IABA: An Improved PNN Alogrithm for Anomaly Detection in Network Security Management

LIU Wu[1)], DUANHai-xin[1)],REN Ping[2)], WU Jian-ping[1)]
1) Network Research Center of Tsinghua University 100084, Beijing, China
2) School of Economics & Management, Chongqing Normal University Chongqing, P. R. China

Abstract—**Anomaly Detection is very difficult in network security management. In this paper, we consider anomaly detection using an improved probabilistic neural networks. We first introduce a Basic Adaptive Boost Algorithm (BABA) and analysis its drawbacks and then introduce an Improved Adaptive Boost Algorithm (IABA) to classify the detected event as normal or intrusive.**

Keywords Intrusion Detection System, Probabilistic Neural Network, Network Security

I. INTRODUCTION

Intrusion detection is the process of monitoring events in a computer system or network, and analyzing them for security problems or violations of security policy [1]. The events in a computer system may be monitored using sequences of system calls. Usually, such sequences are represented by number strings. However, one should be aware that these numbers are actually nominal symbols corresponding to systems calls, and should not be compared to each other as numbers. There is, for example, generally no ordering relationship. One way to compare these sequences of system calls as symbolic strings is to use Hamming distance [2].

Given a group of labeled patterns (kernel centers), a Probabilistic Neural Networks (PNN) estimates probabilities that a given input pattern belongs to each known class. Selecting the class with highest estimated probability, we can classify the input pattern. Our simulations show that, on symbolic patterns, PNNs with Hamming distance kernels outperform regular PNNs with Euclidena distance kernels.

This paper is organized as follows. In section 2, we introduce some related works. Section 3 describes the basic algorithm BABA based on PPN and Adaptive Boost and analysis its drawbacks. In section 4 we introduce the improved algorithm IABA based on BABA. In section 5 we describe the application of IABA to Anomaly Detection in Network Security Management. Finally we present our conclusions in section 6.

II. RELATED WORKS

IDSs can be classified into Anomaly Detection and Misuse Detection. Anomaly detection finds novel attacks, but the false alarm rate is high. Furthermore, if an attack occurs during training, it may be established as part of the normal behavior. Misuse Detection models attacks as specific patterns, then systematically scans the system for their occurrences [6][7].

Hofmeyr [3] considered host-based anomaly detection. The behavior of a process running under super-user level privileges was monitored through sequences of its system calls. The process itself was treated as a black box, that is, details of its algorithms and data structures were not analyzed. Those corresponding to normal behavior were stored in a database. To test a new trace for intrusion, its fragments were compared with those in the database. If a fragment exited, it was considered normal; otherwise a mismatch was conted or a similarity measure based on Hamming distance was calculated.

Marin [5] compared variations (stide and t-stide) of the mismatch counting similarity measure described in [3] with a data mining approach (RIPPER, [4]) and a finite state machine, hidden Markov model (HMM). The methods under study performed well, though no clear winner was determined. Ghosh [5] studied probabilistic properties of system call sequences. They compared performance of Hotelling’s T2 and Chi-Square multivariate tests, and a first-order Markov model for anomaly detection. Florez [8] applied Adaptive Boost with three different neural network topologies for classification of system call sequences. Yeung [9] used Parzen-window nonparametric density estimation, which is very closely related to probabilistic neural networks, for network-based anomaly detection. In [10], the authors used their hypothesis testing approach to novelty detection applying HMMs and information-theoretic modeling for host based intrusion detection using system call and shell command sequences. The authors found that dynamic modeling using HMM worked better than static models for system call sequences.

III. BASIC ALOGRITHM BARA AND DRAWBACKS

A. *Probabilistic neural networks*

Probabilistic Neural Networks (PNN) is a special case of radial basis function networks developed for classification problems [14][15]. The output of PNN is discrete and takes values from 1 to K, where K is the number of classes that the network is designed to recognize. Given a training set, a PNN can be built directly without iterative training. Such a network, built upon a good, representative training sample, has a good generalization capability. Given sufficient training data, it converges to the optimal Bayesian classifier.

A PNN has two layers. The first layer calculates a radial basis function (e.g., a Gaussian) for distances between the input vector and each of the kernel vectors. The second layer computes K sums of the radial basis kernel outputs and finds the largest sum. The class, corresponding to the largest sum,

This work is supported by grants from the National Natural Science Foundation of China (Grant No. 60203044, 90412010) , China 863 Project #2008BAH37B04 and 242 program#(242)2007A07

has the maximum probability of being correct, and is given as the output by the network.

When the network is designed, the kernel centers (weights) of the first layer are set to the training input vectors. Each weight vector in the second layer has K elements and contains a priori probabilities for that kernel vector belonging to each class. Typically, the element corresponding to the target class for this training pair is set to 1, and the other elements are 0. Thus the K sums in the second layer are computed over those kernels that belong to the same class in the training set.

Let $\{(x^i, y^i)\}_{i=1}^{M}$ be a training set of M input-output pairs (x^i, y^i) . Each target vector $y^i = (y_1^i, \cdots, y_k^i)$ has K elements. Only one element, y_k^i of y^i corresponding to the target class k, is set to 1, the others $(y_j^i : j \neq k)$ are 0. The PNN designed on this training set is as follows. The first (hidden) layer has M neurons computing the Gaussian kernel functions for input vector x, $h_i = e^{-\frac{\|x-x^i\|^2}{2\sigma^2}}, i = 1, \cdots, M$

Here, h_i is the output of the i-th hidden unit, x^i is the i-th training vector, M is the number of training vectors, and x is the input vector to be classified. σ is a parameter having impact on the generalization ability of the network. When it is near zero, the network performs as a nearest neighbor classifier. The larger σ is, the more nearby training vectors influence the decision.

The second (output) layer has K neurons. Each neuron calculates the sum, $o_k = l_k \sum_{i=1}^{M} y_k^i h_i, k = 1, \cdots, K$

Where o_k is the k-th output, l_k is the loss associated with incorrect classification of a pattern from class k, h_i is the output of the i-th hidden unit, y_k^i is the known probability that the i-th training vector belongs to class k, M is the number of training vectors, and K is the number of classes. Note that, if normalized, the output o_k becomes the a posteriori probability of input vector x belonging to the class k. the output neuron with the largest sum o_k wins, and the input x is classified to class k.

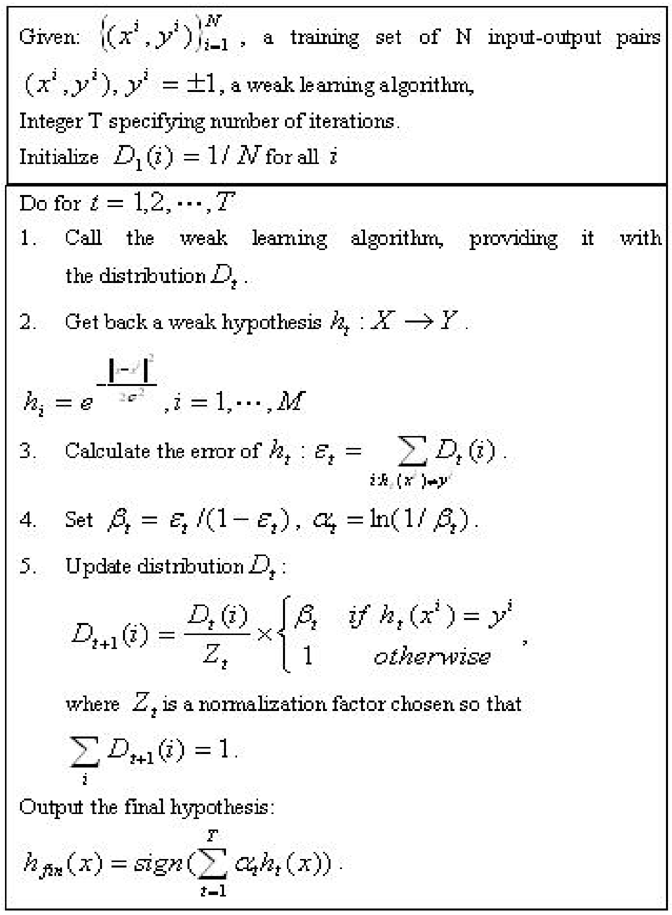
Now consider a two-class problem where targets y are scalars $y^i = \pm 1$. The output layer now can be reduced to one neuron $o = \sum_{i=1}^{M} y^i h_i$

Here, h_i is the output of the i-th hidden unit and M is the number of training vectors. The sign of this sum indicates the class, $y = sign(o)$, and the absolute value may serve as a "degree of belief" that this classification is correct.

B. *Basic Adaptive Boost Algorithm (BABA)*

Boosting is a committee machine learning meta-algorithm that can be used to improve (boost) performance of an underlying weak learning algorithm. A weak learning algorithm is defined as an algorithm that performs better than a random guessing, i.e. correctly classifies more than half the input vectors. Boosting combines several weak classifies (hypotheses) in a committee machine, so the resulting ensemble becomes a strong classifier that is able to correctly classify almost all input vectors.

Adaptive Boost [13][14] assigns each weak hypothesis a weight value depending on how good this hypothesis performed on the training set. The final hypothesis is calculated as a weighted sum of outputs of weak hypotheses. Moreover, during the training, each input/output pair is also assigned a weight depending on how difficult this training pair has been so far. Thus, each subsequence weak hypothesis will pay more attention to hard examples, and more consideration will be given to its performance on such examples.



```
Given: $\{(x^i, y^i)\}_{i=1}^{N}$ , a training set of N input-output pairs
$(x^i, y^i), y^i = \pm 1$, a weak learning algorithm,
Integer T specifying number of iterations.
Initialize $D_1(i) = 1/N$ for all $i$

Do for $t = 1, 2, \cdots, T$
1.  Call the weak learning algorithm, providing it with
    the distribution $D_t$.
2.  Get back a weak hypothesis $h_t : X \to Y$.
$h_i = e^{-\frac{\|\cdot\|^2}{2\sigma^2}}, i = 1, \cdots, M$
3.  Calculate the error of $h_t$ : $\varepsilon_t = \sum_{i:h_t(x^i) \neq y^i} D_t(i)$.
4.  Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$, $\alpha_t = \ln(1/\beta_t)$.
5.  Update distribution $D_t$:
    $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & if\ h_t(x^i) = y^i \\ 1 & otherwise \end{cases}$,
    where $Z_t$ is a normalization factor chosen so that
    $\sum_i D_{t+1}(i) = 1$.
Output the final hypothesis:
$h_{fin}(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$.
```

Fig. 1 The Basic Adaptive Boost Algorithm

Pseudo code for BABA is given in Fig. 1. The algorithm maintains a weight distribution D over the training set. Initially, all weights are equal, and then more weight is given to incorrectly classified examples. Adaptive Boost calls the weak learning algorithm, providing it with the distribution D. The weak learner is to take these weights into the account, e.g. by weighting the training error with D, or by sub-sampling the training set using D as a probability distribution, the error ε of misclassified examples. It can be assumed that $\varepsilon \le 1/2$; if not, negating the answers of weak hypothesis will do the trick. The smaller the error ε, the larger is the weight, and the more influence this weak hypothesis will have in the end.

This error ε is also used to update the training set weights D. Weights for misclassified examples are increased, and weights for correctly classified examples are decreased; as a result, the next weak hypothesis will focus on the examples that are hard to recognize. After that the distribution D is normalized so that its sum is equal to one.

The algorithm proceeds until T weak hypotheses are obtained. The final hypothesis works as follows. Each of T weak hypotheses is presented with the input, and then its output is weighted with its importance α , and summed up with weighted outputs of other weak hypotheses. The sign of the sum gives the predicted class label, and the magnitude can be used as a measure of "confidence margin".

The number of weak classifiers T is the only tunable parameter of the algorithm. For larger T the training error drops exponentially, and the generalization error continues to decrease long after the training error reaches zero, as the confidence margins widen. Subsequent increase of the generalization error rarely happens and can be avoided using a verification set. On the other hand, using many weak hypotheses requires more time to run them; hence, a very large T is inefficient.

C. Drawbacks of BABA

1. In BABA, there is a basic assumption: $\varepsilon \le 1/2$, which may leads to the following drawbacks. On the one hand, function $h_t : X \to Y$ is determined by user, and ε is related to actions before, so it requires that $\varepsilon_t \in [0,1]$. On the other hand, if the assumption $\varepsilon \le 1/2$ is not satisfied, function $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & if\ h_t(x^i) = y^i \\ 1 & otherwise \end{cases}$ will be meaningless;
2. In addition, there exist the following drawbacks in function $\beta_t = \varepsilon_t / (1-\varepsilon_t)$ and $\alpha_t = \ln(1/\beta_t)$:
   a) When $\varepsilon_t = 1$, function β_t is meaningless, and when $\varepsilon_t = 0$, function α_t is meaningless;
   b) When $\varepsilon_t \to 1$, function $\beta_t \to +\infty$; when $\varepsilon_t \to 1$, $\alpha_t \to -\infty$; when $\varepsilon_t \to 0$, $\alpha_t \to +\infty$, which may affect the function's performance. Especially in $h_{fin}(x) = sign(\sum_{t=1}^{T} \partial_t h_t(x))$, if some item tends to $+\infty$ or $-\infty$, other items will not take any effects.

So, we think that in BABA, the functions β_t , α_t and D_t should satisfy the following conditions:

a) When $\varepsilon_t = 0$, $\beta_t > 0$ should be very small; when $\varepsilon_t = 1/2$, $\beta_t = 1$; when $\varepsilon_t > 1/2$, β_t should increases monotonically, and when $\varepsilon_t \to 1$, function β_t should increase rapidly;

b) When $\varepsilon_t = 0$, α_t should be very big; when $\varepsilon_t = 1/2$, $\alpha_t = 1$; When $\varepsilon_t = 1$, α_t should be very small; α_t should decreases monotonically, and when $\varepsilon_t \to 0$, function α_t should decrease rapidly;

c) Function D_t should always be meaningful for all $\varepsilon_t \in [0,1]$

To overcome the drawbacks in BABA, we will introduce an improved algorithm in the next section.

IV. IMPROVED ADAPTIVE BOOST ALGORITHM (IABA)

A. Some Functions used for our Revised Algorithm

Let $\delta \le 1/2$ be a parameter. For $s \in [0,1]$, we define 3 functions, β_t , δ_t and D_t as follows.

1. Function $\beta_t := \begin{cases} \delta/(1-\delta) & \varepsilon_t \le \delta \\ \varepsilon_t/(1-\varepsilon_t) & \delta < \varepsilon_t < 1-\delta \\ (1-\delta)/\delta & 1-\delta \le \varepsilon \le 1 \end{cases}$

Generally, we take the parameter δ as a small number. Function $\beta^{\delta}(s)$ satisfies the following conditions:

(1) When $s \le \delta$, $\beta^{\delta}(s) = \delta$;

(2) When $s \ge 1-\delta$ $\beta^{\delta}(s) = 1/\delta$

(3) $\beta^{\delta}(1/2) = 1$

(4) $\beta^{\delta}(s)$ is a continuous function on $[0,1]$

2. Function $\delta_t := 1/\beta_t$

Generally, we take the parameter δ as a small number. Function $\alpha^{\delta}(s)$ satisfies the following conditions:

(1) When $s \le \delta$, $\alpha^{\delta}(s) = 1/\delta$

(2) When $s \geq 1-\delta$, $\alpha^{\delta}(s) = \delta$

(3) $\alpha^{\delta}(1/2) = 1$

(4) $\alpha^{\delta}(s)$ is a continuous function on $[0,1]$

3. Function D^i_{t+1} is defined as follows:

$$D^{i,\delta}_{t+1} := \frac{D^{i,\delta}_t}{Z_t}\begin{cases}\min\{\beta^{\delta}_t, 1\} & h_t(x^i) = y^i \\ \max\{\beta^{\delta}_t, 1\} & h_t(x^i) \neq y^i\end{cases}$$

Where $Z_t := \sum_{i\in N, h_t(x^i)=y^i} \min\{\beta^{\delta}_t, 1\} + \sum_{i\in N, h_t(x^i)\neq y^i} \max\{\beta^{\delta}_t, 1\}$

Thus, we have that $\sum_{i=1}^{N} D^{i,\delta}_{t+1} = 1$

Here we give a parameter δ. If we want to encourage the good performance and penalize the bad performance, then we can decrease the parameter δ.

Generally we take δ a very small positive number. We can check that these functions overcome the disadvantages of the before ones.

Using the above functions, by the algorithm, we can prove the following properties.

B. The Improved Algorithm

Step 0: take a training set of N input-output pairs $(x^i, y^i), i = 1,\cdots,N$, and take an integer T specifying number of iterations. Take the initial $G_1(i) = 1/N$ for all $i = 1,\cdots,N$, and take an initial weak learning algorithm $h_1(x^i)$ stochastically. Let $t = 1$.

Step 1: Let $I_t := \{i \mid sign(h_t(x^i)) = y^i\}$ and $J_t := \{1,\cdots,N\} \setminus I_t$. Calculate the error of h_t:

$$\varepsilon_t := \sum_{i\in J_t} D_t(i)$$

Go to Step 2;

Step 2: If $\varepsilon_t = 0$, then stop and take $h_{fin}(x) := h_t(x)$; otherwise go to Step 3;

Step 3: If $t = T$, then go to Step 6; otherwise, for the positive number $0 < \delta < 1/2$, take

$$\beta_t := \begin{cases}\delta/(1-\delta) & \varepsilon_t \leq \delta \\ \varepsilon_t/(1-\varepsilon_t) & \delta < \varepsilon_t < 1-\delta \\ (1-\delta)/\delta & 1-\delta \leq \varepsilon \leq 1\end{cases}$$

$$\delta_t := 1/\beta_t$$

And update the distribution D_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t}\begin{cases}\min\{\beta_t, 1\} & sign(h_t(x^i)) = y^i \\ \max\{1, \beta_t\} & otherwise\end{cases}$$

Go to Step 4;

Step 4: Stochastically take a weak learning algorithm $\bar{h}_t(x)$, and let $K_t := \{i \mid sign(\bar{h}_t(x^i)) = y^i, i = 1,\cdots,N\}$. If $K_t \leq I_t - 1$, let $h_{t+1}(x := \bar{h}_t(x))$ and $t := t+1$, go to Step 1; otherwise go to Step 5;

Step 5: Update $h_t(x)$. Let $i_t \in J_t$, take

$$h_{t+1}(x) := h_t(x) + q\,sign(y^{i_t})\prod_{i\in I_t}(x - x_i)^2 \qquad (1)$$

and $t := t+1$, go to Step 1; otherwise go to Step 6;

Step 6: If $I_t \setminus K_t = \phi$, then let $h_{t+1}(x := \bar{h}_t(x)$ and $t := t+1$, go to Step 1; otherwise let $i_t \in I_t \setminus K_t$, and let

$$h_{t+1}(x) := \bar{h}_t(x) + q\,sign(y^{i_t})\prod_{i\in I_t}(x - x_i)^2 \qquad (2)$$

$t := t+1$, go to Step 1;

Step 7: Take $t_{fin} = t$ and $h_{fin}(x) := sign\sum_{t=1}^{t_{fin}} \alpha_t h_t(x)$.

For the above algorithm, we have the following results:

Theorem 1 There exists a q_0 such that when $q \geq q_0$, $t_{fin} \leq N$ for any $0 < \delta < 1/2$; furthermore, if $T \geq N$, then $h_{fin}(x^i) = y^i, \forall i = 1,\cdots,N$

Proof: Firstly, we can prove that for any $1 \leq t \leq T$, one of the following conditions holds: $I_t = N$ and $I_{t+1} \leq I_t - 1$.

In fact, if $\varepsilon_t = 0$, then $I_t = N$. If $\varepsilon_t > 0$, let

$$q_t := \max_{j\in J_i}\left\{\frac{|h_t(x^j)|}{\prod_{i\in I_t}(x^j - x^i)^2} + 1\right\}$$, then when $q \geq q_t$,

1) if $K_t \leq I_t - 1$, then by Step 4, we have $I_{t+1} = K_t \leq I_t - 1$;

2) if $K_t > I_t - 1$, by Step 5, we know that $I_{t+1} \leq I_t - 1$. In fact, if $K_t > I_t - 1$, by (1), when $q \geq q_t$, we know that $sign(h_{t+1}(x^{i_t})) = sign(y^i)$, which means that $sign(h_{t+1}(x^{i_t})) = y^i$. Hence, $h_{t+1}(x^{i_t}) = y^i$, $\forall i \in I_t$ and $h_{t+1}(y^{i_{t}} = y^{i_t})$, where $i_t \notin I_t$. Thus, $I_{t+1} \leq I_t - 1$.

Therefore, when $q \geq q_0$, we must have that $t_{fin} \leq N$ and if $T \geq N$, then $h_{fin}(x^i) = y^i, \forall i = 1,\cdots,N$ ■

V. IMPROVED ADAPTIVE BOOST ALGORITHM (IABA)

Probabilistic Neural Networks were used as the weak learning algorithm for Adaptive Boost. The distribution provided by Adaptive Boost was used to sample the training set—only 1% of training examples were used for PNN kernels. The resulting PNN was verified on the whole training set, however. If, nevertheless, the network was able to classify the whole set correctly, the number of kernels was reduced until the network gave errors on the training set. This was done for two reasons: first, Adaptive Boost would not work for a perfect learning algorithm (such a network would get an infinite weight in committee voting), and second, reducing the number of network parameter is a known way to reduce the structural risk and improve generalization ability.

The value kernel spread was optimized so the network had the least error on the training set. The PNN training error was calculated similarly to the Adaptive Boost algorithm, summing up the distribution weights for misclassified training examples. For "attack" training examples a special consideration was made: if any of the input sequences belonging to a particular session was correctly classified as "attack", then all the sequences from this particular session are considered to be classified correctly. Thus, one alarm would be enough to detect an attack. "Normal" sequences did not get such special treatment and were evaluated individually. This approach discourages false positives.

Ensembles consisting of 3 to 25 PNNs with Hamming distance kernel were trained using IABA. From 200 to 430 ensembles were trained for each set of networks. We also trained from 10 to 40 ensembles of 3 to 22 PNNs with Euclidean distance kernels. The ensembles were trained on week 1-2 data and tested on week 3 and 4 data.

Fig. 2 and Fig. 3 compares week 3 test performance for Euclidean and Hamming distance for different ensemble sizes. It can be seen from Fig. 2 and 3 that, PNNs with Hamming distance kernels achieve much better performance earlier with smaller ensembles than that of PNNS with Euclidean distance.

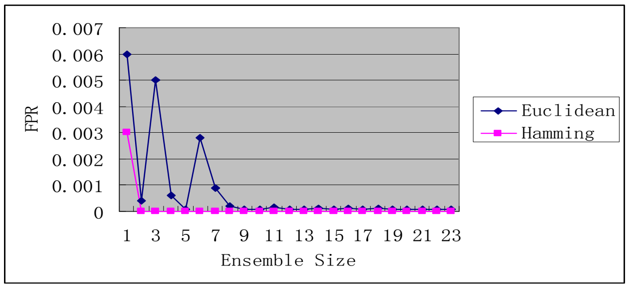


Fig. 2 Average false positive (FPR) rates

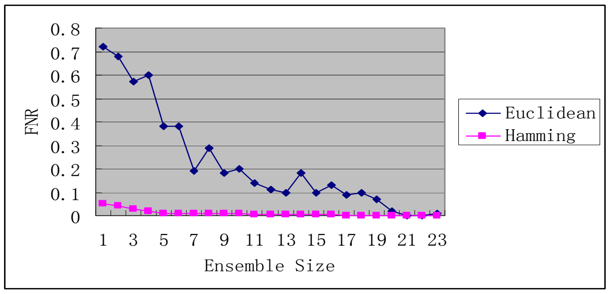


Fig. 3 Average false Negative (FNR) rate

VI. Conclusion

In this paper, we discussed data mining techniques for building intrusion detection models. We first introduced a Basic Adaptive Boost Algorithm (BABA) and analysis its drawbacks and then introduce an Improved Adaptive Boost Algorithm (IABA) to classify the detected event as normal or intrusive.

References

[1] Wu LIU, WU Zhi-you, DUAN Hai-xin, LI Xing, WU Jian-ping, Probabilistic neural networks based network security management, J. of Communication and Computer (ISSN1548-7709), Feb. 2008, Volume 5, No.2, Page(s):19-24.

[2] Wu Liu, Study on Intrusion Detection Technology with Traceback and Isolation of Attacking Sources, PhD Thesis, 2004.

[3] S. A. Hofmeyr, S. Forrest, and a. Somayaji. "Intrusion detection using sequences of system calls". *Computer Security*,. *6*(3): 151-180, 1998.

[4] W. Lee, S. J. Stolfo, and P. K. Chan. "Learning patterns from Unix process execution traces for intrusion detection", in Proc. AAAI Workshop: AI Approaches to Fraud Detection and Risk Management, 2003, pp. 50-56.

[5] J. Marin, D. Ragsdale, and J. Sirdu, "A hybrid approach to the profile creation and intrusion detection", in Proc. DARPA Information Survivability Conference & Exposition II, 2001, pp. 69-78.

[6] D. Endler, Intrusion detection: Applying machine learning to Solaris audit data, in Proc. Annual Computer Security Applications Conference 2002, pp. 268-279.

[7] J. Cannady, Artificial Neural networks for misuse Detection, in Proc. National Information Systems Security Conference, 2001, pp. 443-456

[8] G. Florez and L. Boggess, Using Adaptive Boost with Different Network Topologies for System Call Analysis, in smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Complex Systems and Artificial Life, Volume 12, Proc. Of the Artificial Neural Networks in Engineering Conference, 2002, pp. 135-140.

[9] D.-Y. Yeung and C. Chow, Parzen-window network intrusion detectors, in Proc. 16th International Conference on Pattern Recognition, 2002, vol. 4, pp. 385-388.

[10] D.-Y. Yeung and Y. Ding, Host-based intrusion detection using dynamic and static behavioral models, Pattern recognition, vol. 36, no. 1, pp. 229-243, Jan. 2003.

[11] D.F. Specht, Probabilistic Neural Networks, Neural Networks, vol. 3, no. 1, pp. 109-118, 1990.

[12] C.M. Bishop, Neural Networks for pattern Recognition. New York: Oxford University Press, 2005, pp. 177-191.

[13] Y. Freund and R. E. Schapire, Experiments with a New Boosting Algorithm, in Proc. Machine Learning: International conference, 2006, pp. 148-156.

[14] Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Computer and System Sciences, vol. 55, no. 1, pp. 119-139, Aug. 1997.