Online Training of SVMs for Real-time Intrusion Detection

Zonghua Zhang and Hong Shen
Graduate School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Tatsunokuchi, Ishikawa, 923-1292, Japan
E-mail:{zonghua, shen}@jaist.ac.jp

Abstract

To break the strong assumption that most of the training data for intrusion detectors are readily available with high quality, conventional SVM, Robust SVM and one-class SVM are modified respectively in virtue of the idea from Online Support Vector Machine (OSVM) in this paper, and their performances are compared with that of the original algorithms. Preliminary experiments with 1998 DARPA BSM data set indicate that the modified SVMs can be trained online and the results outperform the original ones with less support vectors(SVs) and training time without decreasing detection accuracy. Both of these achievements benefit an effective online intrusion detection system significantly.

1. Introduction

As computer network is prevailing on earth and playing increasingly vital roles in modern society, its security is being paid more and more attention because a major attack can significantly reduce the capability of information systems, and any exploitable weakness of networks that can be used by hackers and criminals would cause great loss to people. As the backup measures of intrusion preventions such as user authentication, authorization and encryption etc., intrusion detection techniques become ever more concerned and some achievements have been applied widely with narrowly satisfied performance.

Traditionally, the criteria to evaluate the efficiency of an IDS is the tradeoff between the ability to detect novel attacks and the ability to minimize the false positive rate. Over the past decades, many intrusion detection techniques drawn from statistical analysis [14], data mining [6], support vectors machines(SVMs)[4, 10] etc. have been proposed and applied to various observable subjects. Most of these available approaches focus on improving detection accuracy and restraining false alarms, and they can achieve satisfactory results in terms of such criteria given enough time. However, in practice, intrusion detection is a real-time task, that is, intrusions should be detected as soon as possible or at least before the happening of the risks. In addition, there is usually an initial training period for an intrusion detector to learn the observable subject's behavior, and most existing methods are based on the assumption that labelled training data are readily available with high quality. In fact, intrusion detectors should undergo frequent retraining to incorporate new examples periodically into the training data for classifying novel attacks and changed normal behavior. Therefore, running time and training time should also be considered in addition to detection accuracy and false alarms to design an effective IDS.

This paper focuses on improving the performance of existing intrusion detection methods, i.e., conventional SVM [1], Robust SVM [4] and One-class SVM [10] respectively in terms of training time and running time based on the idea extended from Online SVM [5]. In our improved approaches, observable subjects, i.e. system calls of processes, are trained and tested one by one in a sequence rather in a batch ,and the classification are undertaken and completed before next object is received, which somehow is similar to online training.

The rest of paper is organized as follows. Section 2 reviews some related work. Section 3 introduces three different SVMs together with their modified algorithms. In section 4, we present the experiment and analyze the results, and section 5 concludes the work.

2. Related Work

Generally, observable subjects, models and techniques are three elements central to intrusion detection [6]. The idea of building program profiles with short sequences of system calls executed by running privileged programs for intrusion detection was proposed by Forrest et al [3], which is based on the assumption that sequences of system calls in an intrusion are noticeably different from that of normal. Warrender et al [13] ever presented that the choice of data

Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)
0-7695-2051-0/04 $ 20.00 © 2004 IEEE

stream(short sequences of system calls) is the more important decision than the particular method of analysis, but subsequent studies did not support this conclusion adequately. Ye et al [14] investigated the frequency property and the ordering property of computer audit data, and pointed that intrusion detection techniques based on the ordering property can hardly provide a feasible solution that produces good performance with low computational overhead due to the scalability problem of complex data models, especially when the intrusive audit data are mixed with white noises of normal audit data, whereas the frequency property can provide a viable tradeoff between computational complexity and intrusion detection performance.

Liao et al [7] applied K-Nearest Neighbor(KNN) classifier to label program behavior as normal or intrusive, specifically, each system call in the process was treated as a word and the collection of system calls over each program execution as document, and thus the system call frequencies were used as subjects to represent program behavior. Based on the assumption that number of normal instances is significantly larger than that of anomalies, and Nguyen [10] employed one-class SVM [11] to identify "outliers" amongst positive examples(normal behaviors) and treated them as negative examples(abnormal behaviors), but the unchanged patterns which can not reflex *concept drift* limit its application. Additionally, based on the assumption that an attack occurs during the training process, and the undesired intrusive behavior could be regarded as the normal one and thus undermine anomaly detector's accuracy [8], Hu et al [4] used Robust SVM [12], which can solve the over-fitting problem effectively introduced by the noise in the training data set, to anomaly detection over noisy audit data.

3. Methodology

In this section, we would give a brief introduction about conventional SVM, RSVM and one-class SVM, and then modify them using a general algorithm drawn from Online SVM [5], prove of these modified methods are also given.

3.1. Three SVMs on Different Assumptions

Given a training sample: $D_l = \{x_i, y_i\}_{i=1}^{l}$, x_i is the ith input vector, $x_i \in R^n$, $y_i \in [+1, -1]$, l is the total number of input vectors and n is the dimension of the input space. Suppose the relation between x and y is $y = sgn(f(x)+\varepsilon)$, where $sgn(x) = 1$, if $x \geq 0$ and $sgn(x) = -1$, if $x < 0$, the task uncovering function f is called classification. SVC is a maximization(minimization) algorithm used to identify a set of linear separable hyperplanes in the feature space whose formula like $f(x) = \langle w, x\rangle + b$, and $2/\|w\|$ can be regarded as a canonical representation of the separating hyperplane. Maximization of the margin between the positive examples and negative examples can be transferred to the following problem [1]:

$$\begin{cases} min & \frac{1}{2}\|w\|^2 \\ s.t. & y_i(\langle w, x_i\rangle + b) \geq 1 \quad \forall i, \end{cases} \tag{1}$$

The general dual objective function by applying the Lagrangian multiplier can be rewritten in a matrix form as follows[5]:

$$L_D = \frac{1}{2}\beta^T K\beta - \langle c, \beta\rangle, \tag{2}$$

where c is an $l \times l$ vector, $\beta = \{\beta_1, ..., \beta_l\}$ and $K = \{K_{ij}\}$, $K_{ij} = y_i y_j K(x_i, x_j)$, while $K(x_i, x_j)$ is called kernel function, which can be selected such formulas as $K(x_i, x_j) = \langle x_i, x_j\rangle^d$ or $K(x_i, x_j) = e^{\|x_i - x_j\|/\sigma}$. The feasible solution of Eq.(2) should satisfy the KTT[1] conditions as follows:

$$\begin{cases} \beta_i = 0 \Leftrightarrow y_i f_i > 1, \\ 0 < \beta_i \leq C \Leftrightarrow y_i f_i \leq 1, \end{cases} \tag{3}$$

The hyperplane $f(x)$ can be expanded from the kernel as follows:

$$f(x) = sgn\left(\sum_{i \in SV} \beta_i y_i K(x, x_i) + b\right). \tag{4}$$

In order to solve over-fitting problem of soft margin SVM due to noisy training data, Robust SVM [12] only minimize the margin of the weight w instead of minimizing the margin and the sum of misclassification errors. The objective function can be written as following:

$$\begin{cases} min & L_D = \frac{1}{2}\beta^T K\beta - \langle c, \theta\rangle \\ s.t. & \sum_{i=1}^{l} \beta_i y_i = 0, \beta_i \geq 0, \forall i, \end{cases} \tag{5}$$

where $\theta = \langle \gamma, \beta\rangle$, $\gamma = \{\gamma_1, ...\gamma_l\}$, and $\gamma_i = 1 - \lambda D^2(x_i, x^*_{y_i})$, $\lambda \geq 0$ is a pre-determined regularization parameter measuring the influence of averaged information(distance to the class center), and $D^2(x_i, x^*_{y_i})$ represents the normalized distance between data point x_i and the center of the corresponding classes, $(x^*_{y_i}, y_i \in \{+1, -1\})$, in the feature space. The slack variable $\lambda D^2(x_i, x^*_{y_i})$ can be justified by considering it as part of the margin.

Another adapted algorithm, called one-class SVM algorithm, identifies "outliers" amongst positive examples and use them as negative examples. After mapping between input data space X and high-dimensional feature space H via a kernel, origin is treated as the only member of the second class. Then "relaxation parameters" is used to separate the point of the first class from the origin. To compare with above algorithms, we can write the objective function as:

$$\begin{cases} min & L_D = \frac{1}{2}\beta^T K\beta \\ s.t. & 0 \leq \beta_i \leq \frac{1}{vl}, \sum_{i}^{l} \beta_i = 1, \end{cases} \tag{6}$$

where $v \in (0,1)$ is a parameter that controls the trade off between maximizing the margin from the origin and containing most of the data in the region generated by the hyperplane. The general decision function with kernel expansion is:

$$f(x) = sgn\left(\sum_{i=1}^{l} \beta_i k(x_i, x) - \rho \right). \quad (7)$$

If α_i meets the subject conditions, ρ can be recovered as:

$$\rho = \sum_{j=1}^{l} \beta_j k(x_j, x_i). \quad (8)$$

3.2. Modified SVMs

In the original SVMs, training data are supplied and computed in batch by solving the quadratic programming problems $Eq.(2,5,6)$, therefore, it is time consuming to classify a large data set and can not meet the demands of online application, especially for intrusion detection, which needs periodically retraining because of the update of the training data. With this objective in mind, we consider the case that training data can not be get at one time or supplied in sequence.

For supervised SVMs, e.g., conventional SVM and Robust SVM, one example can be given for each class, the hyperplane with a maximum margin for these two examples can be found by solving the objective function $Eq.(2,5)$. When a new example available, corresponding to the KKT conditions, two cases need to be considered, i.e, if it can be classified by the current optimal boundary correctly, then the example is not a support vector, otherwise, a new hyperplane should be determined so that it can classify three examples. The new hyperplane can be found by minimizing objective function with the SVs obtained from the current hyperplane and the new example. At the kth step, the set of SVs can be denoted as SV_k, and the existed examples are $\{Sx_i^k, Sy_i^k\}_{i=1}^{|SV_k|}$ respectively. The corresponding hyperplane is(conventional SVM):

$$f_k(x) = sgn\left(\sum_{i=1}^{|SV_i|} \beta_i^k Sy_i^k K(x, Sx_i^k) + b_k \right). \quad (9)$$

Once the hyperplane is updated, the KKT conditions are checked for all k examples, and the example which violate the KKT conditions should be feed back to algorithm as new examples. Borrowed the idea from online SVM [5], we rewrite a general algorithm for three SVMs described above to improve the performance of their training phase:

Algorithm of online training for SVMs

void OnlineTraining( )
{
  set $W_1 = \{x_k, y_k\}$, for $k = 1, 2$, and $|E_2| = 0$,
  // or $k = n, n+1$ and $|E_{n+1}| = 0$
  Minimize $Eq.(4,7,8)$ with W_1 to obtain an optimal boundary f_1.
  for (int $k = 3$; $k <= l$; $k++$) {
  // or $k = [n+3, ..., n+l]$
    Obtain a new element $S_k = \{x_k, y_k\}$;
    if (S_k can be distinguished by f_{k-1}) {
      Add S_k to the corresponding class;
    }
    else {
      $W_k = \{Sx_i^{k-1}, Sy_i^{k-1}\}_{i=1}^{|SV_{k-1}|} \cup S_k$;
      Minimize $Eq.(4,7,8)$ with W_k to obtain a new optimal boundary f_k;
    }
    if ($|E_k| = |\{x_i, y_i | y_i f_k(x_i)$ violates the KKT conditions $\}_{i=1}^{k}| > 0$){
      E_k be input next step as new elements;
    }
  }
  while ($|E_l| > 0$) {
    Minimize $Eq.(4,7,8)$ with $W_l = SV_l \cup E_l$ to obtain an optimal boundary f_l;
  }
}

Actually, the process can start at any kth step, thus some typical intrusions can be kept meanwhile learn to detect novel attacks. However, because of the computational overhead, the number of SVs, n, for the existing examples should not be too large, otherwise, this algorithm can hardly perform better than conventional training methods. Because of the unsupervised nature, One-class SVM takes original point instead of intrusive point and another normal behavior at its initial training stage for subsequent classification.

3.3. Convergence of Modified SVMs

By comparing with the decomposition algorithm(DA), the convergence of the modified standard SVM has been proved in Ref.[5]. We would proof the convergence of modified robust SVM and modified one-class SVM in this section. Instead of solving the large quadratic programming problem at once, DA take small quadratic programming sub-problems as its objective, and thus the iteration solution of the sub-problem bring the solution closer to the optimal solution. The training set is decomposed into working subset B and correcting subset$N$. At each step, m elements exchange between the subset B and N. With the elements exchanged, sub-problem involving the new working set is solved. The exchange between B and N repeat until no example violates the KKT conditions. The convergence of the DA for standard SVC and robust SVC have been proved in

Refs.[2] and [**?**] respectively. Similarly, the dual objective function $Eq.(6)$ of one-class SVM can be rewritten involving the working and correcting sets as follows:

$$\begin{cases} min & \frac{1}{2}[\beta_B^T K_{BB}\beta_B + \beta_B^T K_{BN}\beta_N + \beta_N^T K_{NB}\beta_B \\ & +\beta_N^T K_{NN}\beta_N] \\ s.t. & 0 \le \beta_B \le \frac{1}{vl}, \beta_B + \beta_N = 1. \end{cases} \tag{10}$$

where

$$\beta = \begin{pmatrix} \beta_B \\ \beta_N \end{pmatrix}, y = \begin{pmatrix} y_B \\ y_N \end{pmatrix}, K = \begin{pmatrix} K_{BB} & K_{BN} \\ K_{NB} & K_{NN} \end{pmatrix}.$$

Based on several propositions of DAs, the following corollary given by Lau et al [5] shows that keeping the SVs in the working set will not affect the optimal solution. Here we want to proof that it also holds for both robust SVM and one-class SVM.

Corollary *Moving an element which is not an SV from B to N leaves the cost function unchanged and the solution is feasible in the sub-problem.*

Proof. Suppose $B^{'} = B - \{m\}, N^{'} = N \cup \{m\}, \{m\} \in B - SV$, where"$-$"denotes set substraction, m represents an element which is not SV.
(1) For robust SVM, we have

$$\begin{aligned} & L_D(\beta_{B'}, \beta_{N'}) \\ &= \tfrac{1}{2}[\beta_{B'}^T K_{B'B'}\beta_{B'} + \beta_{B'}^T K_{B'N'}\beta_{N'} + \beta_{N'}^T K_{N'B'}\beta_{B'} \\ &\ +\beta_{N'}^T K_{N'N'}\beta_{N'}] - \gamma(\beta_{B'}^T + \beta_{N'}^T) \\ &= \tfrac{1}{2}[\beta_{B'}^T K_{B'B'}\beta_{B'} + 2\beta_{B'}^T K_{B'N'}\beta_{N'} + \beta_{N'}^T K_{N'N'}\beta_{N'}] \\ &\ -\gamma(\beta_{B'}^T + \beta_{N'}^T) \end{aligned}$$

The optimization problem can be formulated as follows:

$$\begin{cases} min & L_D(\beta_{B'}, \beta_{N'}) \\ s.t. & < y_{B'}, \beta_{B'} > + < y_{N'}, \beta_{N'} >= 0, \\ & -\beta_{B'} \le 0. \end{cases} \tag{11}$$

(2) Similarly, for one class SVM, we have

$$\begin{aligned} & L_D(\beta_{B'}, \beta_{N'}) \\ &= \tfrac{1}{2}[\beta_{B'}^T K_{B'B'}\beta_{B'} + \beta_{B'}^T K_{B'N'}\beta_{N'} + \beta_{N'}^T K_{N'B'}\beta_{B'} \\ &\ +\beta_{N'}^T K_{N'N'}\beta_{N'}] \\ &= \tfrac{1}{2}[\beta_{B'}^T K_{B'B'}\beta_{B'} + 2\beta_{B'}^T K_{B'N'}\beta_{N'} + \beta_{N'}^T K_{N'N'}\beta_{N'}] \end{aligned}$$

The optimization problem can be formulated as follows:

$$\begin{cases} min & L_D(\beta_{B'}, \beta_{N'}) \\ s.t. & 0 \le \beta_{B'} \le \frac{1}{vl}, \\ & \beta_{B'} + \beta_{N'} = 1. \end{cases} \tag{12}$$

We know that the objective function $L_D(\beta_{B'}, \beta_{N'}) = L_D(\beta_B, \beta_N)$, and we note that $N^{'}$ does not contain any SV. Hence, $\beta_{N'} = \mathbf{0}$, for its elements are not SVs, and thus $L_D(\beta_B, \mathbf{0}) = L_D(\beta_{B'}, \mathbf{0})$, where $\mathbf{0}$ is a column vector whose all elements are 0. In addition, we have $\beta_B^T y_B = \beta_{B'}^T y_{B'} = 0$ and the bound constraints of robust SVM are unaffected, and obviously, the bound constraints of one class SVM are unaffected too. Therefore, both the sub-problem of robust SVM and one-class SVM have the same solution with their corresponding proposed algorithms.

4. Experiments

In order to compare our method with the former achievements, we also select the 1998 DARPA data [9] as our experiment data, which consisted of seven weeks of training data and two weeks of testing data.

4.1. Preprocessing of DARPA Data

Basic Security Module(BSM) audit data collected from a victim Solaris machine is used as experiment data here. According to the idea about processing data presented by Liao, et al. [7], system calls that generated by the execution of processes can be regarded as words, and thus the processes be considered as documents. A text categorization problem was formulated here. Any attacks or anomalies during the execution of processes should be detected and thus guarantee the real time intrusion detection. Two text weighting techniques, i.e., frequency weighting and *tf-idf* weighting, can be used to transform the process into a vector. According to the analysis of these two methods in [7], we adopted frequency weighting as our weighting method.

There are 5 out of 35(seven-week training)simulation days free of attacks. Four days were picked arbitrarily out of these five days for training, and the left one for testing. There are altogether 606 distinct processes drawn from more than 2000 sessions during the selected four training days, and 55 intrusion sessions in the other seven-week training data. It is worth noting that an intrusive session may contain only a small part of intrusive processes or even only one, such as *eject, format, ffb, spy* and so on, e.g., *spy* is a multi-day scenario in which a user breaks into a machine with the purpose of finding important information where the user tries to avoid detection(stealthy attack). 400 out of 606 distinct processes and 30 intrusive processes out of 55 intrusion sessions were selected as training data. These 30 distinct intrusive processes cover almost all of the attack types of the DARPA training data. In order to verify the performance of Robust SVM and One-class SVM, 15 out of the original 30 intrusive processes were disguised as normal processes and incorporated into the truly normal ones. The BSM data of the third day of the seventh training week which contains 412 sessions and 5285 normal processes was chosen as normal part of the test data set(some

of them are same in order to count false alarms), while abnormal part of testing data contains 25 out of 55 intrusive sessions. The training data and testing data are illustrated in Table 1.

	clean data (process)		noisy data (process)	
	normal	intrusive	normal	intrusive
Training	400	30	415	15
Testing	5285	25 sessions	5285	25 sessions

Table 1. Experiment Data Sets

There are 35 clear or stealthy attack instances included in 55 intrusive sessions, representing all types of attacks and intrusion scenarios in the seven-week training data. When a process is determined as intrusive, the session that the process is associated is classified as an attack session, and each attacks counts as one detection. The detection accuracy is then calculated as the rate of detected attacks, and the false positive probability is defined as the rate of misclassified normal processes(these two terms are not rigorous symmetry here). One potential problem of intrusion detection here is that, as the time passes, the old hyperplane got by SVMs will no longer accurately distinguish the normalities or anomalies, one solution to handle this *concept drift* is periodically retrain the SVMs, therefore, training time is another important factor worth consideration.

4.2. Results and Discussion

We did experiments over clean training data and noisy training data respectively using different SVMs we presented above. All the SVMs were implemented with the RBF kernel function(i.e. $K(x_i, x_j) = e^{\|x_i - x_j\|/\sigma}$), and the best hyperplane were obtained by varying the regulation parameters. The performance of various SVMs were compared in terms of *detection accuracy, the number of support vectors* and *training time*.

Table 2 illustrated the detection accuracy of conventional SVM, RSVM and One-class SVM over clean training data and noisy training data respectively. The intrusion detection systems usually require the false positive rate lower than 1% due to the fact that too much false positive would make the systems inefficacy. The results showed that SVM, RSM, and One-class almost have the same performance over clean training data, but RSVM has better performance than traditional SVM over noisy data due to its ability of solving the overfitting problem. After modification, the detection accuracy of all the SVCs have a little improvement, which shows that the modified methods have better generalization performance. It is worth pointing out that One-class SVM does not have great change over clean training data and noisy data, because it is an unsupervised approach, which can be trained with unlabelled training data.

Training Data	Algorithms	SVM (%)	RSVM (%)	One-class SVM(%)
Clean Data	Original	80.00	83.33	83.33
	Modified	83.33	80.00	86.67
Noisy Data	Original	50.00	73.33	83.33
	Modified	53.33	76.67	86.67

Table 2. Comparison of the Detection Accuracy when FP. less than 1%

Another factor worth considering is *the number of support vectors*. As we know, classifying new examples by SVCs is the process of solving the quadratic programming problem, and the computational complexity of SVCs is of linear proportion to the number of SVs, therefore, SVMs with less SVs require less running time, which benefits online intrusion detection significantly. As illustrated in table 3, traditional SVM and RSVM have more support vectors over the clean training data than over the noisy training data because of the misclassified elements, and the number of SVs of methods modified by OSVM is generally less than that of the original ones. One-class SVM nearly keep unchanging. After getting the optimal boundaries, we use 5310 testing process to test them for comparing running time and test errors. The results showed that the reduced SVs did not effect test errors.

Training Data	Algorithms	SVM	RSVM	One-class SVM
Clean Data	Original	46	34	37
	Modified	43	32	30
Noisy Data	Original	41	19	37
	Modified	36	19	30

Table 3. Comparison of the Number of SVs

Besides the detection accuracy and support vectors, another problem that needs to be addressed is the *training time* of intrusion detectors. Available intrusion detection methods rely on too strong assumption that labelled training data can be obtained readily with high quality, which undermines their efficiency and limits their application. An ideal IDs should be trained with any provided data even online, therefore, achieve satisfied detection accuracy during a certain training time as short as possible is very important for an

IDS works online. Table 4 shows the ratio of the training time for SVMs and their modified methods with clean data and noisy data respectively.

Training Data	SVM(%)	RSVM(%)	One-class SVM(%)
Clean Data	56.01	51.61	59.4
Noisy Data	65.12	66.67	60.2

Table 4. Ratio of the Training Time for the Modified SVMs/Original SVMs

The training time of the modified SVMs were much less than that of the original ones, while RSVM and One-class SVM need more training time than conventional SVM in order to get high detection accuracy with false positive less than 1%. The training time of modified SVMs we got represent an average performance over 50 trials, and it highly depends on the distribution of the SVs in the training sequence we provided. Our modified algorithms deteriorated severely in the case that anomaly points were provided after most of the normal points had been supplied due to the suddenly change of the boundaries in their nature. Under such cases, modified algorithms performs narrowly better than original algorithms. In our experiment, the fastest trial only takes 8.3s in contrast to he worst trial taking 123.3s, so we took average performance over 50 trivial to compare with the original algorithms.

5. Conclusion

In this paper, we modified traditional SVM, RSVM and One-class SVM respectively in virtue of the idea from OSVM. The preliminary experiments with the 1998 DARPA BSM audit data showed that our modified algorithms outperform the original SVMs in terms of the number of support vectors and training time while keeping comparable detection accuracy. Specifically, the running time of the modified algorithms can be reduced greatly because the fewer support vectors, and the training time can be saved significantly by the effective decomposition of the original algorithms for faster convergence. Both of these two aspects are essential for the design of an satisfactory online IDS. One significant discovery is that the modified One-class SVM can be trained online with unlabelled data sets because of its unsupervised nature, which breaks the strong assumption that most of the existing methods are based on. It also inspires us for further research about the application of online training with related effective unsupervised learning methods for intrusion detection.

Acknowledgment

This research is conducted as a program for the "Fostering Talent in Emergent Research Fields" in Special Coordination Funds for Promoting Science and Technology by Ministry of Education, Culture, Sports, Science and Technology.

References

[1] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery,* 2, 121-167(1998).

[2] C.C.Chang, C.W.Hsu, C.J.Lin. The analysis of decompostion methods for support vector machines. *IEEE Trans. Neural Networks* 11(2000) 1003-1008.

[3] Forrest,S.,Hofmeyr,S.A.,&Longstaff,T.A. A sense of self for UNIX processes. *In proceedings of 1996 IEEE Symposium on Security and Privacy.* Los Alamitos, CA: IEEE Computer Society Press.

[4] Wenjie Hu, Yihua Liao, V.Rao Vemuri. Robust Support Vetor Machines for Anomaly Detection in Computer Security. *The 2003 International Conference on Machine Learning and Applications(ICMLA'03).* Los Angeles, California, June 2003.

[5] K.W.Lau, Q.H.Wu. Online training of support vector classifier. *Pattern Recognition* 36(2003) 1913-1920.

[6] Wenke Lee, Salvatore J.Stolfo. Data Mining Approaches for Intrusion Detection. *In Proceedings of the 7th USENIX Security Symposium,* 1998.

[7] Y.Liao and V.R.Vemuri. Use of K-Nearest Neighbor Classifier for Intrusion Detection. *Computers & Security,* 21(5), pp439-448, Oco,2002.

[8] E.Lundin and E.Jonhsson. Anomaly-based intrusion detection: privacy concern and other problems. *Computer Networks,* Vol.34, pp623-640,2000.

[9] MIT Lincoln Laboratory, *http://www.ll.mit.edu/IST/ideval/.*

[10] Binh Viet Nguyen. An Application of Support Vector Machines to Anomaly Detection. *Research in Computer Science–Support Vector Machine,* report, Fall 2002.

[11] Scholkopf,B. Platt,J.C. Shawe-Taylor,J. Smola, A.J. and Williamson,R.C.2001. Estimating the support of a high-dimensional distribution. *Neural Computation 13(7)*:1443-1471.

[12] Qing Song, Wenjie Hu and Wenfan Xie. Robust Support Vector Machine for Bullet Hole Image Classification. *IEEE Transaction on Systems, Man and Cybernetics Part C.* Vol 32. Issue 4, pp440-448. Nove.2002.

[13] C.Warrender, S.Forrest and B.Pearlmutter. Detecting Intrusion Detection Using System Calls: Alternative Data Models. *In Proceedings of 1999 IEEE Symposium on Security and Privacy,* pp133-145, Oakland, 1999.

[14] Nong Ye, Xiangyang Li, Qiang Chen, Syed Masum Emran, and Mingming Xu. Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data. *IEEE Transaction on Systems, Man, and Cybernetics-Part A:Systems and Humans,* Vol.31, No.4, July 2001.