Proceedings of the 6th WSEAS Int. Conf. on NEURAL NETWORKS, Lisbon, Portugal, June 16-18, 2005 (pp176-182)

Detection of TCP Attacks Using SOM with Fast Nearest-Neighbor Search

Jun Zheng Mingzeng Hu
Department of Computer Science, Harbin Institute of Technology
P.O.Box 320, Harbin Institute of Technology, Harbin, 15001, CHINA

Abstract: - A new approach of anomaly intrusion detection (AID) is proposed in this paper. The Self-Organizing Map (SOM) is used to construct the normal usage profiles of network traffic, and in the training phase and detection phase, the Vector Elimination Nearest-Neighbor Search (VENNS) algorithm is designed and implemented. The design procedure optimizes the performance of AID by jointly accounting for accurate usage profile modeling by SOM codebook and fast vector similarity measure using the fast Nearest-Neighbor search. In data processing, according to the characters of TCP attacks, a novel feature extraction approach of TCP flow state is implemented. Using the DARPA Intrusion Detection Evaluation Data Set, we implement the performance evaluation and comparison analysis. It is shown that the performance and efficiency of anomaly intrusion detection are improved greatly: the training time cost can be shortened about by four times and seven times for detection time cost.
Key-Words: - anomaly intrusion detection; Self-Organizing Map; fast Nearest-Neighbor search; normal usage profile; codebook; quantization error

1 Introduction

With the ever fast development of Internet, the network security becomes the main focus in networking fields. In addition to intrusion defensive techniques, such as firewall and encryption, Intrusion Detection System (IDS) is used as an important security-barrier against network-based computer intrusions.

There are two general approaches to intrusion detection: Misuse Intrusion Detection (MID) and Anomaly Intrusion Detection (AID). Similar to virus detection, MID is based on the similar pattern matching to hunt for the signatures extracted from the known attacks. However, AID constructs the historical or long-term usage profile, named the normal usage profile. And then analysis model of AID looks for deviations of the short-term usage profile from the normal usage profiles. The Fig.1 describes the relation and contrast of main logical procedures in MID and AID. Be compared to a defined baseline of normal usage profiles, the deviations can be treated as the suspicious anomalous events related to the intrusions. So AID has the advantage that it can detect new types of intrusions that are currently unknown as deviations from normal usage.

To date, many machine learning and data mining algorithms have been used in Anomaly Intrusion Detection (AID) [1-7] extensively, including the Clustering [1,2], the Support Vector Machine [3,4], the Self-Organizing Map [5,6] and the general Neural Network [4,7] and so on. However, because of the complexities of algorithms, one of main universal shortcomings of these methods is that these methods are not enough efficient to detect by the real time style, which inhibits AID can be implemented in practice further, especially in the high-speed network environment. In order to solve efficiency problems in AID aforementioned, we propose a novel method of AID in this paper based on Self-Organizing Map (SOM) and the fast Nearest-Neighbor (NN) search algorithm-VENNS. The design procedure optimizes the performance of anomaly intrusion detection by jointly accounting for accurate usage profile modeling by SOM codebook and fast vector similarity measure using the fast NN

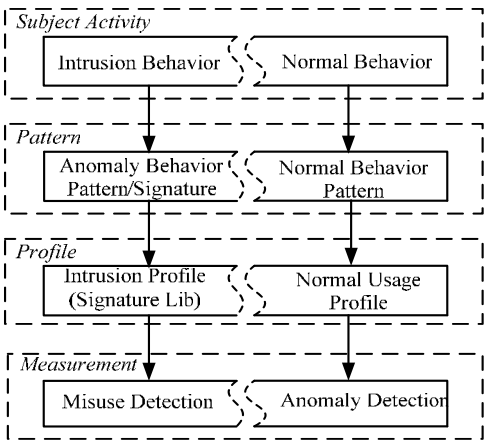


Fig.1. Logic procedures in Intrusion Detection.

search. In data processing, according to the characters of TCP attacks, a novel feature extraction approach of TCP flow state is implemented.

In this paper, we concentrate on the TCP attacks through extracting the TCP header information. Because of complexity and vulnerability, TCP acts as two roles mainly: network attack carrier and network attack target. In the IP traffic of Internet, TCP accounts for 95% or more of the bytes, 85-95% of the packets. [8] Moreover, according to the statistical data from Moore [9], the majority of DoS attack which is main threat to the whole Internet is deployed by using TCP as 90~94%.

The paper is organized as follows: Section 2 explains the basic SOM algorithm and AID framework based on SOM; Section 3 proposes the new fast NN search algorithm -VENNS used in our AID; Section 4 describes the details of experiments over DARPA data set, including the data processing and TCP flow sate quantization. Finally, Section 5 gives conclusions.

2 SOM for Anomaly Intrusion Detection

In this paper, the Self-Organizing Map (SOM) [10] is chosen as anomaly detection model to learn the normal usage behavior for constructing the usage profile.

The reasons that SOM is selected are:

- SOM is one of the unsupervised classification techniques and it is not model-based. We don't need to build the data distribution model. It is important to anomaly detection.
- SOM is a nonlinear projection of high-dimension data to a lower dimensional space, typically the two-dimension plane. It can be effectively utilized to visualize and explore properties of the data. So by SOM, we can observe the distributions of the network traffic usage profiles.
- The topology preserving capability and the automatic generation of probabilities for a dataset can make us to explore the relationships among the multivariate traffic flows in the lower dimensional space straightway.

In this paper, we first need to define the network traffic data, i.e. TCP flow, in form of feature vector:

Definition1: Every TCP Flow is a data point in the n-dimension Euclidian space R^n and R^n is feature space: $\mathrm{TCP.Flow}=\{X \mid X \in R^k\}$, Every TCP Flow is expressed by the form of feature vector: $X=(x_0,x_2,\cdots,x_{k-1})$.

2.1 SOM Algorithm

Definition 2 SOM can be defined as a mapping function from Euclidean space into a certain finite subset C，that is

$Q: R^k \to C$ and $C=\{Y_0,Y_1,.....Y_{m-1} \mid Y_i \in R^k\}$is codebook of SOM. m is the dimension of codebook. The map function satisfies $Q(X \mid X \in R^k\}=Y_p$ and $Y_p=(y_{p,0},y_{p,1},\cdots;y_{p,k-1})$ is called the codeword or the weight vector.

The following are the main training steps involved in SOM to get the codebook:

Input: input vector: $X=(x_0,x_1,\cdots,x_{k-1})$

Output: codebook $C=\{Y_0,Y_1,.....Y_{m-1} \mid Y_i \in R^k\}$

Step1 Initialize codeword with random values: $Y_j(0)$

Step2 To compute the distance between the input vector X_i and the codeword $Y_j(t)$, designate the winner neuron node j^* with the smallest distance. j^* is also called the ***Best Matching Unit (BMU)***.

$$j^* = \arg\min_{1\le j\le m} \|X_i - Y_j(t)\| \qquad (1)$$

The Euclidean distance is chosen as *Quantization Errors (QEs)*:

$$D=\|X_i-Y_j(t)\|=[\sum_{k=1}^{n}(x_{ik}-y_{jk}(t))^2]^{1/2} \quad (2)$$

Step3 To update the winner vectors of the winner node and its neighborhood:

$$y_{jk}(t+1)=y_{jk}(t)+a(t)[x_{ik}-y_{jk}(t)], j\in N(t) \quad (3)$$

$N(t)$:Non-increasing neighborhood function; $a(t)$: Learning rate function, $0<a(t)<1$.

Step4 To repeat Step2 and Step3 until SOM stabilizes

Fig.2. Flowchart of SOM

2.2 Detection Phase

Using the SOM codebook, we can get the normal network traffic profiles. Further in succession, in order to find TCP network intrusions that exhibit as deviations from normal usage profiles, we measure the similarity between current usage behaviors and usage profiles using the ***Quantization Errors*** (*QEs*). The intrusion detection can be treated as the process

of the Nearest-Neighbors search of input TCP Flow feature vectors just described in definition 3.

Definition 3 The intrusion detection can be defined as the following: Given a collection of *k*-dimensional points, *C*, codebook of SOM, a query point *p*, find a codeword *q*, the closest to *p* than any other codewords in *C*, $\{q \mid q \in C \cdot and \cdot \forall r \in C, \|r-p\| > \|q-p\|\}$, $\|q-p\| = QE$. Given the distance deviation threshold e , *p* is an intrusion data point when and only when $\|q-p\| \geq e$.

3 Fast NN Search Algorithm

As aforementioned in Section 2, the computing complexity of system is mainly concentrated on the Euclidian distance computing of the similarity measure between the *k*-dimension feature vectors in both training phrase and detection phrase. The computational cost of measuring squared Euclidean distance is very high because a prohibitive number of mathematical operations are required especially when the input feature vector number and the dimension are large.

In our method, we implement the faster NN search algorithm to accelerate similarity measures in order to implement the high efficient AID. The elimination-based fast search exploits some properties of the distance measure used in the nearest neighbor definition. Here, fast NN search algorithm -"Vector Elimination NN Search" is designed and implemented, where the codewords which cannot be nearer to the given test vector than the current nearest-neighbor are eliminated without incurring the cost of a distance computation.

To suppose input feature vector is $X=(x_0,x_1,\cdots,x_{k-1})$ and the codeword is $Y=(y_0,y_1,\cdots,y_{k-1})$. For *X* and *Y*:

$$S_x = \sum_{l=1}^{k} x_l \ , \ S_j = \sum_{l=1}^{k} y_{j,l} \qquad (4)$$

It is easy to prove that:

$$D(X,Y_j)^2 = \sum_{l=1}^{k}(x_l - y_{jl})^2 \geq k \bullet \left[\sum_{l=1}^{k}\frac{(x_l - y_{jl})}{k}\right]^2 = \frac{(S_x - S_j)^2}{k} \qquad (5)$$

To take the current minimum Euclidean distance *D* as $D_{\min}$ in equation (2), we can get the elimination rule. If Y satisfies inequation (6), then the Euclidean distance between codeword Y and input feature vector can be avoided so that computing cost can be reduced.

$$(S_x - S_j)^2 \geq k \bullet D_{min}^2 \qquad (6)$$

So, the fast NN search algorithm we implement in our AID system is described as following.

Input: codeword $Y=(y_0,y_2,\cdots,y_{k-1})$, input vector: $X=(x_0,x_2,\cdots,x_{k-1})$

Output: Y_j that is the nearest to the input vector *X* and their distance D_{min};

Step1 Compute $S_j = \sum_{l=1}^{k} y_{j,l}$ and $S_x = \sum_{l=1}^{k} x_l$, array $S_j = \sum_{l=1}^{k} y_{j,l}$ in ascending order and store them in the temporary codebook *C'*;

Step2 According to input *X*, to search Y_p by the binary search: $p = j^* = \arg\min_{1 \leq j \leq m} |S_x - S_j|$, and then set $D(X,Y_p)$ as the minimal distance $D_{min} = D(X,Y_p)$;

Step3 Search codeword $Y_{p\pm i}$ ($i=1,2,\cdots,m-p$) around Y_p (Fig.4.), if $Y_{p\pm i}$ satisfies inequation (6) then it will eliminate the Euclidean distance computing between $Y_{p\pm i}$ and *X*. For else, to compute the Euclidean distance *D* between $Y_{p\pm i}$ and *X*, and then to compute D_{min} anew: $D_{min} = \min\{D_{min}, D\}$. Continue searching next codeword;

Step4 Return D_{min} and its related codeword Y_j ($D_{min} = \|X - Y_j\|$).

Fig.3. Flowchart of VENNS

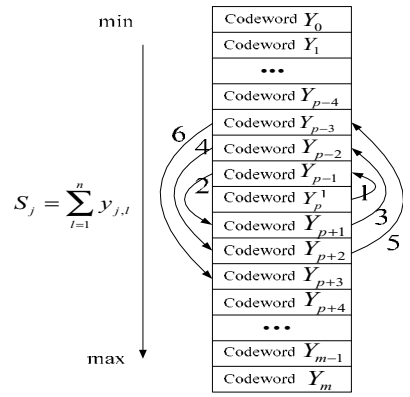


Fig.4. The binary search in VENNS

4 Evaluation Methods and Results

4.1 Data Processing

It is necessary to do data processing to extract the feature attributes from TCP flows, and then, the date normalization will be processed to project whole feature attributes to a unit range no matter the continue attributes or quantitative attributes. In the paper, data processing is focused on TCP traffic. The program of data process in the paper is based on our amend edition of *Libnids* [11]. The aim of feature extraction is to achieve the maximum difference degree between usual usage behaviors and intrusion behaviors. A feature vector of the TCP traffic flow is shown in Table 1. The FlowState attribute is the most important among all feature attributes and we introduce the new method to present it in the following section. In order to get the correlation and statistical information in a certain time interval, the time window is used in the paper (120 seconds).

Table1. Feature attributes of TCP Flow feature vector

Feature Attribute	Description
SrcIP	source IP address
DestIP	destination IP address
SrcPort	source port
DestPort	destination port
PktSize	average packet size in one TCP Flow
SrcBytes	the number of bytes from source
DestBytes	the number of bytes from destination
FlowState	TCP Flow closed state
Fre_SrcIP	frequency of a certain source IP in time-window
Fre_DestIP	frequency of a certain destination IP in time-window

4.2 Quantization of TCP Flow state

There are nine flags involved in the connection establishment of TCP 3-way handshake protocol and the connection close of TCP 4-way handshake protocol (Fig.5.). We devised a 9-bit number to identify the connection state. The flag will be set to 1 if that corresponding flag is observed during the establishment-close process. Otherwise, it will set to 0. The decimal function $Sum(Flag_0, Flag_{1,\ldots\ldots,} Flag_8)$ with the non-repeating value is used to quantitate the whole connection process:

$$Sum(Flag_0, Flag_{1,\ldots\ldots,} Flag_8) = \sum_{i=0}^{8} Flag_i \bullet 2^i \qquad (4)$$

$$= RST_{active} \bullet 2^0 + RST_{passive} \bullet 2^1 + SYN \bullet 2^2 + ACK_{syn} \bullet 2^3 + ACK \bullet 2^4 +$$

$$+ FIN \bullet 2^5 + ACK_{fin} 2^6 + FIN' \bullet 2^7 + ACK'_{fin} \bullet 2^8$$

According to Eq.(10), the TCP connection with the normal close can be described as follows:

$$Sum = (111111100)_2 = (508)_{10}$$

Consider the situation that, with TCP protocol, the certain flag repeat in a TCP Flow, we substituted the 32-bit number (4 bytes) for the 9-bit number, as in Fig. 5. So, if the occurrence time of one certain flag is less than 15, the sum will not be repeated. The number of occurrence time will take value of 15 if it exceeds 15. ($RST_{passive}$/ RST_{active} occurrence time is less than 3).

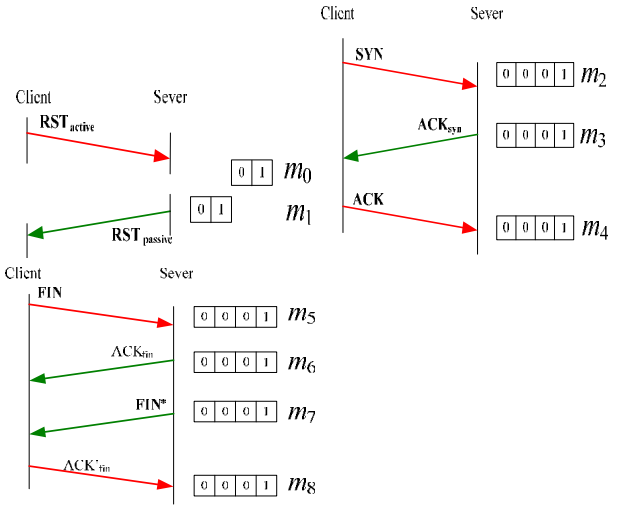


Fig.5. Quantization of TCP Flow state

Most attacks can result in the abnormal state of connection according to the TCP protocol. Generally, 14 connection-closing states are summarized in [12]. We find that method is clumsy relatively in data preprocess program. What is more important, these 14 states cannot include all the complicated instances of TCP connection state. By state quantization, every state of connection can be easy mapped to an int data range affording the state synopsis attribute directly leading to the whole improvement of feature vector.

4.3 Experiments and Results

4.3.1 Data Set

The Intrusion Detection Evaluation Data Set of 1999 DARPA [13, 14] has been widely used in the community of intrusion detection. The data set of DARPA 1999 includes data of five weeks. The data sets of Week1, Week2 and Week3 are train data set. But Week1 and Week3 data sets have entirety normal network traffic logs attack free. Week2 data set has labeled attack data in order to train the supervised learning classifiers based anomaly intrusion detection, so the Week2 is not suit in our experiments. The data sets of Week4 and Week5 data sets are the test data sets including attacks for detection. Table 3 gives the statistic information about the DARPA 1999 data set after the data processing that we extract

the TCP Flow feature vectors from the original *outside.tcpdump* log files in the data set.

We examined only the *outside.tcpdump* logs to reduce the complexity of our experiments. The codebook is designed by the data set attack free in week1 and week3. Consider the fact that the target of this paper is limited in the scope of TCP Flows and not general, we test for 35 instance TCP attacks in the Week 4 and Week 5 presented in Table 2 [13]. The experiment computer is Dawning Sever (Linux7.2/PIII800/RAM1G).

Table2. Attacks in experiments

Attack Name	Num.
Back (DoS)	3
Selfping (Probing)	3
Portsweep (Probing)	12
Satan (Probing)	2
Mscan (Probing)	1
Ntinfoscan (Probing)	3
Apache2 (DoS)	3
Queso (Probing)	3
Neptune/SYN-flood (DoS)	3
Processtable (DoS)	2
Total	35

4.3.2 Parameter Selection in SOM

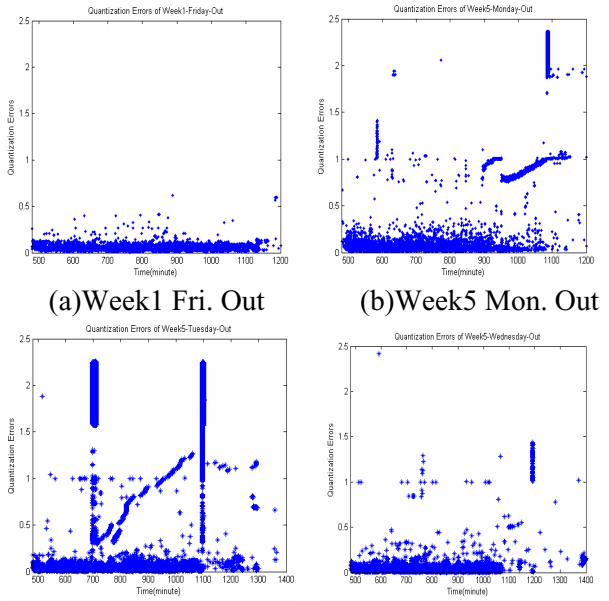
The parameters of SOM used to train codebooks are selected by experience in the Table5. Particularly, two dimension sizes of codebook (30×30 and 40×40) are selected to creating the network traffic usage profile in order to do some comparisons.

Table 3 Parameters in SOM

Parameter	Value
Topology	Hexa
Neighborhood function	Bubble
Dimension	30×30 40×40
Learning rate function	Linear
Training rate of first phase	0.5
Radius in first phase	20
Training length of first phase	50000
Training length of second phase	500000
Training rate of second phase	0.02
Radius in second phase	10

4.3.3 Experiment Results

As we use ***QE*** to evaluate the on-detecting network traffic feature vectors, Fig.6 (a-d) presents the ***QE*** distributions (30×30 codebook) in outside traffic of the training data of week1 Fri. and the testing data of some week5 days. By *QEs*, these figures can give the better explain to understand the data distribution in the training or testing phrase. From the Fig.6 (a), we can observe that *QEs* of the training data are well regulated and don't change with the large deviations. However, the strong contrast in Fig.6 (b) (c) (d) of the testing data including attacks are the sharp variations of *QEs* due to the high values of attack traffic QEs. Markedly, on 18:04, Neptune attack (a sort of SYN-flood DoS attack) can be viewed with QEs values ranging from 1.8038 to 2.3063 in Fig.6 (b).



(a)Week1 Fri. Out (b)Week5 Mon. Out

(c)Week5 Tue. Out (d)Week5 Wed. Out

Fig.6. *QE* distributions of training data and testing data.

In Table 4 the whole detail detection rates (DRs) and false positive rates (FPRs) are presented according the different thresholds e *(QEs)*. By varying detection thresholds, the Receiver Operating Characteristic (ROC) curves of DR and FPR are illustrated in Fig.7. for fair comparison of two different sizes of codebook. Fig.7. shows that the DR is trend to 100% fast by changing the threshold smaller; but the FPR also rises too. In the ROC curves, the eight points in inflexion areas suggest the ideal situations with the higher DRs and the lower PFRs. Table 4 just presents such situations where two codebooks achieve the better performance with DR-FPR- e (98.034%,1.03%,0.76) and (96.623%,1.12%,0.80). So, adjusting detection thresholds is necessary to for the certain sizes of codebook in order to achieve the tradeoff outcomes with some parameters having been selected.

Table 4. Thresholds for codebook 40×40 and 30×30

Threshold ε (QEs)	0.92	0.88	0.84	0.78	0.76	0.72	0.68	0.66
40×40 DR (%)	45.666	51.230	84.123	96.123	98.034	98.631	99.232	100
40×40 FPR (%)	0.61	0.63	0.78	0.91	1.03	2.04	3.21	10.32
Threshold ε (QEs)	0.96	0.92	0.88	0.82	0.80	0.76	0.72	0.70
30×30 DR (%)	44.126	46.323	81.113	94.123	96.623	97.872	98.001	100
30×30 FPR (%)	0.81	0.83	0.89	1.09	1.12	2.42	3.74	18.87

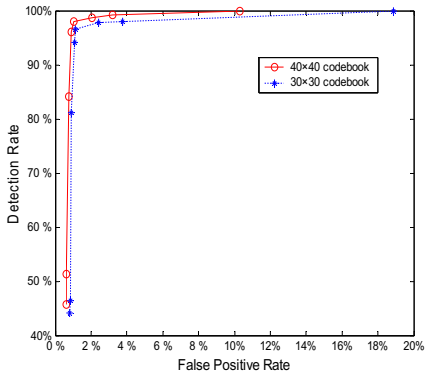
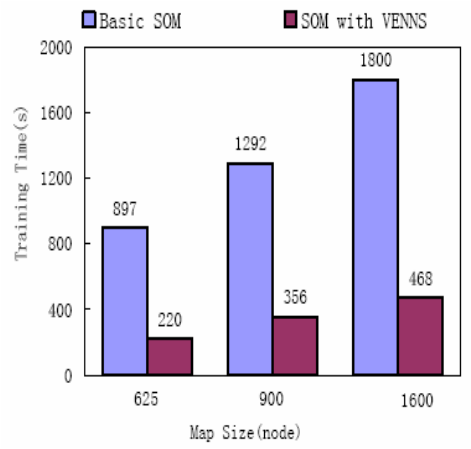
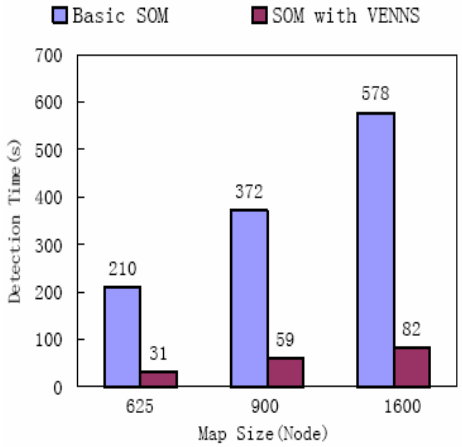


Fig.7. ROC curves of DR and FPR for codebook of 40×40 and 30×30

In addition to the codebooks of 40×40 and 30×30, we use additional the codebook 25×25 to test efficiency differences between two different approaches: Basic SOM and Fast SOM with our VENNS implemented in our AID system. Fig.8. shows the different system performances in the codebook training time and detection time according to various codebook sizes. The comparison outcome is very impressive because of computing time cost reducing greatly. Recall that the training data sets have 450,019 TCP Flows and the test data sets have 621,111 TCP Flows, the results of detection and training time cost of fast SOM with VENNS algorithm are strongly impressed. The detection rate reaches about 7575 units every second which outperforms greatly the SVM classifier detection rate 3445 units every second in [3] even omitting the effect of different data dimensions. The comparison between our method and SVM validates the higher performance of fast SOM used in our AID framework. It is deserved to point that [5] used basic SOM to implement the network based intrusion detection which is similar with our method in some meaning. Fig.8. can explain the advantage of our method over [5].



(a) Training



(b) Detection

Fig.8. Time cost comparisons between AID based the basic SOM and AID based the SOM with VENNS.

5 Conclusion

The intrusion detection researches are still focus in the network and computer security fields. The paper proposed anomaly intrusion detection framework using the SOM. Further more, this paper has focused on enhancing the overall performances of AID to achieve more efficiency and usability for the high speed network, which is usually neglected by some researches. For reducing the computing cost, the fast NN search-VENNS algorithm to accelerate similarity measures has been implemented to satisfy the need of real-time detection style. The evaluation experiments have clearly confirmed that AID framework in the paper could achieve the high performance.

References:

[1] Sang Hyun Oh, Won Suk Lee: An Anomaly Intrusion Detection Method by Clustering Normal User Behavior. *Computers and Security*, 22(7), 2003, pp. 156-169.

[2] L. Portnoy, E Eskin, S J Stolfo. Intrusion Detection with Unlabeled Data Using Clustering, In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001), Philadelphia, PA, 2001.

[3] S. Mukkamala, A. H. Sung, A. Abraham.: Intrusion Detection Using an Ensemble of Intelligent Paradigms. *Computer Applications* 28(1), 2005, pp. 167-182.

[4] Mukkamala S, Janowski G, Sung AH. Intrusion detection Using Neural Networks and Support Vector Machines. Proceedings of Hybrid Information Systems Advances in Soft Computing, Springer; 2001, pp.121–38.

[5] Depren, M.O.,Topallar, M., Anarim, E.; Ciliz, K.: Network-based anomaly intrusion detection system using SOMs. Signal Proceedings of the IEEE 12th Communications Applications, Turkish, 2004, pp 76-79.

[6] Hoglund A. J., Hatonen K., Sorvan A. S.: A computer Host Based User Anomaly Detection System Using the Self Organizing Map. Proceedings of IEEE World Congress on computational Intelligence, 2002, pp 411-416.

[7] J. M. Bonifaco, E. S. Moreira: An Adaptive Intrusion Detection System Using Neural Network, Research Report, UNESP, Brazil, 1997.

[8] G. J. Miller K. Thompson and R. Wilder.: Wide-area Internet traffic patterns and characteristics. IEEE Transactions on Network, pp 10--23, 1997.

[9] D. Moore, G. Voelker, and S. Savage, :Inferring Internet Denial-of-Service Activity, in Usenix Security Symposium, Washington, D.C., Aug 2001.

[10] Kohonen, T.: *Self-Organization Maps*, Springer-Verlag, Berlin, 1997.

[11] http://libnids.sourceforge.net/

[12] L Wenke, S J Stolfo, K W Mok: A data mining framework for building intrusion detection models. Proc. of the 1999 IEEE Symposium on Security and Privacy, Oakland, 1999, pp.296–304.

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

[14] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, Kumar Das. :The 1999 DARPA Off-Line Intrusion Detection Evaluation, *Computer Networks*, 34 (4), 2000, pp. 579-595.