propagation modeling and analysis. In: Proceedings ninth ACM conference on computer and communication security, November 2002.