# SOM-Based Anomaly Intrusion Detection System

Chun-dong WANG, He-feng YU, Huai-bin WANG, Kai LIU
School of Computer Science and Technology, Tianjin University of Technology, Tianjin 300191, China
michael3769@163.com, emaif@163.com, hbwang@tjut.edu.cn, liukai@163.com

**Abstract.** In this paper, a SOM-based anomaly intrusion detection system is proposed, which can contract high-dimension data to lower, meanwhile keeping the primary relationship between clustering and topology. During the experiment, the theory of SOM is used to train three SOMs on the layers of system, process and network. Although our experiment environment is simpler than the real one, the result shows that it has its reference value for us to build intelligent IDSs. Through the analysis of the monitoring results on the three layers from the hacking tools (NMAP, HYDRA), it is suggested that the approach of detecting and the parameters chosen be correct and effective.

**Keywords:** SOM; neural network; anomaly-based intrusion detection; U-matrix; cluster.

## 1 Introduction

Along with increasing popularization of the Internet day by day, continuous updating and variety of attack behaviors make the traditional rule-based IDS gradually lose its ideal effect. SOM-based IDS, as the representative of the new generation of intelligent IDS, compared with the traditional IDS, has the following special advantages:

(1) The analysis technique of the traditional IDS is mainly based on the model of statistics [1], and depends on several assumptions. SOM-based IDS can be trained through a great deal of instances, can learn knowledge by itself and acquire the ability of prognostication. The whole process is completely abstract calculation, with no emphasis on the assumption of the distribution of the data.

(2) The traditional IDS depicts attack characteristics to be limited by a fixed sequence, and the threshold value is mostly based on the experience. False positive and false negative usually happen and it can not easily identify new attack methods. In the terms of the ability of self-applicable and fault tolerant [2], SOM-based IDS need not understand the detail of knowledge, and can master the inherent relationship of each generous character of the system by itself. After mastering the normal working mode [3] of the system, SOM-based IDS can react to all affairs which deviate from the normal working mode, and then discover new attacks.

## 2 Self-organizing maps (SOMs)

SOM was proposed by professor Kohonen, the neural network expert in 1981. Early researches on neural network-based IDS mostly adopted BP (Back-Propagation) neural network [4-5] for modeling, as well as multi-layer perceptron and Hamming, but they all have restrictions and weaknesses [6]. By contrast with the above mentioned neural networks, the best advantage of SOM is the ability of unsupervised learning, which can transplant the system to new surroundings, and the training data has no marks.

### 2.1 About SOM Algorithm

The algorithm of SOM is recursive. First, every neuron corresponds to a N-dimensions vector Wi=[$w_{i1}$,$w_{i2}$,⋯,$w_{iN}$]. At every stage of training, sampling vector $X_k$=[$x_1$,$x_2$, ⋯,$x_N$] is selected from the training set randomly, then calculate the distance between $X_k$ and all the weight vectors. c is the BMU(best-matching unit), and the minimum distance between c and $X_k$ is:

$$\| X_k - W_c \| = \min_{i=1}^{M} \| X_k - W_i \| \tag{1}$$

Next, update the weight vector of the neuron which is in neighbourhood zone of the winner cell's topology. The rule is as follow:

$$W_i(k+1) = \begin{cases} W_i(k) + \alpha(k)[X_k - W_i(k)], & i \in N_c(k) \\ W_i(k), & i \notin N_c(k) \end{cases} \tag{2}$$

In Formula 2, $N_c$ refers to neighbourhood zone of the centre neuron $w_c$. In the process of learning, the initialization of $N_c(k)$ can be big, then contracts gradually, as follow:

$$N_c = INT(N_c(0)(1 - k/L)), \quad k = 0,1,2,...L \tag{3}$$

In Formula 3, $N_c(0)$ means the initial neighbourhood radius, L, the times of the iteration, INT（.）, the integral function. $N_c$(k1), $N_c$(k2), $N_c$(k3) stand for the topology neighbourhood zone of the winner cell whose iterative times are $k_1$, $k_2$, $k_3$（$k_1 < k_2 < k_3$）.

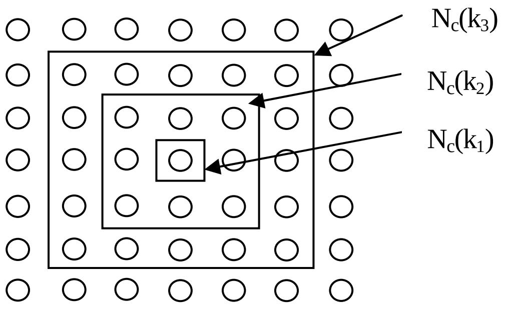

**Fig.1** topology adjacent domains on two- dimension network

Usually, the learning rate $\alpha(k)$ (0<$\alpha(k)$<1) is a constant, which is close to 1.0 in the beginning, then lessens gradually. For example, $\alpha(k)$ can be 0.8(1-k/L). With the increase of the times of the iteration, $\alpha(k)$ tends to zero, which ensures the learning process to refrain from rash action.

## 2.2 The steps of the learning algorithm of SOM

The concrete steps of the learning algorithm of SOM are as follows:

**Step 1.** Setting variables and parameters: Let $X(k) = [x_1(n), x_2(n),\cdots, x_N(n)]^T$ be the input vector, or training sample, $W_i(k)=[w_{i1}(n), w_{i2}(n), \cdots, w_{iN}(n)]T$ be the weight vector, i=1,2, ⋯, M, and the total times of iteration be L.

**Step 2**. Initialization: Initialize the weight vector $W_i$ with a small random number in a certain interval. Let the neighbourhood radius be $N_c(0)$; the learning rate be $\alpha(k)$; and then normalize weight vector $W_i(0)$ and all the input vector X.

$$X^{'} = \frac{X}{\| X \|} \tag{4}$$

$$W_i^{'}(0) = \frac{W_i(0)}{\| W_i \|} \tag{5}$$

In the above formulas, $\| W_i(0) \| = \sum_{j=1}^{N} [w_{ij}(0)]^2$ and $\| X \| = \sum_{j=1}^{N} (x_i)^2$ are Euclidean norm of the weight vector and input vector.

**Step 3.** Data sampling. Select training samples X’ from the input space.

**Step 4.** Approximate matching: According to the standard of the minimum Euclidean distance:

$$\| X^{'} - W_c^{'} \| = \min_i \| X^{'} - W_i^{'} \| \; i = 1,2,...,M \tag{6}$$

select winner cell c, implement the competitive process of neurons.

**Step 5.** Updating: Update the weight vectors of the cordial neuron, who are in the topology neighbourhood zone of the winner cell $N_c(n)$ under the following rules:

$$W_i^{'}(k+1) = W_i^{'}(k) + \alpha(k)[X - W_i^{'}(k)] \tag{7}$$

**Step 6.** Updating the learning rate $\alpha(k)$ and the topology neighbourhood zone, and then normalize the weights after learning.

$$\alpha(k) = \alpha(0)(1 - \frac{k}{L}) \tag{8}$$

$$N_c(n) = INT\left[N_c(0)\left(1 - \frac{k}{L}\right)\right] \tag{9}$$

$$W_i^{'}(k+1) = \frac{W_i(k+1)}{\| W_i(k+1) \|} \tag{10}$$

**Step 7.** Judging whether the times of the iteration k exceeds L or not, if k<= L then turn to step 3, otherwise end the process of iteration.

During the training, output neurons are sorted through adjusting their weight vector, in order to be close to the probability density. Though produced randomly at the beginning, the weight vector of the output neurons get closer and closer to the distribution of the input data after long time running, through updating the weight vector continuously.

## 3 Data sampling, preprocess, standardization and training

### 3.1 Data sampling and preprocess

In this paper, three hosts are used to constitute an LAN. One of the hosts, running Red hat 7.2, works as the testing platform. We mainly sample and preprocess the data from three layers (System layer, Process layer and network layer). The procedure is carried out by the shell program. First of all, the sampling data consist of 1000 sets of relevant data, which imitate the normal operation, and the sample interval is 10 seconds. The normal operations include the usage of familiar basic orders and applications, compilation of programs, application of ftp and telnet, and Web browse etc. Although time and amount of sampling are not enough, we imitated a great deal of actual operation during samplings, which is to a degree representative. As is the choice of parameters, we mainly consider those parameters affecting the system most and likely to be abnormal, while attack happening or after.

On the system layer, we choose 9 Characteristic Parameters:

-S1: usage of virtual memory (kb)
-S2: spare memory (kb)
-S3: exchange speed from disk to memory (kb/s)
-S4: exchange speed from memory to disk (kb/s)
-S5: read-in speed of block device (kb/s)
-S6: write-out speed of block device (kb/s)

-S7: times of interruption per second (including timer interruption)
-S8: CPU processing time of user process (%)
-S9: CPU processing time of system process (%)

On the process layer, we choose 6 Characteristic Parameters:
-P1: total number of process
-P2: number of process at the state of running
-P3: CPU time occupancy of every process at the state of running (%)
-P4: memory occupancy of every process at the state of running (%)
-P5: virtual memory occupancy of every process at the state of running (%)
-P6: initial time of every process at the state of running

On the network layer, we choose 6 Characteristic Parameters:
-N1: number of TCP connection
-N2: port number of the connection
-N3: IP address of the connection
-N4: connection state
-N5: number of error packet accepted
-N6: number of error packet sent

Besides numeral quantity, there is also non-numerical quantity in the characteristic data. For numeral quantity, it keeps the initialized value. For IP address, it intercepts the last part. For the non- numerical quantity, it is inverted to the numeral quantity. For example, for the state of TCP, we respectively use: 1,2,3,4,5,6,7,8,9,10,11,12 to represent: Established, SYN-Sent, SYN-Recy, Closed, Listen etc; for the starting time of the process, we change the notation of ":" into ".".

The corresponding parameter is mainly obtained from the filtration of the results, which come from different UNIX system commands, including: Vmstat, ps, Netstat, top and so on. The programme is completed with script language. The interception, conversion and coordination of the data are finished under these commands, including: sed, awk, cat, cut, grep etc. The data of sampling is stored in three files (systemlayer.dat, processlayer.dat, networklayer.dat) according to a certain format.

## 3.2 Standardization

Among the characteristic values selected, the differences are very obvious. So, in order to balance the effects of the training result, we can standardize every characteristic value into the area of 0~1, which is the demand of the application of SOM. The formula of standardization is as follows:

$$nv[i] = \frac{v[i]}{\sqrt{\sum_{K} v[k]^2}} \tag{11}$$

In Formula 11, nv[i] is the standardized value of i, v[i], the value size of i, and K, the number of the characteristic vector.

## 3.3 Data training

Map training is carried out according to the above-mentioned algorithm, mainly using SOM_PAK [7-8]. Three Maps should be trained on three different layers in our experiment. The number of neurons in each output layer of each Map is 12x8=96. The process of training is divided into two stages. At the first stage, the weigh vector of each neuron is sorted. At the beginning, the neighbourhood radius is chosen relatively bigger, generally equaling to the diameter of the Map, finally changing into 1, and the bigger learning rate should also be chosen, gradually changing into 0 with the increase of the training times. In order to make the weigh vector of each neuron more accurate, the second stage should be adjusted. At this stage, the smaller learning rate and the neighbourhood radius should be chosen correspondingly.

U-matrix (unified distance matrix) is a visual method of SOM cluster structure. The U-matrix chart shows the distance between the weight vector of a certain neuron and that of its adjacent neuron. Usually, different gray levels are used to express the size of the distance. Fig2, 3, 4 are three U-Matrix figures, respectively expressing the results of SOM training on three layers (system layer, process layer and network layer). Each small black mark in the figure means a neuron unit.

**Table 1.** Training parameters of SOMs on layers of system, process and network

| | System layer | Process layer | Network layer |
|---|---|---|---|
| Dimensions of input vector | 9 | 6 | 6 |
| Topology | Rhombus | Rhombus | Rhombus |
| Adjacent domain function | Bubble | Bubble | Bubble |
| Dimension of Map’s X direction | 12 | 12 | 12 |
| Dimension of Map’s Y direction | 8 | 8 | 8 |
| Function of learning rate | Linear | Linear | Linear |
| Times of training at phase one | 10000 | 10000 | 10000 |
| Initial value of learning rate at phase one | 0.1 | 0.3 | 0.3 |
| Initial radius at phase one | 10 | 10 | 10 |
| Times of training at phase two | 100000 | 100000 | 100000 |
| Initial value of learning rate at phase two | 0.02 | 0.01 | 0.02 |
| Initial radius at phase two | 3 | 3 | 3 |

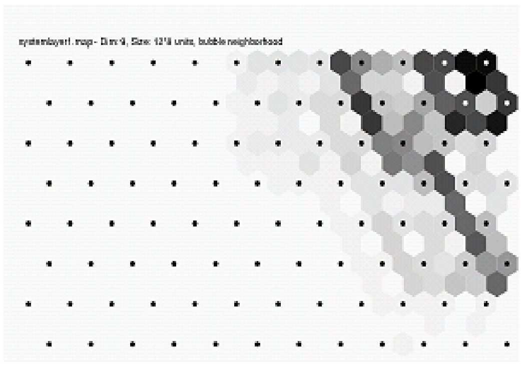

**Fig.2** U-Matrix map of system layer

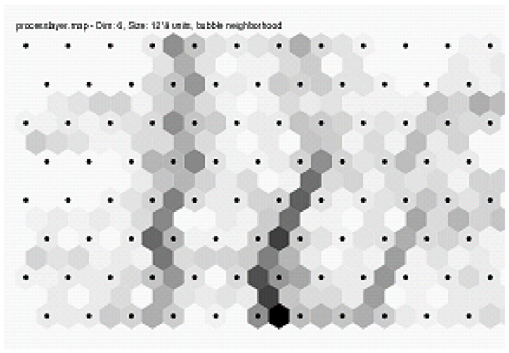

**Fig.3** U-Matrix map of process layer

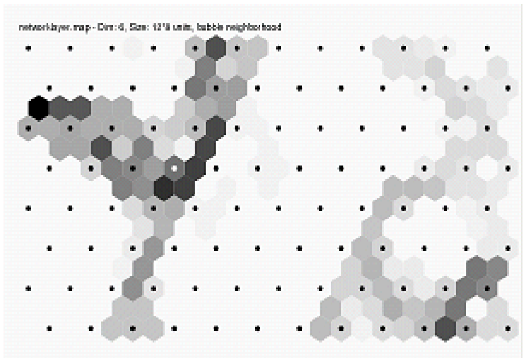

**Fig.4** U-Matrix map of network layer

In the learning algorithm of SOM, different choices of training parameters have different influence on training results. In this paper, different Maps are mainly obtained through changing the training stages, learning rate and initialized radius at the two stages. Finally, make sure the parameter value relatively small, and then, train the sampling data according to the above-mentioned parameter. The results can be viewed in Table2. The main content of the result is the training time and corresponding quantization error.

**Table 2.** Training time and quantization error at two phases

| Index of performance | System layer | | Process layer | | Network layer | |
|---|---|---|---|---|---|---|
| | Phase 1 | Phase 2 | Phase 1 | Phase 2 | Phase 1 | Phase 2 |
| Training time | <1s | 12s | <2s | 10s | <2s | 11s |
| Quantization error | 235.6 | 78.3 | 137.3 | 23.8 | 30.8 | 0.3 |

Table 2 shows the average value of the quantization error. In order to determine the alarm threshold values, we must follow the formula 1 to work out the distance between the data of normal training instances and the corresponding BMU, here it is called BMU Distance. After working out every BMU Distance of every input vector, several distance values can be got from one BMU, and the biggest one is used as the alarm threshold. But not every neuron has such a distance value, and the alarm threshold value is specified as 0 when this happens.

Table 3 shows the alarm threshold values of every neuron of the corresponding system layer MAPs. The most left column of the table shows the X-coordinate value of the MAP, respectively getting the integer between 0 and 11; the most above row shows the Y-coordinate value of the MAP, respectively getting the integer between 0 and 8.

The alarm threshold values of process layer and network layer can be obtained using the same method, and respectively show in Table 4 and Table 5.

**Table 3.** Alarm threshold values on the system layer

| Y / X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 802 | 120 | 59 | 558 | 92 | 23 | 883 | 214 |
| 1 | 49 | 135 | 70 | 91 | 35 | 68 | 26 | 55 |
| 2 | 88 | 145 | 101 | 140 | 0 | 89 | 75 | 215 |
| 3 | 95 | 103 | 112 | 157 | 0 | 91 | 206 | 68 |
| 4 | 682 | 150 | 142 | 0 | 0 | 111 | 83 | 83 |
| 5 | 417 | 168 | 166 | 413 | 95 | 70 | 87 | 72 |
| 6 | 0 | 220 | 0 | 98 | 99 | 145 | 86 | 86 |
| 7 | 396 | 0 | 0 | 0 | 60 | 167 | 90 | 89 |
| 8 | 1034 | 0 | 204 | 0 | 0 | 55 | 96 | 290 |
| 9 | 347 | 1117 | 0 | 551 | 2881 | 263 | 0 | 166 |
| 10 | 0 | 657 | 0 | 0 | 0 | 0 | 32 | 143 |
| 11 | 6868 | 0 | 120 | 538 | 0 | 0 | 481 | 34 |

**Table 4.** Alarm threshold values on the process layer

| Y<br>X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 6.2 | 3.7 | 90.6 | 0 | 13.7 | 38.8 | 41.4 | 332.2 |
| 1 | 0 | 12.1 | 77.0 | 50.1 | 0 | 6.0 | 75.9 | 100.1 |
| 2 | 27.1 | 0 | 60.6 | 20.9 | 6.8 | 19.4 | 140.1 | 7.1 |
| 3 | 123.6 | 0 | 18.6 | 120.5 | 33.3 | 10.4 | 25.5 | 20.1 |
| 4 | 0 | 0 | 44.7 | 0 | 36.2 | 14.7 | 103.3 | 17.2 |
| 5 | 45.8 | 0 | 0 | 11.2 | 136.4 | 0 | 0 | 82.9 |
| 6 | 0 | 227.2 | 0 | 0 | 16.1 | 4.3 | 162.3 | 0 |
| 7 | 40.7 | 125.9 | 95.0 | 39.5 | 0 | 15.7 | 0 | 56.7 |
| 8 | 15.5 | 0 | 56.9 | 12.4 | 58.7 | 12.5 | 0 | 58.7 |
| 9 | 8.1 | 2.0 | 17.7 | 0 | 0 | 36.1 | 44.5 | 75.1 |
| 10 | 0 | 21.8 | 0 | 13.6 | 0 | 94.5 | 72.2 | 38.3 |
| 11 | 60.7 | 28.6 | 111.0 | 7.6 | 17.9 | 56.8 | 126.6 | 559.0 |

**Table 5.** Alarm threshold values on the network layer

| Y<br>X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.99 | 0 | 0 | 0.87 | 0.083 | 0 | 0.39 | 0.24 |
| 1 | 0 | 0.33 | 0 | 0 | 0 | 0.13 | 0 | 5.07 |
| 2 | 4.88 | 0 | 0 | 0 | 0.64 | 0 | 0.21 | 0 |
| 3 | 4.41 | 0.29 | 0.01 | 0 | 0 | 0 | 0 | 0.04 |
| 4 | 5.69 | 0 | 0 | 0.18 | 0.81 | 0 | 1.94 | 0 |
| 5 | 5.34 | 0 | 0.21 | 2.47 | 0 | 0.00 | 0 | 0.00 |
| 6 | 0.21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0.00 | 0 | 0.36 | 0 | 0 | 0 |
| 8 | 1.50 | 0 | 0 | 0 | 0 | 0 | 0 | 5.88 |
| 9 | 0 | 0 | 0.00 | 0 | 0 | 1.50 | 0 | 0 |
| 10 | 0.58 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 1.58 | 0 | 2.90 | 0 | 4.59 | 0 | 0 | 4.23 |

## 4 Experiment result and its analysis

Using the learning algorithm of SOM mentioned above, three Self-Organizing Maps were trained from the training data on the layers of system, process and network. And the alarm thresholds were determined. If considering every BMU as a cluster, then the cluster can be used to determine the normal or abnormal of new instances. This paper uses Distance Function $f_d(s, K)$ to measure the distance between new instance s and cluster K, and uses it to decide whether the new instance abnormal or not. The concrete function is as follows:

$$f_d(s, Normal) = \min\{ f_d(s, K_i) \mid K_i \in C\} \tag{12}$$

$$x_{abnormal}(s) = \begin{cases} 1 & if \quad f_d(s, Normal) \geq t_i \\ 0 & otherwise \end{cases} \tag{13}$$

In the above formulas, C is the set of normal subspace. $t_i$ is used to distinguish the threshold value between the normal class and the abnormal. $f_d$ uses Euclidean distance:

$$f_d(s, K_i) = \| s - w_{K_i} \| \tag{14}$$

In the above formulas, $w_{K_i}$ is the weight vector matrix of the cluster $K_i$ .

In order to achieve the purpose of intruding the important system, the main purpose of the inner intruder was not to intrude the Client, but running suspicious procedures via the Client. On the account of detecting the abnormal of Client in this experiment, we mainly detect the abnormal situation of system while it is running suspicious operation or suspicious procedures. So, anomaly intrusion on the Client is a vital aspect of internal network intrusion detection.

This paper uses NMAP and Hydra to produce the abnormal data of the system, in the meantime, the monitor procedure Monitor was run at the backstage. Through the detection during running these two tools, the abnormal circumstances of different layers are different, which mainly shows that the ratio of the abnormal data in the whole detected data is different. The size of ratio also reflects that the influences on different layers are different, also shows that, in the meantime, the invader is not continuous to make a system working abnormal during intrusion. Therefore, in order to enhance the detection rate, the sampling data should be sampled several times in the process of anomaly-based intrusion detection. Detection rate on different layers during intrusion with the same set of data can be seen from the Table 6.

**Table 6.** Detection rate of different layers during intrusion

| Tools | Detection rate of system layer | Detection rate of process layer | Detection rate of network layer |
|---|---|---|---|
| NMAP | 0% | 36% | 100% |
| HYDRA | 66% | 50% | 86% |

Supposing the time of intrusion is Ta, and the probability of discovering abnormity effectively with sampling one set of data at $t_i$ is p, so, after continuous sampling n sets of data in Ta, the final anomaly-based intrusion detection rate is:

$$P(A) = 1 - (1 - p)^n \tag{15}$$

Under the situation of not changing the threshold value of warning, if you want to make the rate of anomaly-based intrusion detection achieve more than 99%, just ensure $1-(1-p)^n$ >0.99, the same as $(1-p)^n$ <0.01. Fig 5 shows the relation diagram of n and p, when P(A) is more than 99% in the Formula 15. The numerical label in the form is the corresponding coordinate of Y axis. As long as values are taken above the curve, the total detection rate can achieve more than 99%.

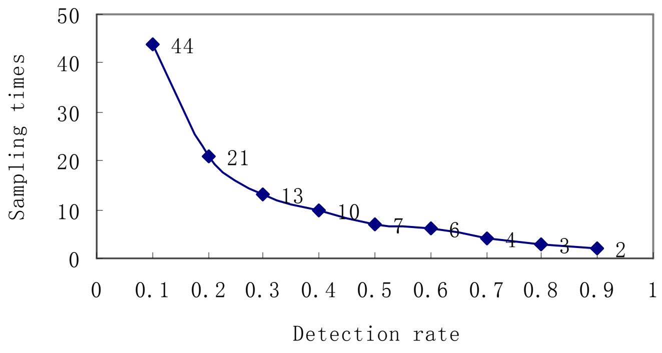

**Fig 5.** Relation diagram of n and p

The detection result above shows that the way of sampling many times makes the detection rate raise to a satisfactory extent, while not needing to change the size of the threshold value of warning, either raise the false positive rate. From Table 6 and Fig 5, we discover that, in order to make the rate of anomaly-based intrusion detection produced by Hydra on the system layer achieve more than 99%, we should sample 6 times during the running of Hydra; in order to make the rate of the anomaly-based intrusion detection produced by NMAP and Hydra on the process layer achieve more than 99%, we should sample 7 times and 13 times respectively during their running; in order to make the rate of the anomaly-based intrusion detection produced by NMAP and Hydra on the network layer achieve more than 99%, we should sample 1 time and 3 times respectively during their running. Result shows that the method of anomaly-based intrusion detection and the choice of the monitor parameter are viable and meaningful.

## 5 Conclusions

In this paper, a novel way of anomaly-based intrusion detection using SOMs is proposed. During the experiment, we use the theory of SOM to train three SOMs on the layers of system, process and network. Although our experiment environment is simpler than the real one, the result shows that it has its reference value for us to build intelligent IDSs. In larger and more complicated real experiment environment, the characteristic value should be selected more extensively, the bound of the time for training the normal sets of data should be a little bit larger, the data should also be a little bit more, and the dimension of the Maps should also be chosen a little bit larger. Thus, training time will consumedly increase, but as long as the training time is restricted in a certain bound, it is acceptable.

**Acknowledgements.** “Computer Software and Theory” of Tianjin Pivot Subject Establishment, and “The Higher Education Institution Science and Technology Development Fund” (No. 2006BA19) of Tianjin Municipal Education Commission, and Wireless Network Research Institute of Tianjin University of Technology supported this work.

## References

1. K.Ord. Outliers in statistical data: V.barnett and t.lewis, 1994, 3rd edition. International Journal of Forecasting, 12(1):175-176, 1996.
2. T.Kohonen.Self-Organizing Maps, Volume 30 of Springer Series in Information Sciences. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).
3. Zanero S,Savaresi S M.Unsupervised Learning Techniques for an Intrusion Detection System[C].Proceedings of 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, 2004.
4. Zhu Dangfeng，Huang Chunhui. Research on Intrusion Detection Technique Based on Rapidly BP Learning Algorithm. Network Security Technology and Application, (8): 36-37,33, 2006.
5. WEI Shengjun，HU Changzhen，JIANG Fei. An Intrusion Detection Method Based on Improved BP Neural Network Algorithm. Computer Engineering, 31(13):154-155,158, 2005.
6. Martin T.Hagan, Howard B.Dcmuch, Mark Beale. Neural Network Design. Beijing: China Machine Press, 2002.
7. T.Kohonen, J.Hynninen,J.kangas, J.laaksonen. SOM_PAK, The Self-Organizing Map Program Package, Version 3.1, 1995.
8. The Self-Organizing Map Program Package. http://www.cis.hut.fi/research/som_pak/