Differential Equations and Dynamical Systems,
Vol. 10, Nos. 1 & 2, January & April 2002, pp. 201–214.

Computer Network Intrusion Detection: A Comparison of Neural Networks Methods

Vu N. P. Dao and V. Rao Vemuri

Abstract

This paper demonstrates that artificial neural network (ANN) techniques can be used in detecting intruders logging onto a computer network when computer users are profiled accurately. The paper compares the performance of the gradient descent back propagation (BP) with momentum, the conjugate gradient BP, and the quasi-Newton methods in detecting intruders using synthetically generated authorized user and intruder profiles.

1. Introduction

Computer security issues are becoming headline grabbing items. According to one news report, our computer systems are being attacked hundreds of times-each day. Internet security, and our ability to enforce it, is likely to determine the extent to which we can harness cyberspace for conducting business-and indeed our own lives. Internet security is a catchall term and can mean different things to different people. Network security, server security, user authentication, user authorization, data integrity, data privacy and many more come under the general rubric of Internet security.

[0]AMS (MOS) 2001 Subject Classifications 82C32, 68M10, 94CXX.
[0]

According to one taxonomy [1], Internet security issues fall into two broad categories: preventive and enabling. The preventive aspect of Internet security has so far captured much attention primarily due to the activities of the headline-grabbing malicious hackers, network intruders, and other black-hat denizens of the cyberspace. Prominent among the defenses one can provide in this area are well-designed routers, firewalls, intrusion detection systems and anti-virus software. The enabling aspect of Internet security addresses the problem from a different perspective. Instead of using technologies like private lines, X.25 and frame relay circuits over a private provider, here one focuses on methods of securely transporting sensitive data over the Internet. Proponents of this approach offer the Virtual Private Network (VPN) as a potential solution. VPN offers secure site-to-site connectivity using private (i.e., using data encryption and authentication technology) and virtual circuits (i.e., no physical end-to-end connectivity).

In any on-line activity, whether it is e-mail, file transfer, web surfing, downloading, or commercial trade, a number of component activities-say, the web server, data transaction protocol, client side software and operating systems software-come into play. A failure or intrusion at any one of these levels may compromise the entire transaction. Data transaction security protocols and encryption methods provide security to data in transit; they do not provide security on either end of the transaction. A flaw in the web server software, for instance, may allow an intruder access to the complete transaction records without breaking any encrypted text. Therefore, intrusion detection continues to be an important problem in the overall context of network security.

The focus of this paper, therefore, is on intrusion detection as a preventive measure. The objective is to explore the potential of artificial neural networks as on-line, real-time tools for detecting intrusive activities as they occur. Toward this goal, the objective is to construct a model that captures a set of user attributes and determine if that set of user attributes belongs to the class of authorized users or to that of intruders. The premise is that any user logging onto a computer can be uniquely characterized by a set of attributes and those attributes can be monitored, measured and quantified [2].

The relevance of ANN's in intrusion detection becomes apparent when one views the intrusion detection problem as a pattern classification problem. Using profiles of authorized computer users [2], one can train an ANN to recognize the authorized users in the incoming traffic, thus separating authorized traffic from intrusion traffic. More importantly, many of the current intrusion detection techniques cannot detect new and novel attack patterns. It is precisely in this area that the learning and generalization capabilities of ANN's can be exploited.

The rest of the paper is organized as follows. Section 2 defines and formulates the problem of using neural network for intrusion detection. Section 3 gives an overview of the different neural network methods used in detecting the intruders. Section 4 covers the generation of the training and test data sets. Section 5 summarizes the results and compares the performances of the various neural network methods. Section 6 summarizes the research findings. Finally, Section 7 gives a brief conclusion of the research and points to new research areas.

2. Mathematical Formulation of the Problem

Mathematically speaking, the problem can be stated as follows. Given a *training data* set S, the goal is to establish a mapping from any given input vector x to an output class d.

$$S = \{(x_i, y_i) : x_i \in R^p, d\}$$

Here x = input vector, y = corresponding output, and d = the desired output.

From a modeling perspective [3], the objective is to seek a model that provides the best fit to the given training data and the best prediction capability with future observed data (also known as *test data* set), while minimizing model complexity.

The above objective is typically accomplished by building a model (i. e., the neural net, or equivalently, the mapping function) and train it prior to using that model for intrusion detection. During the training phase, a performance index, defined in terms of the error, e is minimized.

$$e = d - y$$

3. The Back Propagation Class of Training Methods

The back propagation method is a popular technique to train multi-layer feed-forward neural networks in a supervised manner. This method is well known and details can be found in the published literature [4, 5]. What follows is a brief synopsis of the methods and formulas used in this study. In each of these methods, the term "training" refers to the systematic procedure used to adjust the "weights" in a weighted sum of inputs that is responsible for activating a neuron. The sigmoidal activation [5] is used in this study. The back propagation methods are the gradient descent with momentum (GDM), the conjugate gradient descent (CGP) and the quasi Newton (BFGS).

3.1. GD with Momentum BP (GDM)

The simplest back propagation method is the gradient descent method. In the gradient descent method, the network weights are updated in a manner that the value of the performance index decreases most rapidly. That is, the new weight vector w_{k+1} is obtained according to:

$$\mathrm{w}_{k+1} = \mathrm{w}_k - \alpha \mathrm{g}_k$$

where the parameter α is the learning rate and g_k is the gradient of the error with respect to the weight vector. The negative sign indicates that the new weight vector w_{k+1} is moving in a direction opposite to that of the gradient.

However, the gradient descent method is slow to converge. Gradient descent with momentum (GDM), a modified version of the gradient descent, is faster. In GDM, the new weight vector w_{k+1} is defined by

$$\mathrm{w}_{k+1} = \mathrm{w}_k - \alpha \mathrm{g}_k + \mu \mathrm{w}_{k-1}$$

where μ is the so-called momentum term. The GDM allows a network to respond not only to the local gradient, but also to trends in the error surface. Without momentum a network may get stuck in a shallow local minimum. With momentum a network can slide through such a minimum [5, 6]. Momentum can be added to BP method learning by making weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the gradient descent BP rule. The magnitude of the effect that the last weight change is allowed to have is mediated by the momentum term, μ, which can be any number between 0 and 1. When the momentum term is 0, the weight change is based solely on the gradient. When the momentum term is 1 the new weight change is set to equal the last weight change and the gradient information is simply ignored.

3.2. Conjugate Gradient BP (CGP)

The basic gradient descent BP algorithm adjusts the weights in the steepest descent direction. This is the direction in which the performance function is decreasing most rapidly. Although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than the steepest descent method. In the conjugate gradient algorithms the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size which will minimize the performance function along that line. The version of conjugate gradient method used here is due to Polak and Ribiere (CGP) [6, 7]. The search direction at each iteration is determined by updating the weight vector as:

$$\begin{aligned} \mathrm{w}_{k+1} &= \mathrm{w}_k + \alpha\, \mathrm{p}_k \\ \mathrm{p}_k &= -\mathrm{g}_k + \beta_k \mathrm{p}_{k-1} \\ \beta_k &= \frac{\Delta \mathrm{g}_{k-1}^T \mathrm{g}_k}{\mathrm{g}_{k-1}^T \mathrm{g}_{k-1}} \\ \Delta \mathrm{g}_{k-1}^T &= \mathrm{g}_k^T - \mathrm{g}_{k-1}^T \end{aligned}$$

3.3. Quasi-Newton BP (BFGS)

Newton's method is an alternative to the conjugate gradient methods for fast optimization. Newton's method often converges faster than conjugate gradient methods. The weight update for the Newton's method is:

$$\mathrm{w}_{k+1} = \mathrm{w}_k - \mathrm{A}_k^{-1} \mathrm{g}_k$$

where A_k is the Hessian matrix of the performance index at the current values of the weights and biases. When A_k is large, it is complex and time consuming to compute w_{k+1}. Fortunately, there is a class of algorithms based on the works of Broyden, Fletcher, Goldfarb, and Shanno (BFGS) [5, 8] that are based on Newton's method but which don't require intensive calculation. This new class of method is called quasi-Newton method. The new weight w_{k+1} is computed as a function of the gradient and the current weight w_k.

4. Creation of Training and Test Data Sets

Creation of training data is central to the success of any neural networks method. One of the challenges is in creating intrusion scenarios. Earlier works have shown that computer users exhibited unique characteristics, and that computer users can be profiled accurately based on their attributes [2, 10]. By analyzing the data logs collected from the Computer Security Laboratory at the University of California, Davis, computer users were profiled based on four attributes [2]. These attributes were: (i) the command set used by the user, (ii) the login host, (iii) the time of login, and (iv) the time required to execute each of the commands entered (i.e., the CPU time).

Figures 1 and 2 illustrate how two attributes-the command set and time of login-can be used to identify a user. On the x-axis of Figures. 1a and 2a are the 100 most common commands used by all profiled users. The y-axis represents the time in weeks, and the z-axis represents the percentage of each command used in relation to the 100 command set. In Figures. 1b and 2b, the x-axis represents the login time, the y-axis represents the time in weeks, and the z-axis represents the number of commands each user entered. Figure 1 summarizes the command set and the login time of User 1 on host A; similarly Figure 2 summarizes the command set and login time of User 2 on host C. From Figures. 1 and 2, one can see that User 1 and User 2 used different command sets and at different frequencies during the time of login. If only two attributes are capable of showing this level of discrimination, it is reasonable to expect that the use of more attributes will have better discrimination capability.

To test this hypothesis further, two data sets for training and testing the neural networks were created. For simplicity in testing, these two data sets were created on the same host. Each of these two data sets now has three varying attributes-command set, time of login and CPU time, and one fixed attribute-host. The characteristic of the data set and how they were used for testing are covered next.

4.1. Characteristics of Data Sets

The objectives of creating data sets for training and testing were twofold. First is to test the capability of the neural networks to classify data from authorized users and intruders that are linearly separated (i.e., two distinctive classes).

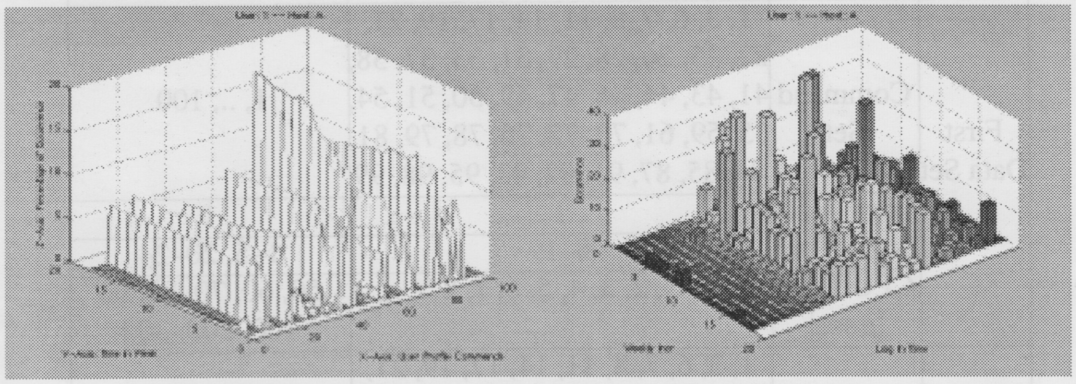

Figure 1. User 1 Characteristic on Host A-(a)Command Set (b)Login Time.

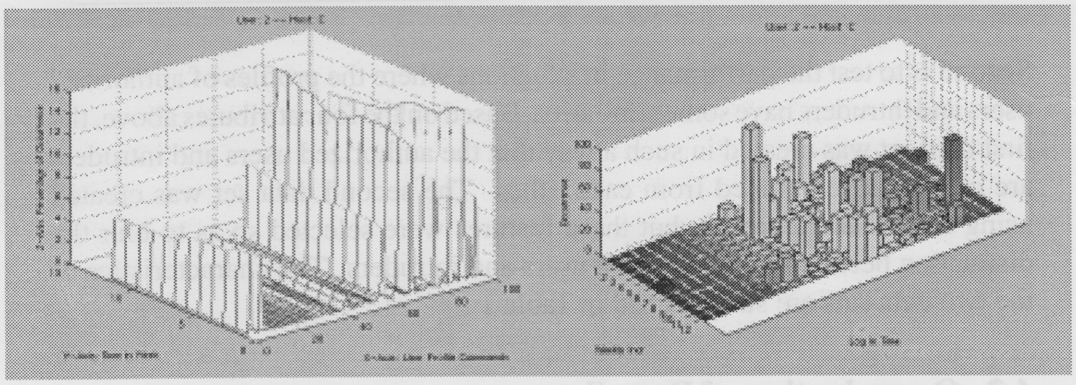

Figure 2. User 2 Characteristic on Host C-(a)Command Set (b)Login Time.

Table 1. User Profile of Characteristics of Generated Test Data

	User Features	**Authorized Users**	**Intruders**
First Data Set	Command Set	1, 3, 6, 7, 8, 11, 14, 17, 19, 21, 22, 25, 26, 28, 29, 31, 33, 37, 38, 41, 43, 44, 46, 47, 49, 50, 51, 54, 55, 59, 61, 70, 72, 76, 78, 79, 81, 84, 85, 87, 91, 92, 93, 95, 98, 99	1, .., 100
	Login Host	1	1
	Login Time	6–19	1–5, 20–24
	CPU Time [μsec]	1, 2, 3, 4, 5, 6, 7, 8	50, 60, 70, 80, 90 100, 200, 300
Second Data Set	Command Set	1, 3, 6, 7, 8, 11, 14, 17, 19, 21, 22, 25, 26, 28, 29, 31, 33, 37, 38, 41, 43, 44, 46, 47, 49, 50, 51, 54, 55, 59, 61, 70, 72, 76, 78, 79, 81, 84, 85, 87, 91, 92, 93, 95, 98, 99	1, .., 100
	Login Host	1	1
	Login Time	6–19	1–7, 18–24
	CPU Time [μsec]	1, 2, 3, 4, 5, 6, 7, 8	5, 6, 7, 8, 10, 20, 40, 50, 60, 70, 80, 90, 100, 200, 300

Second is to test the performance in situations where the profiles of authorized users and intruders have some similarity. Based on the four attributes above, the first data set was created in such a way that the authorized users and intruders are clearly differentiated from each other. The second data set was created using the same feature set, but these features were selected so as to blur the distinction between the authorized users and intruders. The characteristics of the two data sets are summarized in Table 1.

4.2. Organization of Data Sets

The next step is to organize the data set into a suitable format for training and testing. The generated data was organized into two parts. The first part is for training. Each training input sample came with a desired output. Here, ninety percent (90%) of the input data was generated as authorized traffic

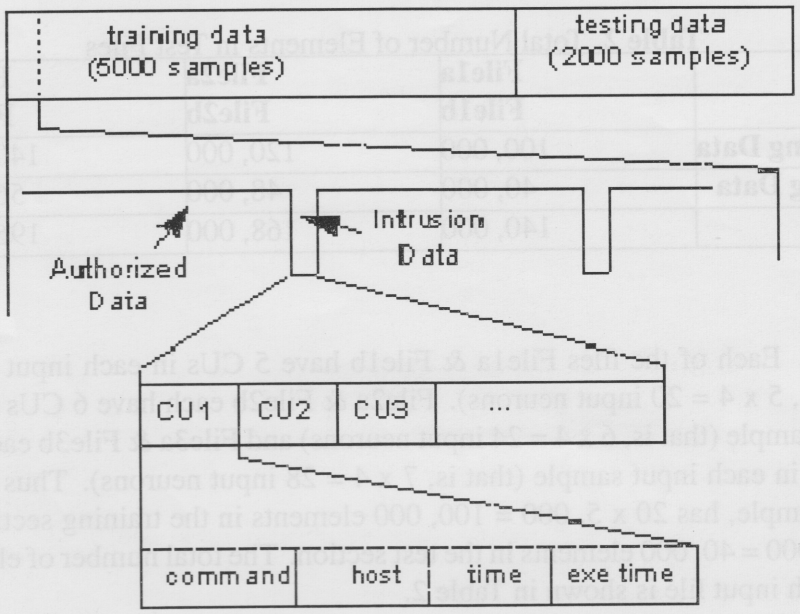

Figure 3. Construction of the Generated Data.

and ten percent (10%) as intrusion traffic. The second part is for testing the performance of the methods. In this part, ninety eight percent (98%) of the traffic were generated to be authorized traffic and two percent (2%) of the traffic to be intrusion traffic.

In both parts of the training and test data section, several bursts of intrusion data are inserted into the authorized data stream. Each of the generated input data files has 7000 samples. The first 5000 samples are the training data, and the next 2000 samples are the test data. Authorized traffic is designated *Class Positive* and unauthorized traffic as *Class Negative*. Figure 3 illustrates the structure of the generated data files.

In Figure 3, an input sample is defined as a unit of data being fed into the neural network at one time; one sample can consist of many command units (CUs). A CU is defined as a set of four elements-the UNIX command, the login host, the time of login, and the execution time of the command (i.e., CPU time). The objective is to test the neural networks for detecting intrusion traffic with the fewest number of CUs. With that objective, three test files for the first data set and three test files for the second data set are generated. These six files are identified as follows: File1a, File1b, File2a, File2b, File3a and

Table 2. Total Number of Elements in Test Files

	File1a File1b	**File2a File2b**	**File3a File3b**
Training Data	100, 000	120, 000	140, 000
Testing Data	40, 000	48, 000	56, 000
Total	140, 000	168, 000	196, 000

File3b. Each of the files File1a & File1b have 5 CUs in each input sample (that is, 5 x 4 = 20 input neurons). File2a & File2b each have 6 CUs in each input sample (that is, 6 x 4 = 24 input neurons) and File3a & File3b each have 7 CUs in each input sample (that is, 7 x 4 = 28 input neurons). Thus File1a, for example, has 20 x 5, 000 = 100, 000 elements in the training section and 20 x 2000 = 40, 000 elements in the test section. The total number of elements for each input file is shown in Table 2.

5. Simulation Results and Performance Comparison

Simulation results are summarized in Table 3 and Table 4. In these tables, topology specifies the neural network architecture. For example, a topology of {20, 10, 1} indicates 20 input neurons, 10 hidden neurons, and 1 output neuron. The parameters α and μ indicate the learning rate and the momentum constants of the gradient descent BP with momentum. The quantity e is the mean of the square error (MSE) of the difference between the actual output y and the desired output d. An epoch is one complete presentation of the training data. 'FP' and 'FN' are the false positive and false negative rates in classifying users. A false positive is classifying an authorized user as an intruder; conversely, a false negative is classifying an intruder as an authorized user. Each method terminates when any of the following condition occurs: (i) MSE $\leq \epsilon$, (ii) Epochs = 500, and (iii) when the gradient undergoes negligible change from one epoch to the next (i.e., typically exp(-10) in the simulation).

Table 3. Summary of Simulation Results for the First Data Set

	File1a Input		**File2a Input**		**File3a Input**	
	Parameters	Results	Parameters	Results	Parameters	Results
GDM	{20, 17, 1} $\alpha = 0.1$ $\mu = 0.75$	FP=N/A FN=N/A 500 epochs $\epsilon = N/A$	{24, 14, 1} $\alpha = 0.05$ $\mu = 0.75$	FP=0 FN=0 140 epochs $\epsilon = exp - 5$	{28, 17, 1} $\alpha = 0.05$ $\mu = 0.70$	FP=0 FN=0 500 epochs $\epsilon = 0.07$
CGP	{20, 13, 1}	FP=0 FN=0 90 epochs $\epsilon = exp - 5$	{24, 10, 1}	FP=0 FN=0 50 epochs $\epsilon = exp - 5$	{28, 16, 1}	FP=0 FN=0 40 epochs $\epsilon = exp - 5$
BFGS	{20, 13, 1}	FP=0 FN=0 45 epochs $\epsilon = exp - 5$	{24, 11, 1}	FP=0 FN=0 60 epochs $\epsilon = exp - 5$	{28, 11, 1}	FP=0 FN=0 40 epochs $\epsilon = exp - 5$

5.1. Simulation Results

Table 3 summarizes results obtained when the first data set was applied as input. Since the gradient descent with momentum (GDM) method did not perform as well as the other two methods (CGP and BFGS), it was excluded from being used to test the second data set. Table 4 summarizes results obtained when the second data set was used as input for the conjugate gradient and quasi Newton methods.

Table 4. Summary of Simulation Results for the Second Data Set. (* Methods stop due to gradient being too small)

	File1a Input		**File2a Input**		**File3a Input**	
	Parameters	Results	Parameters	Results	Parameters	Results
CGP	{20, 15, 1}	FP=0.04% FN=15% 500 epochs $\epsilon = 0.01$	{24, 14, 1}	FP=0.01% FN=17.5% 500 epochs $\epsilon = 0.17$	{28, 16, 1}	FP=0 FN=17.5% 450 epochs* $\epsilon = 0.18$
BFGS	{20, 17, 1}	FP=015% FN=12.5% 500 epochs $\epsilon = 0.01$	{24, 15, 1}	FP=0.15% FN=5% 500 epochs $\epsilon = 0.01$	{28, 16, 1}	FP=0.05% FN=25% 100 epochs* $\epsilon = 0.18\%$

5.2. Performance Comparison

After the three described neural networks were trained and tested with the first data set {i.e., File1a, File2a, File3a} for the first case and the second data set {i.e., File1b, File2b, File3b} for the second case, the following observations were made.

First, the gradient descent with momentum method did not perform well as the Conjugate Gradient and quasi Newton methods. Furthermore, it was time consuming to tune the learning rate, α, and the momentum parameter, μ, for this method to operate properly. For instance, when the input was File1a, the GDM method was not able to classify the intrusion traffic from the authorized traffic, however it was able to do the classification correctly when the input file was File2a or File3a. When the input file was File2a, GDM requires 140 epochs to converge compared to only 50 epochs for CGP and 60 epochs for BFGS. When the input file was File3a, GDM needed 500 epochs to converge to a 75% false negative error compared to 40 epochs and no false negative for both CGP and BFGS methods. These results indicate that both the CGP and the BFGS methods were superior in classifying authorized users from intruders while maintaining good false negative values and good convergence properties compared to the GDM method.

Second, the conjugate gradient and the quasi Newton methods exhibited roughly the same performance in terms of classification. Both of these methods required simpler NN topology (i.e., a topology with fewer hidden layer neurons) and fewer epochs to converge compared to the GDM method.

Third, the number of CUs used in each input sample affected the performance of the classification of the data. Of all the data sets used, File2a and File2b tended to yield the best performance for the three neural networks methods. This leads one to believe that an input sample consisting of 5 CUs (20 elements) might not contain enough information, while 7 CUs (28 elements) input sample might contain too much information, whereas 6 CUs (24 elements) input samples contains the right amount of information for classification of computer users. This issue needs further study.

6. Summary

In this paper, the problem of computer network intrusion detection is posed as a classification problem. Three different BP neural network methods were applied to solve this problem. For training and testing purposes, two synthetic test data sets were created. From the first test data set, it was shown that the three neural networks methods were capable of classifying authorized users from intruders. The conjugate gradient and the quasi Newton methods yielded superior performance than the gradient descent with momentum method. With the second test data set, the conjugate gradient and quasi Newton methods performed best when each input sample size is 6 CUs, or equivalently 24 elements long.

7. Conclusion and Future Work

From the preliminary results shown in this paper it appears that neural network techniques hold some potential in intrusion detection. The training data and test data used in this study were synthetically generated. Using this generated data, it was shown that the conjugate gradient and the quasi Newton can successfully detect intruders logging into a computer network. These two methods only required an input sample of 6 consecutive CUs from the intruder to classify that the current user is indeed an intruder.

Several issues remain to be investigated. First, it is necessary to evaluate the performance of neural networks with data sets captured by monitoring real traffic. Second, it is also necessary to establish the feature set that best describes users and intruders. The three used in this paper, namely command set, login time, and CPU usage, were selected because they appeared to be reasonable choices after a cursory examination of a data set available at the Security Laboratory of the University of California at Davis. Third, it is necessary to characterize the drifting patterns among authorized users and develop an incremental training procedure. Finally, besides the BP methods, there are other neural networks methods like the radial basis function (RBF) that could be implemented. These issues are being addressed at this time.

Acknowledgment

This work was performed in part under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48, and the Center for Digital Security, University of California, Davis.

References

[1] K. M. Phaltankar, Practical Guide for Implementing Secure Intranets and Extranets, Artech, 1999.

[2] V. Dao, R. Vemuri, and S. Templeton, Profiling Users in the UNIX OS Environment, International Computer Science Conventions Conference, University of Wollongong, Australia, 2000.

[3] M. Shin and C. Park, A Radial Basis Function Approach to Pattern Recognition and Its Applications, ETRI Journal, South Korea, 22(2000).

[4] J. Principe, N. Euliano, and W. Lefebvre, "Neural and Adaptive System-Fundamentals Through Simulations," Wiley, 2000.

[5] S. Haykin, "Neural Networks-A Comprehensive Foundation," 2nd Ed., Prentice Hall, 2000.

[6] MATLAB, "Neural Network Toolbox, Version 3," The Math Works Inc., 1998.

[7] E. Dennis and R. Schnabel, "Numerical Methods for Unconstrained Optimization and Nonlinear Equations," Englewood Cliffs, NJ: Prentice-Hall, 1983.

[8] M. Hagen, H. Demuth, and M. Beale, "Neural Network Design," Boston, MA., PWS Publishing, 1996.

[9] T. Lane and C. Brodley, Temporal Sequence Learning and Data Reduction for Anomaly Detection, Proceedings of the Fifth ACM Conference on Computer and Communications Security, 1998, 150–158.

[10] D. Denning, "An Intrusion Detection Model," IEEE Transactions on Software Engineering, 1987.

Vu N. P. Dao: Lawrence Livermore National Laboratory 7000 East Avenue, Livermore, CA 94550, USA.

V. Rao Vemuri: Department of Applied Science, University of California, Davis P. O. Box 808, L-794, Livermore, CA 94550, USA.