

Profiling Program and User Behaviors for Anomaly Intrusion Detection Based on Non-negative Matrix Factorization

Wei Wang, Xiaohong Guan, *Senior Member, IEEE*, and Xiangliang Zhang

Abstract—Profiling program and user behaviors is an effective approach for detecting hostile attacks to a computer system. A new model based method by Non-negative Matrix Factorization (NMF) is presented in this paper to profile program and user behaviors for anomaly intrusion detection. In this new method, the audit data streams obtained from sequences of system calls and UNIX commands are used as the information source. The audit data is partitioned into segments with a fixed length. Program and user behaviors are, in turn, measured by the frequencies of individual system calls or commands embedded in each segment of the data, and NMF is applied to extract the features from the blocks of audit data associated with the normal behaviors. The model describing the normal program and user behaviors is built based on these features and deviation from the normal program and user behaviors above a predetermined threshold is considered as anomalous. The method is implemented and tested with the system call data from the University of New Mexico and the Unix command data from AT&T Research lab. Experiment results show that the proposed method is promising in terms of detection accuracy, computational expense and implementation for real-time intrusion detection.

Index Terms—Anomaly detection, intrusion detection, computer security, non-negative matrix factorization, profiling, system call, shell command.

I. INTRODUCTION

Intrusion Detection is an important technique in the defense-in-depth network security framework and a hot topic in computer network security in recent years. In general, Intrusion Detection System (IDS) is classified into two categories depending on the modeling methods used: misuse detection and anomaly detection. In misuse detection, evidences of malicious behaviors are matched against pre-defined descriptions of attacks, or signatures.

The research presented in this paper was supported in part by the National Outstanding Young Investigator Grant (6970025), National Natural Science Foundation (60243001) and 863 High Tech Development Plan (2001AA140213) of China.

Wei Wang (e-mail: wwang@sei.xjtu.edu.cn), Xiaohong Guan (e-mail: xhguan@sei.xjtu.edu.cn) and Xiangliang Zhang (e-mail: xiangliangzhang@126.com) are with the State Key Laboratory for Manufacturing Systems (SKLMS) and Center for Networked Systems and Information Security (CNSIS), Xi'an Jiaotong University, Xi'an, China. Xiaohong Guan is also with the Center for Intelligent and Networked Systems, Tsinghua University, Beijing, China.

Although misuse 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 profile of normal behavior and attempts to identify the patterns or activities that deviate from the normal profile. Based on the concept of profiling normal behaviors, an important feature of anomaly detection is that it can detect unknown attacks. However it may also cause significant false alarm since the model to describe complete normal behaviors is very difficult to obtain.

There are multiple levels that intrusion detection can be built upon in the real computer network system. A big challenge is to choose features that best characterize the user or system usage patterns, so that abnormal activities are clearly distinguished from normal activities. Many types of data sources can be used including Unix Shell commands, audit events, keystroke, system calls, and network packages, etc. Selecting data source is the first crucial step in building a profiling method for intrusion detection. Early studies on anomaly detection mainly focus on profiling system or user behaviors from monitored system log or accounting log data ([1, 2, 3]). Examples of the data derived from these logs are: CPU usage, time of login, duration of user session, names of files accessed, etc. More recently, Schonlau et al. attempts to detect masquerades by profiling user behaviors using truncated command sequences ([4]). Experiments with six masquerade detection techniques: Bayes one-step Markov, Hybrid multi-step Markov, IPAM, Uniqueness, Sequence-Match and Compression, are performed and compared in [5]. Lane et al. proposed a learning algorithm for analyzing user shell command history to profile normal user behaviors and detect anomalies ([6]). The algorithm attempts to address the “concept drift” problem when the normal user behavior changes.

In recent years, a lot of research activities in anomaly detection focus on learning program behaviors and building the profiles with system call sequences as data sources. Forrest et al. introduced a simple anomaly detection method based on monitoring the system calls invoked by active and privileged processes ([7]). The profile of normal behavior is built by enumerating all fixed length of unique and contiguous system calls that occur in the training data, and

unmatched sequences in actual detection are considered abnormal. This approach was extended by various other methods. Lee et al. used data mining approach to study a sample of system call data to characterize the sequences contained in normal data by a small set of rules. In monitoring and detection, the sequences violating those rules are treated as anomalies ([8]). Warrender et al. proposed Hidden Markov Model (HMM) method for modeling and evaluating invisible events based on system calls ([9]). Wespi et al. further developed Forrest's idea and proposed a variable length approach ([10]). Asaka et al. developed another approach based on the discriminant method in which an optimal classification surface is first learned from samples of the properly labeled normal and abnormal system call sequences ([11]). The surface is then used as a basis to decide the normality of a new system call sequence. Rough Set Theory method is applied by our research group to learn the rule set modeling the normal behaviors with much smaller size of training datasets and improved detection accuracy ([12]). We have developed another Principle Component Analysis based method for anomaly intrusion detection with less computation efforts ([13]). Yihua Liao et al. proposed an algorithm based on k-Nearest Neighbor method and profiled the program behavior according to the trace of each process independently ([14]).

Although there are many impressive progresses on program profiling, there are still many open issues to be resolved. A protected computer system could produce a large amount of audit data. For example, in the experiments on *sendmail*, 112 messages would produce a combined trace with the length of over 1.5 million system calls ([7]). Processing the high dimensional data is computationally costly and it is difficult to detect an intrusion in real time before a substantial damage to the computer system is done.

Profiling user behaviors is as important as profiling programs. It is an effective approach to capture the internal attacks, which may constitutes more than one third of the corporate security incidents ([15]). In a UNIX or Linux system, shell command sequences are easy to collect with little overload. Therefore, UNIX shell command is a good data source for profiling user behaviors and capturing user behaviors is not very costly. On the other hand, user behaviors vary very widely and profiling user behavior is more difficult in comparison with profiling program behaviors. The hackers may even slowly adopt their behaviors to match the normal profile and escape the alarms of the IDS.

In this paper, we profile both program behaviors and user behaviors simultaneously for anomaly intrusion detection with system call sequences and Unix command sequences as the data sources. A novel method, Non-negative Matrix Factorization (NMF) developed by Lee and Seung to obtain a reduced representation of data in pattern recognition ([16]),

is applied to build the normal behavior models. It differs from other methods by its use of non-negativity constraints in representation and is suitable for analyzing the frequency characteristics of individual system calls or commands in one block of audit data since frequencies are all positive.

NMF has been successfully applied to image representation such as human face recognition, document classification, and color categorization ([17]). In this paper, we apply NMF to build normal behavior models in anomaly intrusion detection. An obvious advantage of NMF method is its computational efficiency. Profiling program behavior based on NMF avoids the time-consuming training process on individual program profiles separately and the computational costs of learning behaviors are significantly reduced. A large amount of high dimensional data is used on our experiments, and program or user behaviors were profiled by the frequencies property (other than local ordering property) of the system calls or commands. The large amount of data is partitioned into segments with a fixed length. The numbers of occurrences of individual elements in each segment of the data are counted and placed column by column as data vectors to construct a matrix representation. Using the NMF, the data matrix can be factorized into two simple matrices, one as the base and the other as encoding coefficients representing normal or abnormal features. The normal behaviors are profiled based on the features extracted by NMF. If the features contained in any segment of data deviates significantly from those of the normal behaviors learned in training datasets, the data is considered from the anomalous behaviors.

II. PROFILING MODEL BASED ON NON-NEGATIVE FACTORIZATION

A. Non-negative Matrix Factorization

Given an initial dataset expressed by a $n \times m$ matrix V , where each column is an n -dimensional non-negative vector of the original data (m vectors), it is possible to find two new matrices (W and H) in order to approximate the original matrix $V \approx WH$ ([16]), or

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

The dimensions of the factorized matrices W and H are $n \times r$ and $r \times m$, respectively. Usually, r is chosen to be smaller than n and m , or $(n+m)r < nm$, so that W and H is smaller than original matrix V . (1) can be rewritten column by column as $v \approx Wh$, where v and h are the corresponding columns of W and H . In other words, each data vector v is approximated by a linear combination of the columns of W , weighed by the components of h . Therefore, W is regarded as containing a basis that is optimized for the linear approximation of V ([18]). Each column of matrix W contains basis vectors and the column of H contains encoding coefficients. The data matrix V 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 $V \approx WH$, one of the iterative approaches is given by the following rules ([18]):

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}} \quad (2)$$

$$W_{ia} \leftarrow W_{ia} \frac{(V H^T)_{ia}}{(W H H^T)_{ia}} \quad (3)$$

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

B. Profiling Model Based on NMF

Profiling program or user behaviors based on NMF includes three steps: data preparation, features extraction and classification and decision, as showed in Figure 1.

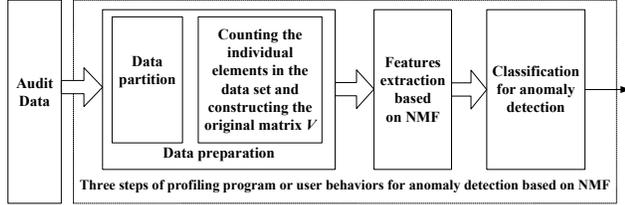


Fig. 1. Three steps of profiling program and user behaviors based on NMF.

1) *Data Preparation*: Suppose an observation dataset is divided into m blocks, each contains l elements, and there are totally n unique elements from the dataset. The observation data can be expressed by a $n \times m$ Matrix V , where each element V_{ij} stands for the frequency of i -th unique element occurs in the j -th block, and

$$l = \sum_{i=1}^n V_{ij}, \quad j = 1, 2, 3, \dots, m \quad (4)$$

2) *Features Extraction*: By the iterative updating rules (2) and (3), the training dataset for normal behaviors represented by $V_{n \times m}$ are approximately factorized into the $n \times r$ bases W and $r \times m$ coefficients H . H represents the features learned from the original matrix V , where the features associated with each block of data are represented by column vector t in H .

From (2) and (3), it is clear that

$$\sum_{i=1}^r H_{ij} = \sum_{i=1}^n V_{ij}, \quad j = 1, 2, 3, \dots, m \quad (5)$$

From (4), (5) is also written as

$$\sum_{i=1}^r H_{ij} = l, \quad j = 1, 2, 3, \dots, m \quad (6)$$

Based on (6), it is found that the sum of the elements in each column of H is a constant equal to the fixed length by which the data set is partitioned. Based on this property, H

can be normalized via linear regression ([19, 20]) and each column of H can be represented by a single value.

Given m r -dimension vectors in a matrix $H_{r \times m}$, we can find a parameter vector $C_{1 \times r}$ by linear regression such that

$$y = CH \quad (7)$$

where $y = 1_{1 \times m}$. In this way, each column vector t in H is represented by value 1. Therefore, each block of normal training data can be represented by a single value 1 and the normal behaviors are thus profiled.

3) *Classification and decision*: Using the basis W learned from normal training dataset, we obtain the coefficients H' from the actual dataset by (2).

For a column vector t' in H' , using the parameter vector C defined by (7), we can normalize it by $t_1 = Ct'$. The features of the corresponding block of the actual dataset t' are normalized to a single value t_1 . We then consider t_1 as the *detection index* of the block. Compared t_1 with 1, the abnormal behavior is distinguished if

$$|(t_1 - 1)| > \varepsilon \quad (8)$$

where ε is a predefined threshold.

III. PROFILING PROGRAM BEHAVIORS

A. Data Set

To facilitate comparison, we use system call sequences of a UNIX-based host machine at University of New Mexico (UNM), collected by Forrest et al. ([7]) since they were widely used for testing many other methods. The data set can be downloaded at <http://www.cs.unm.edu/~immsec/>, and the procedures of generating the data are described in [7]. The experiments are based on two datasets of *sendmail* and their descriptions are summarized in Table I.

TABLE I
THE DATASETS USED IN THE EXPERIMENTS

Normal sequences	2 sequences of <i>sendmail</i> daemon and 2 other sequences (plus and queue), generated by <i>sendmail</i> under various normal conditions.
Abnormal sequences	4 syslog attacks, 2 decode attacks and 1 unsuccessful attacks based on attack scripts called sm5x.

B. Profiling Program Behaviors and Anomaly Detection

The preprocessing of the datasets is performed in the following two steps:

(1) Partition the separated datasets into small blocks of system calls by a fixed length and combine them as one large dataset;

(2) Count the numbers of occurrences of individual system calls in each block and place them column by column to construct matrix V .

There are totally 16 abnormal sequences in the *sendmail* system call data and the largest length of these 16 abnormal

sequences is 1861 and the smallest is 254. The most abnormal sequences have about 1500 system calls. Therefore block size is determined as 1500. In actual applications, the abnormal sequences may not be known beforehand and block sizes are usually determined by experience. The 56×2247 matrix V is formed for the normal behaviors of *sendmail* processes. In the 16 abnormal sequences of system calls in *sendmail*, 7 sequences have more than 1500 system calls, and only the first 1500 system calls data in the 7 abnormal sequences are used. We added them to the matrix and get a 56×2254 matrix V with the first 2247 columns for the normal data and the last 7 columns for abnormal data. The descriptions are shown as Table II.

The first 500 blocks of data in V are used for training and the rest for testing. In testing anomaly detection, each block of system calls, represented by the corresponding predict value t_1 , is classified as normal or abnormal according to the decision rules defined by (8). For the purpose of comparison, different r and threshold ϵ are selected in the experiments.

TABLE II
DESCRIPTION OF THE DATA SET IN THE EXPERIMENTS

Numbers of system calls in the data set	Numbers of unique system calls in the data set	Numbers of normal blocks	Numbers of abnormal sequences
Over 3.3 million	56	2247	7

C. Results and Analysis

Good testing results are obtained by applying NMF to profile program behaviors for anomaly detection. Figure 2 shows the experimental results with $r=20$.

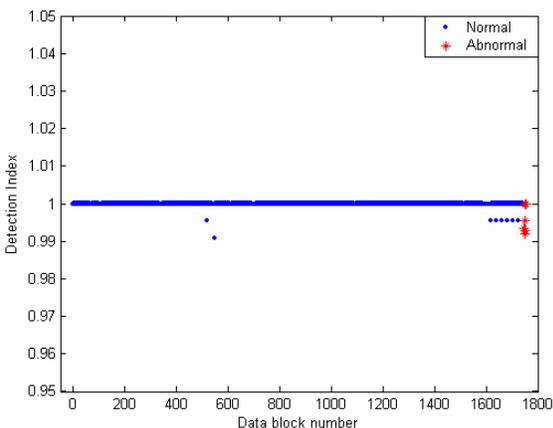


Fig.2. Experimental results on *sendmail* system call data with $r = 20$. The y-axis represents the detection index and x-axis represents for the blocks of system call data. The star (*) and the dot (•) stands for the detection index for abnormal data and normal data respectively.

Table III shows the successful detection and false alarm rate in the experiments. From table III, we can see that the performance of detection is stable, and r is not a sensitive parameter to the experimental results. The threshold ϵ is determined by experience as a trade-off among detection accuracy and false alarm rate. It can be made low enough to obtain 100% detection rate but the false alarm rate could also be high. In our experiment, two ϵ values are selected for comparison.

TABLE III
DETECTION RATE AND FALSE-ALARM RATE WITH DIFFERENT r AND ϵ

r	ϵ	Detected Abnormalities (7 abnormal sequences in total)	False Alarms (1747 normal sequences in total)	Detection rate	False - alarm rate
1	ϵ_1	7	47	100%	2.69%
	ϵ_2	4	8	51.7%	0.46%
2	ϵ_1	7	47	100%	2.69%
	ϵ_2	4	8	51.7%	0.46%
10	ϵ_1	7	47	100%	2.69%
	ϵ_2	4	8	51.7%	0.46%
30	ϵ_1	7	47	100%	2.69%
	ϵ_2	4	8	51.7%	0.46%

IV. PROFILING USER BEHAVIORS

A. Dataset

The UNIX commands datasets collected in AT&T's Shannon Research Laboratory are used in our experiments ([4]). The experimental data consist of user names and the associated command sequences (without arguments). Fifty users are included with 15000 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 fifty 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 [4] for more about the details of the contamination procedure.

B. Profiling User Behavior and Masquerades Detection

Suppose the training dataset of one user contains n unique commands, it is represented by matrix $V_{n \times 50}$. Given parameter r , the matrix $V_{n \times 50}$ can be approximately factorized into the $n \times r$ W as the base and $r \times m$ coefficients H by the iterative update rules (2) and (3). Using the basis W learned from the normal training data set, we can get the coefficients H' on the testing data set by the updated rules (2). We then apply the classification rule defined by (8) to determine if a command sequence is as normal or abnormal.

C. Results and Analysis

Similar to the system call data, experiments are performed on the UNIX commands data for different r which is not a sensitive to the outcomes. The testing results on User 24 are showed in Figure 2 as an example. It is observed that simulated masquerade data are located at Blocks 69~89 with shade background in the Figure and our method can easily catch them all.

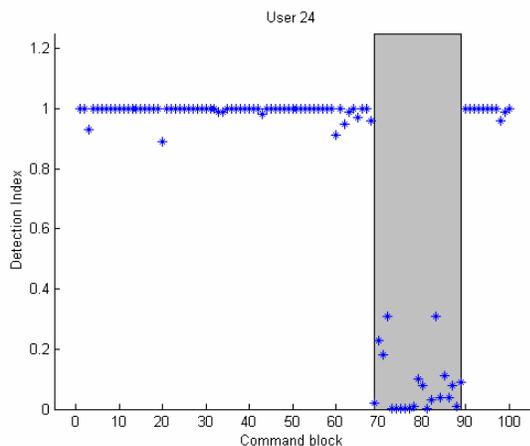


Fig.3. Experimental result of User 24. The dark background indicates simulated masquerades data.

TABLE IV
THE FALSE-ALARM AND MISSING ALARM RATE WITH COMPARISON.

Method	False alarm (%)	Missing alarm (%)
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
NMF	$\epsilon=0.2$	3.9
	$\epsilon=0.3$	2.7
	$\epsilon=0.4$	1.9

The false alarm and missing alarm rates are listed in Table IV in comparison with other 6 methods reported in [5]. All the 6 other methods are computationally more costly than NMF method. The compression approach takes into account the reversible mapping of data appended from training to test. In the Sequence-match method, a similarity measure is considered for each new command. The two methods consider each command in each block of data. Bayes one-step Markov, IPAM and Hybrid multi-step method are based on transition property of command sequences and it is inevitable to analyze command one by one in-depth in each block of data. On the other hand, NMF based method only consider the frequency property and each block of the data is treated as a whole. Therefore the computation is significantly reduced.

In Table IV, it is seen that NMF method is better than the first four methods in terms of both false alarm and missing alarm rates. The false alarm rate of NMF is better than Bayes one-step Markov but the missing alarm rate is worse. As mentioned, NMF method is computationally more efficient. The uniqueness approach is based on the idea that commands not previously seen in the training data may indicate attempted masquerade ([5]). 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 each block of the data. If the data set is large, the computational time may increase significantly. NMF method extracts the features when factorizing the original data matrix. The computation expense does not increase with the size of the dataset since the required iterations in (2) and (3) does not change much with the matrix size.

V. CONCLUDING REMARKS

In this paper, we present a new algorithm based on Non-negative Matrix Factorization (NMF) for profiling program and user behaviors for anomaly intrusion detection. The characteristics of the method and concluding remarks are summarized as follows.

Dividing the massive original data into small blocks and counting the numbers of individual elements in each block of data to construct a simple matrix so that a large amount of data can be represented and reduced. The detection results are not sensitive to how the matrix is factorized. Therefore we often chose $r = 1$, and features of each block of data are extracted and then expressed by a single number so that normal program and user behaviors can be easily profiled and anomaly detection can be easily implemented without any additional classifier. It is thus an effective method to process a mass of audit data in real-time with low overhead.

The method takes into account the frequency property of system calls and commands, while many other methods consider transition property. Since there is no need to consider individual system calls or commands, the computational cost of NMF method is low and suitable for real-time intrusion detection.

Another advantage of NMF method is its wide applicability. It can process all kind of the audit data even with mixture, system call, UNIX commands, etc of large size. It is effective tool applicable to a broad range of anomaly intrusion detection.

However, there are also disadvantages for the methods analyzing frequency property such as NMF. 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 but the sequences are quite different, NMF can hardly detect the anomaly. The other methods based on transition analysis may detect such the anomaly without difficulty.

The method is implemented and tested with the system call data from the University of New Mexico and the Unix command data from AT&T Research lab. It is able to

process massive data with many varieties in real-time and experiment results show that the method is promising in terms of detection accuracy, computational expense and implementation for real-time intrusion detection.

Research in progress is on how to mix the frequencies property with the transition information on the sequences of system calls and commands to achieve lower false alarm and missing alarm rates. The ways how to partition data blocks and determine block sizes are also being investigated.

ACKNOWLEDGMENT

We thank Weixiang Liu, Artificial Intelligence and Robotics Institute of Xi'an Jiaotong University for his fruitful suggestions and comments. We also would like to thank anonymous reviewers for their valuable comments.

REFERENCES

[1] S. E. Smaha. *Haystack: An intrusion detection system*. In Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference, 1988.

[2] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and 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.

[3] D. Anderson, T. Frivold, and 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.

[4] M. Schonlau and M. Theus. *Detecting Masquerades in Intrusion Detection Based on Unpopular Commands*. Information Processing Letters, 76, pp.33-38, 2000.

[5] M. Schonlau and W. Dumouchel, W.-H. Ju, A. F. Karr, M. Theus and Y. Vardi. *Computer Intrusion: Detecting Masquerades*. Statistical Science, Vol.16, No. 1, pp.58-74, 2001.

[6] T. Lane and C. E. Brodley. *Temporal sequence learning and data reduction for anomaly detection*. In Proceedings of 5th ACM Conference on Computer & Communication Security, 1998.

[7] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. *A sense of self for Unix processes*. Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, Los Alamos, CA, pp.120-128, 1996.

[8] W. Lee and S. Stolfo. *Data Mining Approaches for Intrusion Detection*. In Proceedings of the 7th USENIX Security Symposium, Usenix Association, pp.79-94, January 1998.

[9] C. Warrender, S. Forrest and B. Pearlmutter. *Detecting Intrusions Using System Calls: Alternative Data Models*, Proceedings of 1999 IEEE Symposium on Security and Privacy, pp.133-145, 1999.

[10] A. Wespi, M. Dacier and H. Debar. *Intrusion Detection Using Variable-Length Audit Trail Patterns*. In Debar, H., Me, etc editors, Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000), No. 1907 in LNCS.

[11] M. Asaka, T. Onabuta, T. Inoue, S. Okazawa and S. Goto. *A New Intrusion Detection Method Based on Discriminant Analysis*. IEICE Transactions on Information and Systems E84D (5): 570-577, 2001.

[12] Z. Cai, X. Guan, P. Shao, Q. Peng and G. Sun. *A Rough Set Theory Based Method for Anomaly Intrusion Detection in Computer Networks*. Expert Systems, vol. 18, No. 5, pp.251-259, Nov 2003.

[13] W. Wang, X. Guan and Xiangliang Zhang. *A Novel Intrusion Detection Method Based on Principle Component Analysis in Computer Security*. In Proceedings of the International IEEE Symposium on Neural Networks, Dalian, China. Lecture Notes in Computer Science, Vol. 3174, pp. 657-662, Aug 2004.

[14] Y.H. Liao and V. R. Vemuri, *Use of K-nearest Neighbor Classifier for Intrusion Detection*. Computer & Security, vol. 21, No 5, pp.

439-448, 2002.

[15] Computer Security Institute, CSI/FBI computer crime and security survey results quantify financial losses, Computer Security Alert 181 (1998).

[16] D. D. Lee and H. S. Seung. *Learning the parts of objects with nonnegative matrix factorization*. Nature, 401: 788-791, 1999.

[17] W. Liu, N. Zheng and X. Li. *Review on Nonnegative Matrix Factorization*. Technical Report. Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, 2004.

[18] D. D. Lee and H. S. Seung. *Algorithms for nonnegative matrix factorization*. In T. Leen, T. Dietterich, and V. Tresp (eds.). Advances in Neural Information Processing Systems 13. MIT Press: Cambridge, MA, 2000.

[19] D. M. J. Tax, *One-class classifiers*. Phd thesis. Pattern Recognition Group on the Delft University of Technology, 2001.

[20] P. J. Rousseeuw and A.M. Leroy, *Robust regression and outlier detection*. John Wiley & Sons, 1987.