Proceedings of the 2001 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 5-6 June, 2001

HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification

Zheng Zhang, Jun Li, C.N. Manikopoulos, Jay Jorgenson, Jose Ucles

ECE Department, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA
Department of Mathematics, CCNY, Convent Ave. at 138 ST., New York, NY 100031, USA
Network Security Solutions, 15 Independence Blvd. 3rd FL., Warren, NJ 07059, USA
zxz9622@njit.edu, jxl1727@njit.edu, manikopoulos@adm.njit.edu, jjorgenson@mindspring.com

Abstract—In this paper we introduce the Hierarchical Intrusion DEtection (HIDE) system, which detects network-based attacks as anomalies using statistical preprocessing and neural network classification. We describe our system architecture and the statistical preprocessing technique and components. We tested five different types of neural network classifiers: Perceptron, Backpropagation (BP), Perceptron-backpropagation-hybrid (PBH), Fuzzy ARTMAP, and Radial-based Function. Our results indicate that BP and PBH provide more efficient classification for our data than the alternatives. We also stress-tested the entire system, which showed that HIDE can reliably detect UDP flooding attacks with attack intensity as low as five to ten percent of background traffic.

Index Terms—Network Security, Intrusion Detection, Anomaly Detection, Statistical Preprocessing, Neural Network Classification, IDS, IDA, HIDE.

I. INTRODUCTION

The ubiquity of the Internet poses serious concerns on the security of computer infrastructures and the integrity of sensitive data. Network intrusion detection aims to protect networks and computers from malicious network-based attacks. The underlying assumption of intrusion detection is that an attack will noticeably affect system performance or behavior. Intrusion detection techniques can be partitioned into two complementary approaches: *misuse detection*, and *anomaly detection*. *Misuse detection systems*, such as [1][2], model the known attacks and scan the system data for occurrences of these patterns. *Anomaly detection systems*, such as [3], [7], flag intrusions by observing significant deviations from typical or expected behavior of the system or users.

Statistical modeling followed with classical or neural network classification have been utilized in some anomaly intrusion detection systems. For example, NIDES [3] represents user or system behaviors by a set of statistical variables and detects the deviation between the observed and the standard activities. A system that identifies intrusions using packet filtering and neural networks was introduced in [4]. The work of Ghosh et al [7] studied the employment of neural network classifiers to detect anomalous and unknown intrusions against a software system. In [12], Kolmogorov-Smirnov statistics was used to model and detect Denial-of-Service and Probing attacks.

We proposed the Hierarchical Intrusion DEtection (HIDE) system in [8]. HIDE is an anomaly network intrusion detection system, with hierarchical architecture, that uses statistical models and neural network classifiers to detect attacks. Here, we report our experimental results of the performance of five different types of neural networks, as well as the results of traffic intensity stress-testing on HIDE.

The rest of the paper is organized as follows. The system architecture of HIDE is outlined in Section 2. Section 3 introduces the statistical model that we are using. Section 4 provides the five neural networks we tested. The simulation environment is described in Section 5. The experimental results using neural network classifiers are presented in section 6. Section 7 reports the stress-testing results on HIDE. Section 8 draws some conclusions and outlines future work.

II. SYSTEM ARCHITECTURE

Our system is a distributed hierarchical application, which consists of several tiers with each tier containing several Intrusion Detection Agents (IDAs). IDAs are IDS components that monitor the activities of a host or a network. Different tiers correspond to different network scopes that are protected by agents affiliated to them.

ISBN 0-7803-9814-9/$10.00 © 2001 IEEE

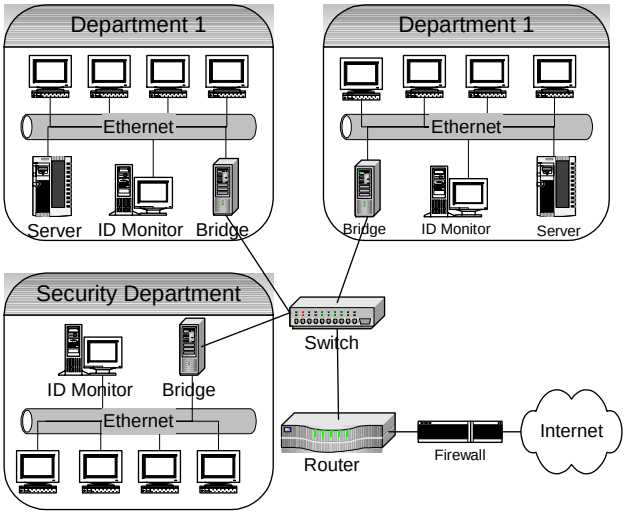

Fig. 1 Sample Network

For the sample network shown in Fig. 1, the intrusion detection system can be divided into 3 tiers. Tier 1 agents monitor system activities of the servers and bridges within a department and periodically generate reports for Tier 2 agents. Tier 2 agents detect the network status of a departmental LAN based on the network traffic that they observe as well as the reports from the Tier 1 agents within the LAN. Tier 3 agents collect data from the Tier 1 agents at the firewall and the router as well as data of Tier 2 agents. The system hierarchy is shown in Fig. 2.

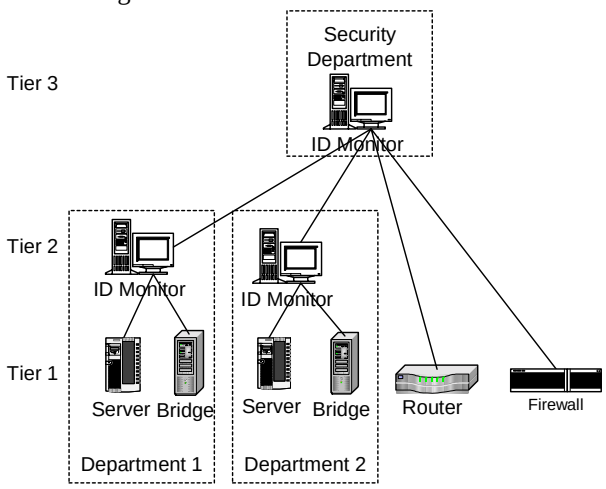

Fig. 2 System Hierarchy

Because this system is distributed and hierarchical, the IDAs of all tiers have the same structure. A diagram of an IDA is illustrated in Fig. 3, which consists of the following components: the probe, the event preprocessor, the statistical processor, the neural network classifier and the post processor. The functionalities of these components are described as below:

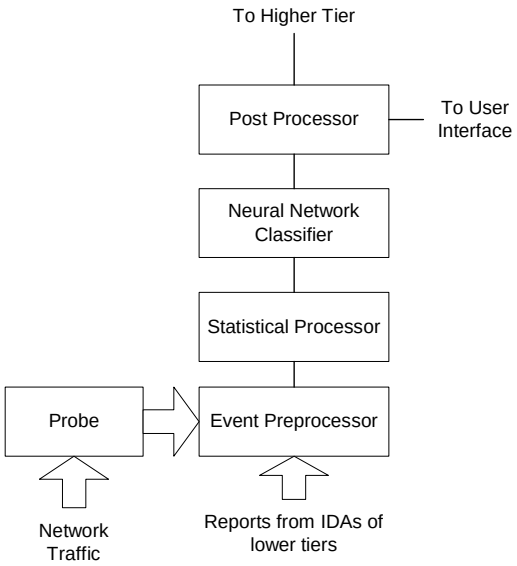

Fig. 3 Intrusion Detection Agent

- *Probe:* Collects the network traffic of a host or a network, abstracts the traffic into a set of statistical variables to reflect the network status, and periodically generates reports to the *event preprocessor*.
- *Event Preprocessor:* Receives reports from both the *probe* and *IDAs* of lower tiers, and converts the information into the format required by the *statistical model*.
- *Statistical Processor:* Maintains a reference model of the typical network activities, compares the reports from the *event preprocessor* to the reference models, and forms a stimulus vector to feed into the neural network classifiers. We will further discuss the statistical algorithms in Section 3.
- *Neural Network Classifier:* Analyzes the stimulus vector from the *statistical model* to decide whether the network traffic is normal or not. Section 4 will introduce the neural network classifiers used in the system in detail.
- *Post Processor:* Generates reports for the agents at higher tiers. At the same time, it may display the results through a user interface.

III. Statistical Model

Statistical methods have been used in anomaly intrusion detection systems [3]; however, most of these systems simply measure the means and the variances of some variables and detect whether certain thresholds are exceeded. SRI's NIDES [5][3] developed a more sophisticated statistical algorithm by using a χ^2-like test to measure the similarity between short-term and long-term profiles. Our current statistical model uses a similar algorithm as NIDES but with major modifications. Therefore, we will first briefly introduce some basic information about the NIDES statistical algorithm.

In NIDES, user profiles are represented by a number of probability density functions. Let S be the sample space of a random variable and events $E_1, E_2,\ldots, E_k$ a mutually exclusive partition of S. Assume P_i is the expected probability of the

occurrence of the event E_i, and let $P_i^{'}$ be the frequency of the occurrence of E_i during a given time interval. Let N denote the total number of occurrences. NIDES statistical algorithm used a χ^2-like test to determine the similarity between the expected and actual distributions through the statistic:

$$Q = N \times \sum_{i=1}^{k} \frac{(P_i^{'} - P_i)^2}{P_i}$$

When N is large and the events $E_1,E_2,\ldots,E_k$ are independent, Q approximately follows a χ^2 distribution with $(k-1)$ degrees of freedom. However in a real-time application the above two assumptions generally cannot be guaranteed, thus, empirically, Q may not follow a χ^2 distribution. NIDES solved this problem by building an empirical probability distribution for Q, which is updated daily in a real-time operation.

In our system, since we are using a neural network classifier to identify possible intrusions, we are not so concerned with the actual distribution of Q. However, because network traffic is not stationary and network-based attacks may have different time durations, varying from a couple of seconds to several hours or longer, we need an algorithm which is capable of efficiently monitoring network traffic with different time windows. Based on the above observations, we used a layer-window statistical model, Fig. 4, with each layer-window corresponding to a monitoring time slice of increasing size.

The newly arrived events will first be stored in the event buffer of layer 1. The stored events are compared with the reference model of that layer and the results are then fed into the neural network classifier to decide the network status during that time window. The event buffer will be emptied once it becomes full, and the stored events will be averaged and forwarded to the event buffer of layer 2. This process will be repeated recursively until the top level is reached where the events will simply be dropped after processing.

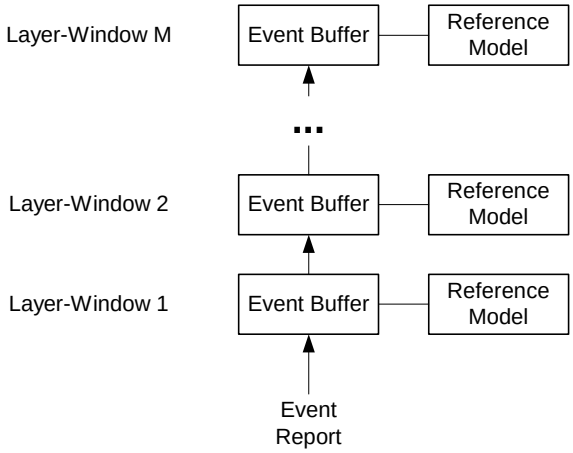

Fig. 4 Statistical Model

The similarity-measuring algorithm that we are using is shown below:

$$Q = f(N).[\sum_{i=1}^{k} |p_i^{'} - p_i| + \max_{i=1}^{k} (|p_i^{'} - p_i|)]$$

where $f(N)$ is a function that takes into account the total number of occurrences during a time window.

Besides similarity measurements, we also designed an algorithm for the real-time updating of the reference model. Let $\overline{p}_{old}$ be the reference model before updating, $\overline{p}_{new}$ be the reference model after updating, and $\overline{p}_{obs}$ be the observed user activity within a time window. The formula to update the reference model is

$$\overline{p}_{new} = s \times \alpha \times \overline{p}_{obs} + (1 - s \times \alpha) \times \overline{p}_{old}$$

in which α is the predefined adaptation rate and s is the value generated by the output of the neural network. Assume that the output of the neural network classifier is a continuous variable t between –1 and 1, where –1 means intrusion with absolute certainty and 1 means no intrusion again with complete confidence. In between, the values of t indicate proportionate levels of certainty. The function for calculating s is

$$s = \begin{cases} t, \text{if } t \geq 0 \\ 0, \text{otherwise} \end{cases}$$

Through the above equations, we ensured that the reference model would be updated actively for typical traffic while kept unchanged when attacks occurred. The attack events will be diverted and stored, as attack scripts, for future neural network learning.

IV. Neural Network Classifiers

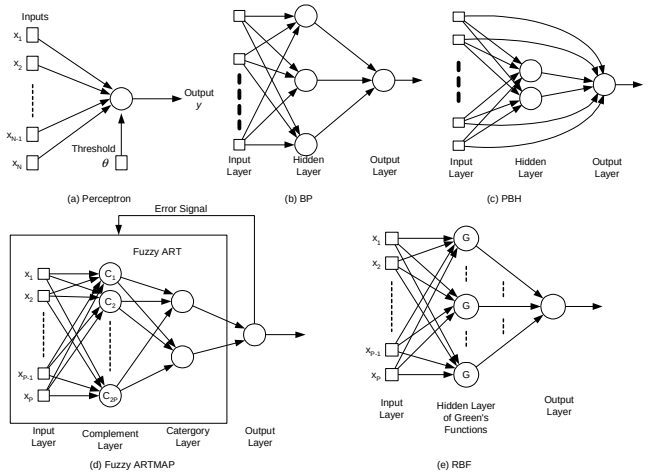

Fig. 5 Neural Network Architectures

Neural networks are widely considered as an effective approach to classify patterns. However the high computation requirements and the long training cycles have hindered their applications. In [4][7], BP neural networks were used to detect anomalous user activities. In [8], we deployed a hybrid neural network paradigm [6], called perceptron-backpropagation-hybrid (or PBH) network, which is a superposition of a perceptron and a small backpropagation network. In order to comprehensively investigate the performances of neural networks, we examined five different types of neural networks: Perceptron, BP, PBH, Fuzzy ART MAP and RBF.

The perceptron [9], Fig. 5(a), is the simplest form of a

neural network used for the classification of *linearly separable* patterns. It consists of a single neuron with adjustable synapses and threshold. Although our data sets will not, in general, be linearly separable, we are using the perceptron as a baseline to measure the performances of other neural networks.

The Backpropagation network [9], or BP, Fig. 5(b), is a multiplayer feedforward network, which contains an input layer, one or more hidden layers, and an output layer. BPs have strong generalization capabilities and have been applied successfully to solve a variety of difficult and diverse problems. Here we tested BP networks with the number of hidden neurons ranging from 2 to 8.

The Perceptron-backpropagation hybrid network [6], or PBH, Fig. 5(c), is a superposition of a perceptron and a small backpropagation network. PBH networks are capable of exploring both linear and nonlinear correlations between the input stimulus vectors and the output values. We tested PBH networks where the number of hidden neurons ranged from 1 to 8.

The Fuzzy ARTMAP [10] in its most general form is a system of two Fuzzy ART networks ART_a and ART_b whose F2 layers are connected by a subsystem referred to as a "match tracking system". We are using a simplified version of Fuzzy ARTMAP [11], Fig. 5(d), which is implemented for classification problems. We tested ARTMAP networks with the number of category neurons ranging from 2 to 8.

The Radial-basis function network [9], or RBF, Fig. 5(e), involves three entirely different layers. The input layer is made up of source nodes. The second layer is a hidden layer of high enough dimension, which serves a different purpose from that in a BP network. The output layer supplies the response of the network to the activation patterns applied to the input layer. We tested RBF networks with hidden neurons ranging from 2 to 8.

V. The Simulation Environment

We used a virtual network using simulation tools to generate attack scenarios. The experimental testbed that we built using OPNET, a powerful network simulation facility, is shown in Fig. 6. The testbed is a 10-BaseX LAN that consists of 11 workstations and 1 server.

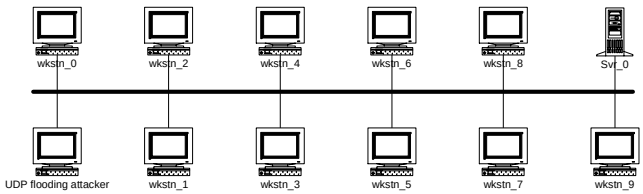

Fig. 6 Simulation Testbed

We simulated the *udp flooding attack* within the testbed. To extensively test the performances of neural networks, we ran four independent scenarios with different typical traffic loads and attack traffic. For each simulation scenario, we collected 10,000 records of network traffic. We divided these data into two separate sets, one set of 6000 records for training and the other of 4000 records for testing. In each scenario, the system was trained for 100 epochs.

VI. Results On Neural Networks

We evaluated the performances of each of the neural networks based on the *mean squared root errors* and the *misclassification rates* of the outputs. The misclassification rate is defined as the percentage of the inputs that are misclassified by neural networks during one epoch, which includes both *false positive* and *false negative* misclassifications.

Table 1 lists the traffic loads of the four simulation scenarios we ran for neural network testing.

Table 1 Traffic Loads of The Four Simulation Scenarios

	Background Traffic	Attack Traffic
Scenario 1	600kbps	50kbps
Scenario 2	600kbps	100kbps
Scenario 3	2Mbps	50kbps
Scenario 4	2Mbps	100kbps

In the rest of this section, we will present and analyze the simulation results of the neural networks one by one.

1) Perceptron

The mean squared root errors and the misclassification rates of the perceptrons within the four simulation scenarios are tabulated in Table 2.

Table 2 The simulation results of perceptrons

	MSR Error	Misclassification rate
Scenario 1	0.686	0.167
Scenario 2	0.716	0.202
Scenario 3	0.739	0.234
Scenario 4	0.635	0.119

We can see that the perceptron performed poorly in all the four scenarios: Mean squared root errors are between 0.6 and 0.7; and misclassification rates are between 0.1 and 0.2. Both the MSR errors and the misclassification rates are unacceptably high for an IDS.

2) Fuzzy ARTMAP and RBF

The results of Fuzzy ARTMAP and RBF nets are shown in Fig. 7 and Fig. 8. The x-axis values of the figures represent the number of category neurons in Fuzzy ARTMAP and the hidden neurons in RBF. The y-axis values represent the lowest Mean Squared Root Errors and the lowest Misclassification Rates that these neural nets achieved within the 100 epochs.

ISBN 0-7803-9814-9/$10.00 © 2001 IEEE

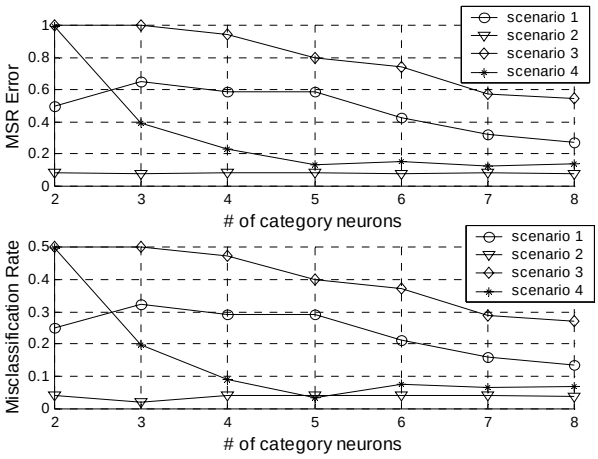

Fig. 7 Results of Fuzzy ARTMAP

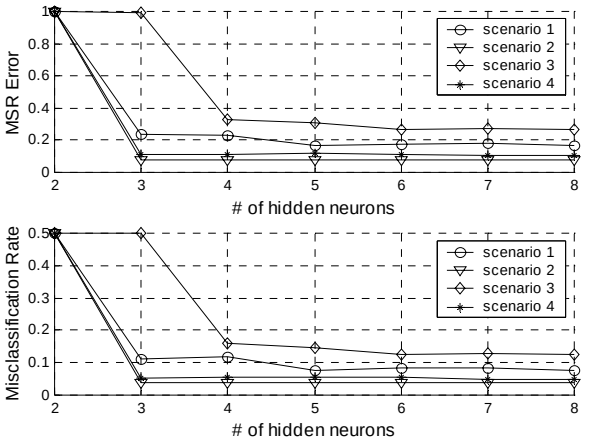

Fig. 8 Results of RBF

From the above figures, we can see that, as the number of hidden neurons increases, the performances of both ARTMAP and RBF networks improve. In most of the cases, both of them outperformed the perceptron.

3) BP and PBH

The results of BP nets are illustrated in Fig. 9 and Fig. 10. The figures indicate that BP and PBH networks have similar performances, and that both neural networks consistently perform better than the other three types of neural networks. The curves in these figures are flat: the MSR errors and misclassification rates do not decrease as the number of hidden neurons increases. We believe the reason is that, because we only deployed one attacking technique, *UDP flooding attack*, in our simulations, our data sets are too simple and not challenging enough for BP and PBH.

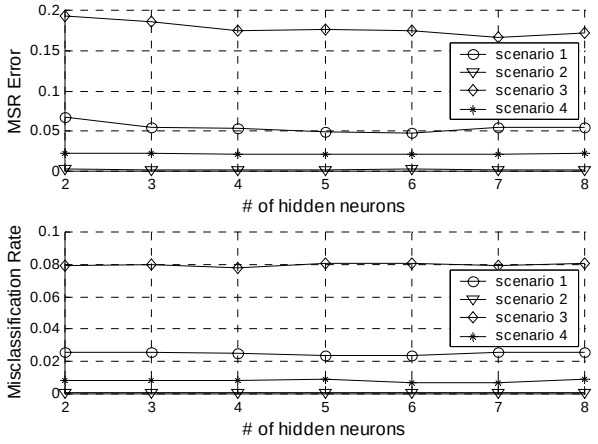

Fig. 9 Results of BP

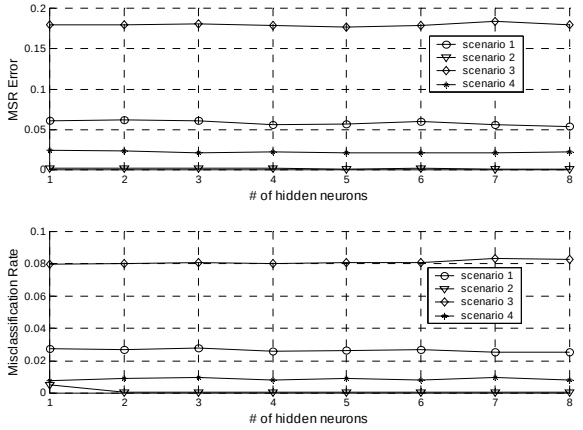

Fig. 10 Results of PBH

In the future, we will incorporate more types of Denial-of-Service attacking techniques into our simulation, thus providing additional tests, and possibly greater challenges, for the neural networks under consideration.

VII. Stress Testing The HIDE System

In this section, we will stress test the sensitivity, and thus effectiveness, of HIDE. We simulated various *UDP flooding attack* scenarios using the test bed we introduced in section 5. The traffic loads of the simulations we ran for stress testing are specified in Table 3. From section 6, we can see that BP and PBH performed the best among the five neural networks tested, and that, for these two neural networks, increment in the number of hidden neurons did little to help with their performances. Therefore, the neural network that we chose for stress testing was a BP network with 2 hidden neurons.

TABLE 3 THE TRAFFIC LOADS FOR STRESS TESTING

Background traffic	Attack Traffic
600 Kbps	10Kbps, 20Kbps, 30Kbps, 40Kbps, 50Kbps, 70Kbps, 100Kbps, 200Kbps
2Mbps	10Kbps, 50Kbps, 150Kbps, 200Kbps

Through the simulations, we expect to see the changes in *the*

Mean Squared Root Errors and *the Misclassification Rates* of the system as functions of the background traffic and attack traffic volumes.

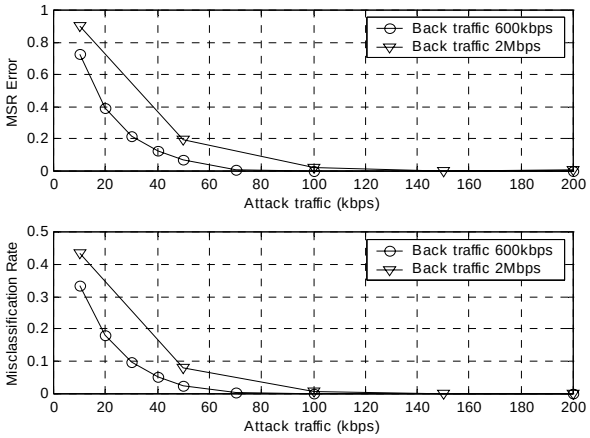

Fig. 11 The results of Stress Testing

The results of stress testing are shown in Fig. 11. From the figure, we can see that both the MSR errors and the misclassification rates decrease as the attack level increases. This is because that traffic patterns of higher-volume attacks yield greater differences from the reference model than the differences created by lower-volume attacks. We can also notice that, for a certain attack level, the performance for the 600Kbps background traffic is consistently better than that of the 2Mbps background. One plausible explanation is that intruders can cover their behavior patterns from being detected under high background traffic scenarios more easily.

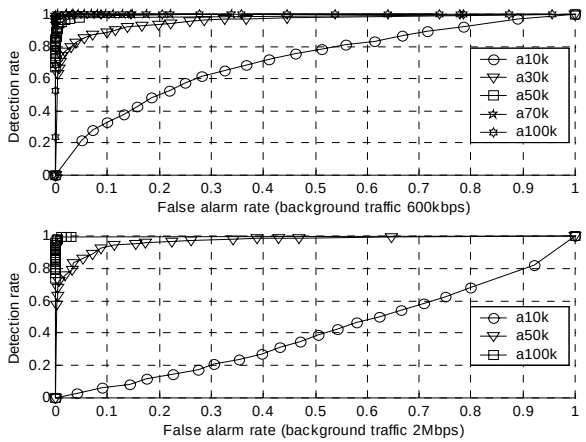

Fig. 12 ROC Curves

Fig. 12 shows the ROC (Receiver Operating Characteristic) curves of some selected simulation scenarios. The x-axis of the figure is the false alarm rate, which is the rate of the typical traffic events being classified as intrusions; the y-axis of the figure is the detection rate, which is calculated as the ratio between the number of correctly detected intrusions and the total number of intrusions. For each curve, the point at the upper left corner represents the optimal detection with high detection rate and low false alarm rate. From the figure, we can observe the same tendency as in Fig. 11: The detection performance improves as the attack intensity increases. In fact, we can see that, when the attack level is 70Kbps for 600Kbps background traffic and 100Kbps for 2Mbps background traffic, the system performance approaches the optimum.

VIII. Conclusions

In this paper, we introduced the framework of HIDE, a hierarchical anomaly network intrusion detection system using statistical preprocessing and neural network classification. We described our experiments using five different neural networks. The results showed that BP and PBH nets outperform Perceptron, Fuzzy ARTMAP and RBF. Thus, classification capabilities of BP and PBH are more desirable for statistical anomaly intrusion detection systems. We also presented some of the results of stress testing. The results indicate that the system is very efficient. It can reliably detect UDP flooding attacks with traffic intensity as low as five to ten percent of the background intensity.

Acknowledgements

Our research was partially supported by a Phase I SBIR contract with US Army. We would also like to thank OPNET Technologies, Inc.™, for providing support for the OPNET simulation software.

References:

[1] G. Vigna, R. A. Kemmerer, "NetSTAT: a network-based Intrusion Detection Approach", *Proceedings of 14th Annual Computer Security Applications Conference*, 1998, pp. 25 –34.

[2] W. Lee, S. J. Stolfo, K. Mok, "A Data Mining Framework for Building Intrusion Detection Models", *Proceedings of 1999 IEEE Symposium of Security and Privacy*, pp. 120-132.

[3] A. Valdes, D. Anderson, "Statistical Methods for Computer Usage Anomaly Detection Using NIDES", *Technical report*, SRI International, January 1995.

[4] J. M. Bonifacio, et al., "Neural Networks Applied in Intrusion Detection System", *IEEE*, 1998, pp. 205-210

[5] H. S. Javitz, A. Valdes, "The NIDES Statistical Component: Description and Justification", *Technical report*, SRI International, March 1993.

[6] R. M. Dillon, C. N. Manikopoulos, "Neural Net Nonlinear Prediction for Speech Data", *IEEE Electronics Letters*, Vol. 27, Issue 10, May 1991, pp. 824-826.

[7] A.K. Ghosh, J. Wanken, F. Charron, "Detecting Anomalous and Unknown Intrusions Against Programs", *Proceedings of IEEE 14th Annual Computer Security Applications Conference*, 1998, pp. 259 – 267

[8] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, J. Ucles, "A Hierarchical Anomaly Network Intrusion Detection System Using Neural Network Classification", *CD-ROM Proceedings of 2001 WSES International Conference on: Neural Networks and Applications (NNA '01)*, Feb. 2001

[9] Simon Haykin, *Neural Network A Comprehensive Foundation*, Macmillan College Publishing Company, 1994

[10] G.A. Carpenter, et al, "Fuzzy ARTMAP: An adaptive resonance architecture for incremental learning of analog maps", *International Joint Conference on Neural Networks*, June 1992

[11] NeuraWare Inc., *Neural Computing A Technology Handbook for NeuralWorks Professional II/PLUS and Neural Works Explorer*, NeuralWare Inc., 1998

[12] Joao B.D. Cabrera, B. Bavichandran, R.K. Mehra, "Statistical Traffic Modeling for Network Intrusion Detection", *Proceedings of 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication systems*, Aug. 2000, pp. 466-473