

Performance Analysis of Soft Computing Based Anomaly Detectors

N. Srinivasan¹ and V. Vaidehi²

(Corresponding author: N. Srinivasan)

Department of Information Technology, Madras Institute of Technology¹

Anna University, Chennai, India 600044 (Email: sreeni@gmail.com)

Department of Electronics Engineering, Madras Institute of Technology²

Anna University, Chennai, India 600044

(Received June 27, 2007; revised Nov. 7, 2007; and accepted Mar. 3, 2008)

Abstract

Anomaly detectors have become a necessary component of the computer and information security framework. Some of the numerous drawbacks experienced by the current Anomaly detectors are large number of false positive and false negative alarms, difficulty in processing huge amount of traffic in real time, inadequacy in novel attack recognition and non-scalability. Consequently their efficacy in protecting against anomalies is limited. The use of soft computing techniques like Genetic algorithms, Neural networks and Fuzzy logic in implementing Anomaly detection is perused in this paper. Additionally, a few novel approaches for the detection of anomalies by identifying user actions and network traffic that might compromise a system's secure state, is also proposed. A potential solution to the problem has been contemplated, by comparing the performance of these systems based on various criteria. Characterization of the behavior of a single user (Host based) or a network (Network based) and recognition of anomalies through observation of deviation from normal behavior patterns are conducted to arrive at the solution. The implementations of Genetic algorithm based Anomaly detection system (GAAD), Neural network based Anomaly detection system (NNAD) and Fuzzy Logic based system (FLAD) are reported. Interesting conclusions are deduced from an exhaustive evaluation and comparison of the performance of these systems enabling an administrator to choose the best solution for a given scenario.

Keywords: Anomaly detection, false alarm rate, fuzzy logic, genetic algorithm, neural network, soft computing

1 Introduction

With the massive expansion of computer networks, security in computing environments has assumed an imperative cause for concern. Many modern systems are im-

paired by a diverse set of vulnerabilities due to lack of properly implemented security services and are therefore compromised easily. Anomaly is defined as a set of actions, which guide the transition of a computing system from a normal (secured) state to a compromised state. The definition of anomaly detection as proposed by Bace [4] is, "the process of intelligently monitoring the events occurring in a computer system or network, and analyzing them for signs of violations of the security policy". Activities that compromise the integrity, confidentiality and availability of, or assist in the evasion of a security mechanism in a host or network are monitored.

Anomaly detectors can be classified based on the functional characteristics of the detection methods as Knowledge based and Behavior based [3]. Storage of previously observed attack patterns in a knowledge base is essential for Knowledge based intrusion detection or misuse detection. A misuse is reported if any of the users' behavior matches the stored patterns. One of the main disadvantages of this system is its incapability to detect new vulnerabilities. On encountering a new attack, difficulty in updating the stored patterns is experienced. A record of the normal behavior of a system or the user is maintained in case of Behavior based anomaly detection, and an anomaly is reported if there is a deviation from the normal behavior. According to Ryan [25], an anomaly has occurred if the observed activity of the user deviates from the expected behavior. False alarms, i.e., indicting legitimate users of anomalous behavior, is the chief drawback of anomaly detection. On the basis of architecture it is broadly classified in to two categories, Host based anomaly detectors [13, 18, 25] and Network based anomaly detectors [10].

Soft Computing is an innovative approach to construct computationally intelligent systems consisting of artificial Neural networks [15, 25], fuzzy inference systems [11, 14], approximate reasoning and derivative free optimization methods such as Genetic algorithms [9, 10, 24] etc. In contrast to the conventional artificial intelligence tech-

niques that focus solely on precision, certainty and rigor, the guiding principle of soft computing is the exploitation of the tolerance for imprecision and uncertainty, in addition to low solution cost, robustness and a better rapport with reality [1, 11, 22]. Soft Computing techniques can be enlisted when 1. an algorithmic solution cannot be formulated, 2. huge amounts of data must be obtained for processing, 3. a need to pick a structure from existing data arises.

A potential solution to the problem of detecting anomalies both in an individual system (host) and in a network is explored and elucidated in this paper. To handle large amount of data implicated in this solution, soft computing techniques are used to characterize the user and network behavior, and to facilitate their comprehensive training and production of accurate results respectively. Analysis of the user command sets is required for the Host based anomaly detection, while the network using the network traffic data is characterized by the Network based detection. Statistical techniques are employed in the analysis of System event streams to find patterns of activities that appear to be abnormal. The soft computing techniques used are Genetic algorithm, Neural networks and Fuzzy Logic. Finally, on comparing the performance and complexities of these systems, the one best suited for a given scenario is singled out. Time and space complexity, detection rate, detection accuracy, false alarm rate, effect of history size on detection accuracy, time required for training, number of runs, command sample size etc, are the indices for evaluation.

The paper is organized into the following sections. The basics of Genetic algorithm and its use in implementing Host based and Network based anomaly detectors is explicated in Section 2. The basic concepts of Apriori algorithm have also been incorporated. A detailed description of the implementation of a Neural network based system is furnished in Section 3. The foundations of Fuzzy logic and design of a Fuzzy Logic based Network anomaly detection is explained in Section 4. Experimentation and their results are discussed in Section 5. The conclusion of the work along with future enhancements is conveyed in Section 6.

2 Genetic Algorithm Based Anomaly Detection

Genetic algorithms (GA) is one of the upcoming fields in computer security and especially in anomaly detection. GA is based on the axiom of Darwin's theory: "Survival of the fittest" [9, 10]. They are capitalized in autonomous learning and decision making, which aid anomaly detection. A simple chromosome-like data-structure is employed by these algorithms and, recombination operators are applied to these structures so as to preserve critical information that provides potential solution to a specific problem [20]. In Genetic algorithms, the properties of the environment are represented by genes (the atomic units).

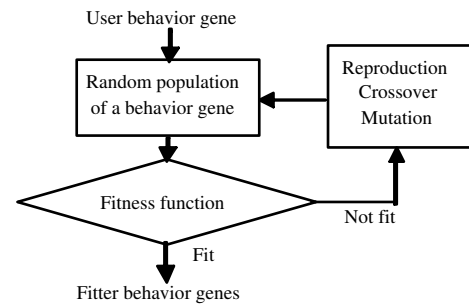


Figure 1: Genetic algorithm in behavior

Processes like mutation and crossover engender the evolution of new genes. The process of evolution continues until genes with the required fitness are found. Genetic algorithm implementation is initiated by the generation of a population of chromosomes. These are subsequently evaluated for reproductive chances, favoring the reproduction of those chromosomes offering a better solution to the objective over the others. In this current generation, fitter behavior genes are generated by applying three genetic operators to the behavior genes. The genetic operators are 1. Reproduction, that selects behavior-genes of relatively higher fitness value from the initial random population and subsequently from the current generation to generate new behavior genes for applying Crossover and Mutation, 2. Crossover, which concatenates parts of two different classes of behavior-gene in the current generation to form new behavior-genes and 3. Mutation, that changes some symbols in the behavior-genes randomly to evolve newer ones. The steps are as shown in Figure 1.

2.1 Host Based GAAD

Genetic Algorithms has been in the literature of Intrusion detection for about half a decade and in the recent past [17, 23], it has been used for classification in other soft computing models. In this work, the robustness and adaptability to changes in the environment of the Genetic algorithms assist in comprehending user behavior in a computing system. Command samples of each user are obtained to characterize the user behavior using a fitness function defined by a 3-tuple $\langle match\ index, entropy\ index, newness\ index \rangle$. A block of user commands of size 'n', where 'n' is a multiple of chromosome size, is adverted in the above mentioned user command sample.

Parameters: Three parameters are considered for characterizing an intruder, as discussed below.

- 1) *Match index:* Match index of a command sample is defined as the ratio of the number of commands that are predicted correctly to its length. The match index parameter is a measure of the regularity in user behavior.

$$Matchindex = \frac{N_1}{l},$$

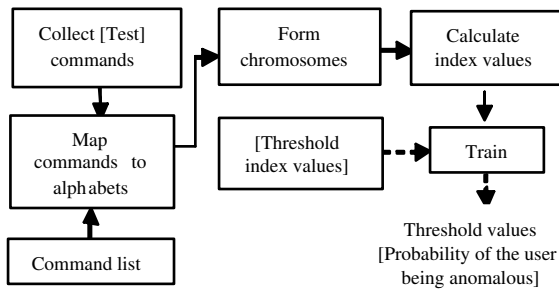


Figure 2: Host based GAAD - Training [Testing]

where, N_1 is the number of commands predicted correctly in the command sample and l is the length of the command sample.

- 2) *Entropy index*: Entropy index is a measure of user behavior dynamics (distribution of commands) in the command sample. Entropy index is given by,

$$Entropyindex = \frac{p(i) \times \log(p(i))}{\log(N_2)},$$

where $p(i)$ is the probability of occurrence of command i in the command sample and N_2 is the number of unique commands in command sample.

- 3) *Newness index*: Newness index of a command sample is defined as the ratio of the count of commands that are not listed in the history of user commands to the cardinality of command sample. The newness index is a measure of the number of commands, which have not occurred earlier.

$$Newnessindex = \frac{N_3}{l},$$

where N_3 is the number of commands in the command sample and in history and l is the length of the command sample.

The procedure involved is as shown in Figure 2.

Training: In the training phase, either the user commands traced in real time or binaries from a data set can be applied. The DARPA Dataset has proven valuable for this purpose. Index values are calculated for every user by training the system and with each user's data assigned to a node. A command list of size 100 is maintained, which is updated as the training is performed with user specific commands. Each command is considered as a gene and these genes together form a chromosome. The training procedure is as follows:

- 1) Collecting the audit data, the commands are executed by a specific user.
- 2) Mapping the commands to the corresponding alphabets.
- 3) Formation of chromosomes.

- 4) Calculation of the fitness function using the 3 tuple value $\langle entropy\ index, match\ index, newness\ index \rangle$. These values are calculated for each chromosome according to the above formula.
- 5) Calculating the average value of the indices for the entire user command set.

Testing: In the testing phase, the index values for the test data are calculated and are matched with the trained using the following steps:

- 1) Mapping the commands to the corresponding alphabets.
- 2) Formation of chromosomes.
- 3) Calculating the indices for each chromosome.
- 4) Comparing the indices with the threshold values and removing the unfit (normal) samples.

The testing algorithm is as follows:

- 1) $M_o = m_u; E_o = e_u; N_o = n_u;$
- 2) $S_a = S;$
- 3) Repeat
 - a. $M_i = M_{i-1} - \alpha; E_i = E_{i-1} + \beta; N_i = N_{i-1} + \gamma;$
 - b. $M_i > x_{jm} \rightarrow S_m = S - S_{xj};$
 - c. $E_i < x_{je} \rightarrow S_e = S - S_{xj};$
 - d. $N_i < x_{jn} \rightarrow S_n = S - S_{xj};$
 - e. $S_a = S_m \cap S_e \cap S_n;$

Until y is not element of S .

- 4) $z = 1 - \frac{(|S_a| - |S|) \times \text{commandsamplesize}}{\text{totalnumberofcommands}},$

where,

S	Set of suspicious user command samples.
S_A	Set of suspicious user command samples in previous iteration.
$ S $	Cardinality of set S .
$ S_A $	Cardinality of set S_A .
x	Typical command sample from S .
i	Index for cycles in the loop where $0 < i \leq$ total number of cycles.
j	Index for cmd samples where $0 < j \leq$ total number of cmd samples.
S_{xj}	j^{th} command sample in user session.
m_u, e_u, n_u	Initial Match, Entropy, Newness index threshold value for user u .
M_u, E_u, N_u	Match, Entropy, Newness index factor for user u .
y	Current command sample of user u .
x_{jm}, x_{je}, x_{jn}	Match, Entropy, Newness index for j^{th} command sample.
M_i, E_i, N_i	Threshold value of Match, Entropy, Newness Index for i^{th} time.
z	Probability of command sample being intrusive.
α, β, γ	Tolerance factors.

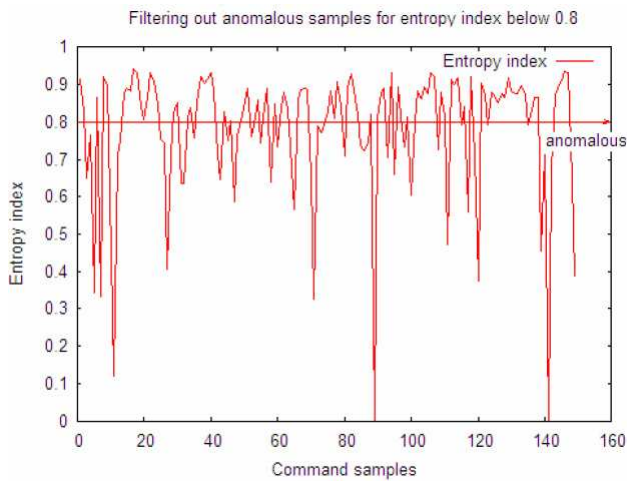


Figure 3: Entropy index of command samples

This algorithm is a modification of the one proposed by Balajinath et al. [5]. The anomalies are detected more accurately by this modification than the one referred to. According to Balajinath, samples get filtered at three levels vis-a-vis., match index, entropy index and newness index, and the samples that do not pass each of the layers will fail the test of the next layer. For example, samples satisfying the criteria of match index will be tested for neither newness index nor entropy index. Contrastingly, in our scheme, whole command sample set is received as input at all three steps. This results in an extensive training of the command history effecting a better accuracy. The testing continues until the last chromosome is removed. Each time as the testing progresses, the indices are altered with tolerance factors α , β and γ .

Samples lying below a match index threshold of 0.05 as illustrated in Figure 4, are considered to be intrusive. Similarly those samples having an entropy and newness index above the threshold of 0.8 and 0.2 as shown in Figures 3 and 5 respectively are intrusive. For instance, sample 97 with match index less than 0.05, entropy index greater than 0.8 is regarded as intrusive by both Steps 3b and 3c.

2.2 Network Based GAAD

The design for building a Network Anomaly detection system using Genetic algorithm is demonstrated in Figure 6. TCPDUMP, a tool for the extraction of the packets transmitted in the network is used for audit collection. Utilizing this, the required header information of all the packets is extracted to characterize the network behavior. Association rules are then formulated using Apriori Algorithm as in [2].

Training: In the training phase, the association rules are formulated from the necessary parameters of the network traffic. A few unwanted sets of parameters are included in the Normal network traffic, resulting in addi-

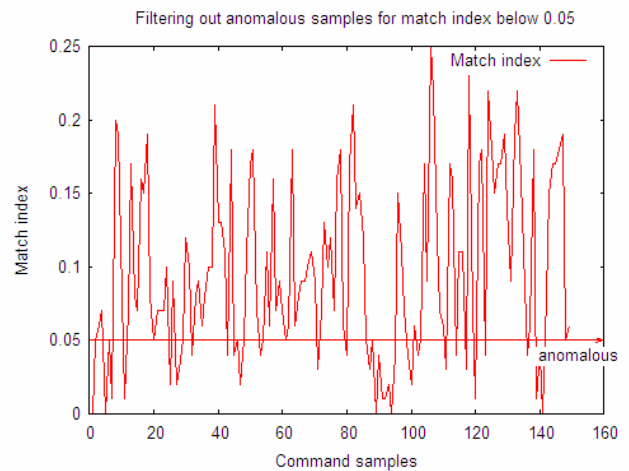


Figure 4: Match index of command samples

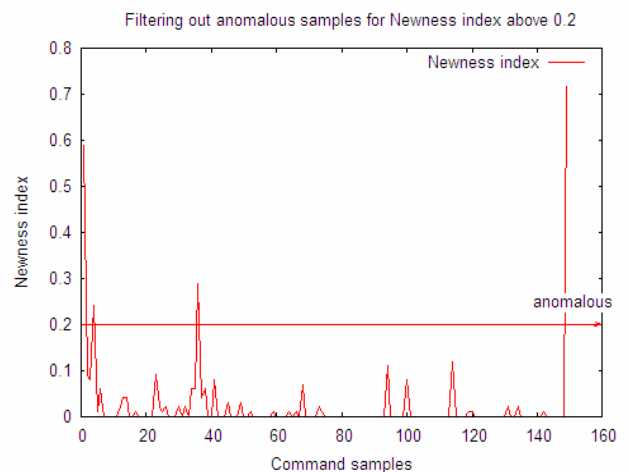


Figure 5: Newness index of command samples Detection of anomalies in a typical user command history using GAAD

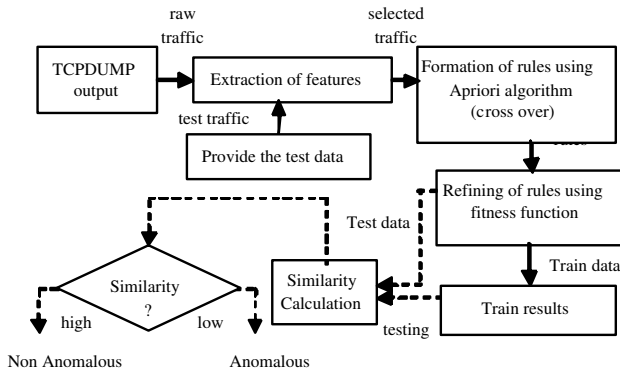


Figure 6: Network based GAAD training [Testing]

tional rules. Hence, the computations become tedious, thereby abating its performance. Hence only a limited number of attributes (axis attributes) [24] including source and destination address, source and destination ports are considered. These parameters are crossed with each other, creating a new set of rules. Each rule is endowed with a minimum support value and confidence value.

The steps involved in training is as summarized below:

- 1) Collect network traffic.
- 2) Reduce the traffic to get the required parameters - Protocol, Source address, Destination address, Source and Destination port numbers.
- 3) Choose protocol as the axis attribute and form frequent association rules.
- 4) Refine the rules using the fitness function. i.e., if support value \geq minimum support value.
- 5) $NewMinSup = (1 - minsup)(1 - \frac{CurrentFeatures}{NumberofFeatures})K + minsup$.
- 6) Rules having support value below the minimum support value are removed. K is a user factor decided upon experimental analysis.
- 7) Store the rules.

Testing: In the testing phase, the rules are postulated in the same manner as in the training phase. These rules are compared with those formed during the training phase, and their similarity is determined using the formula below [7]:

For rules $R1 : X \rightarrow Y$, c , s and $R2 : X' \rightarrow Y'$, c' , s'

$$Similarity(R1, R2) = \begin{cases} 0 & \text{if } (X \neq X' \& Y \neq Y') \\ \max\left(0, 1 - \max\left(\frac{|c-c'|}{\max(c,c')}, \frac{|s-s'|}{\max(s,s')}\right)\right) & \text{if } (X = X' \& Y = Y') \end{cases}$$

$$s = \sum_{\substack{\forall R1 \in S1 \\ \forall R2 \in S2}} similarity(R1, R2).$$

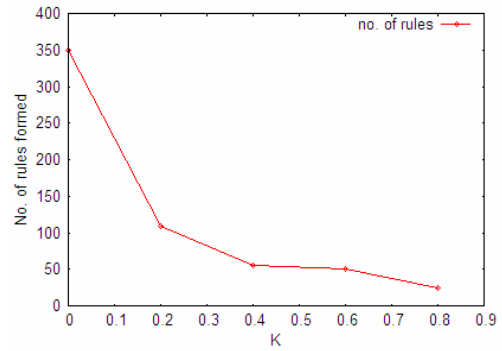


Figure 7: Impact of K on the number of rules generated

The Total similarity is

$$similarity(S1, S2) = \frac{s}{\lfloor |s1| \rfloor} \times \frac{s}{\lfloor |s2| \rfloor},$$

where, c, c' : confidence, the ratio of the number of transactions that contain both X and Y to the number of transactions that contain only X ; s, s' : support, the percentage of the transactions in which both X and Y appear in the same transaction.

The steps involved in testing for anomalies are:

- 1) Network traffic is collected, aided by TCPDUMP.
- 2) Output is reduced to the required parameters - Protocol, Source - Destination addresses and port numbers.
- 3) The protocol is chosen as the axis attribute and frequency association rules are formed.
- 4) The rules are refined with the fitness function as the guidelines i.e., if support value \geq minimum support value.
- 5) The similarity between the current rule set and the reference rule set is computed as per the above formula.
- 6) The test data is declared as intrusive if the similarity is below a threshold.

The variation in the number of rules generated with the variation in the factor K of the fitness function is given in the graph (Figure 7). A K value near 0.5 gives the best accuracy.

3 Neural Network Based Anomaly Detection

Neural networks are parallel architectures, which resemble the animal nervous system. A gamut of inputs is processed by a biological neuron. Artificial Neural networks (ANN) have been developed as generalizations of mathematical models of human cognition or neural biology

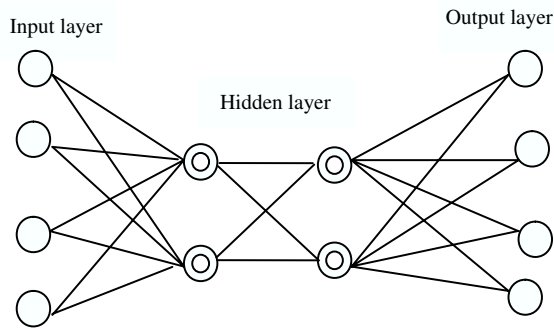


Figure 8: Neural network

[22], based on a few assumptions vis-a-vis, 1. Information processing is achieved by simple elements called neuron, 2. Signals are exchanged by neurons through connection links, 3. The transmitted signal is multiplied by each link, which is associated with its corresponding weight, 4. The net input is converted by an activation function to a corresponding output signal of every neuron. An input layer, one or more hidden layer and an output layer are the components of an ANN. Performance of a neural net is proportional to the number of hidden layers and the number of neurons per layer. The structure is represented in the Figure 8.

The links at all the nodes (as shown in Figure 8) are assigned weights, which are the results of the activation function on sum of the inputs. A sigmoidal activation function used for this purpose is given by,

$$F = \frac{1}{(1 + e^{-w})}$$

3.1 Host Based NNAD

It is assumed that an anomalous behavior differs considerably from a normal one. This difference can be measured and used to detect an intruder [8, 25]. Neural networks have proved to be valuable in the effective detection of intruders [26, 27]. Being a supervised learning method and, equipped with a provision for optimal weight updation and due to the feedback of error, Back propagation model is preferred than other models [16]. Thus, the inputs can be tuned to the desired output. The procedure is as explained in Figure 9.

The Neural Algorithm: Analysis of the user behavior and maintenance of a basic list of 100 most frequently occurring commands in Linux environment are the pre-requisites here. The steps involved in implementing Anomaly detection using a Neural network are as follows.

- 1) Audit collection: This pertains to the tracking of system calls from user commands.

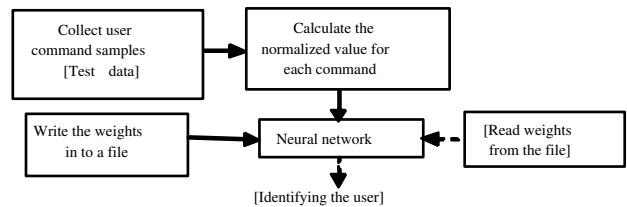


Figure 9: Host based NNAD - Training

- 2) Audit pre-processing: The commands are segregated according to the user id. Command frequency details of each user are calculated.
- 3) Normalization is done.
- 4) A pattern of the No. of commands x 1 for each user is computed.
- 5) Train/Test
If new network
 Perform *<Neural algorithm>*;
else
 Train/Test Neural network.
- 6) If value < threshold
 Declare anomalous;
else
 Perform *<Neural algorithm>*.

Neural Algorithm:

- 1) Initialize system variables.
- 2) If new network
 Initialize all the nodes;
else
 Read the weights from the file.
- 3) Train the network.
- 4) Get the number of patterns.
- 5) For each epoch
 - a. Get all user patterns (command vector).
 - b. For each hidden and output neuron, append the weights and apply the activation function.
 - c. For each hidden and output neuron calculate the error.
 - d. Gather the calculated errors from all the nodes.
 - e. Broadcast the error to all the nodes.
 - f. Update the weights based on the broadcasted error.
- 6) Write the trained network to the file.

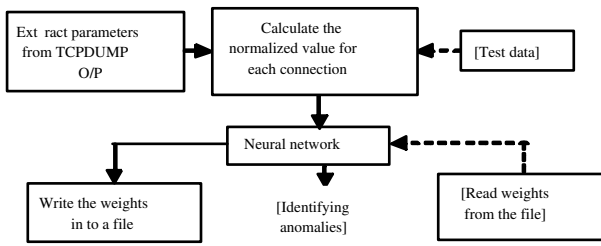


Figure 10: Network based NNAD - Training [Testing]

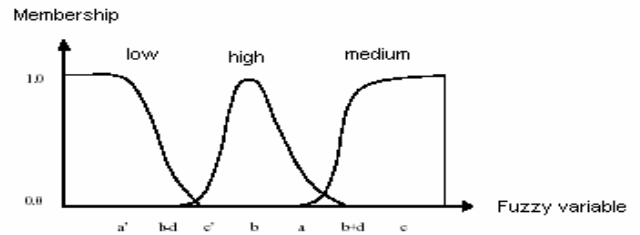


Figure 11: Representation of fuzzy membership

3.2 Network Based NNAD

Similar to Network based GAAD, TCPDUMP data is analyzed to characterize windowed traffic intensity behavior. At any given point of time, there can be many sources of communication with the victim. Prior to collecting and monitoring the network traffic purportedly holding information that reveals the anomalies, the Neural network’s structure is built. Input to the Neural network is supplied by the current intensity of network traffic [6]. All the resources are monitored with ease by supervising the available ports. Due to the practical difficulty in monitoring all the ports, five most important and frequently used ports depending upon the network usage, such as HTTP, FTP, SMTP etc are selected. The Neural network structure is established to contain $(N \times M)$ input nodes, where N is the number of workstations in use and M is the number of ports that are monitored. Two neurons vis-a-vis., intrusive and non-intrusive are the constituents of Output of the Neural network. The procedure is depicted in the Figure 10.

The above process necessitates analyzing the TCPDUMP output, extracting and reducing the necessary parameters. Similar to GAAD, DARPA dataset is used for this purpose. The procedure involved is as explained below.

- 1) The IP addresses and the port numbers are extracted from the dataset.
- 2) Most frequently accessed ports are determined.
- 3) The number of times a source is accessed through the monitored port is determined. The frequency of access of the monitored port on a particular source is calculated.
- 4) The normalized value is written in to a file.
- 5) Test or Train.
- 6) If new network
 Perform $\langle \text{Neural algorithm} \rangle$;
 else
 Test Neural network.
- 7) If value $\langle \text{threshold} \rangle$
 Declare anomalous;

else

 Perform $\langle \text{Neural algorithm} \rangle$.

Neural algorithm analogous to the one expounded in Section 3 is employed. Once the Neural network is created, it is trained and tested with the network traffic collected for 10 seconds. An exponential increase in the time taken for training the network with the number of runs is reported, the results of which are discussed exhaustively in Section 5. Results obtained on testing the same Neural network for incoming traffic are also presented.

4 Fuzzy Logic Based Anomaly Detection

For more complex systems with sparse, imprecise and ambiguous numerical data, the observed input and output situations are approximately correlated by, and the system behavior modeled on fuzzy reasoning [14, 28]. Anomaly detection problem can be tackled by Fuzzy logic because security itself implicates fuzziness. Anomaly is mapped onto fuzzy set variables HIGH, LOW and MEDIUM [31]. The fuzzy membership graph is illustrated in Figure 11.

The fuzzy membership functions are calculated as follows:

$$\begin{aligned}
 high(\mu, a, c) &= \begin{cases} 0 & \mu \leq a \\ 2 \left(\frac{\mu - a}{c - a} \right)^2 & a < \mu \leq \frac{a+c}{2} \\ 1 - 2 \left(\frac{\mu - a}{c - a} \right)^2 & \frac{a+c}{2} < \mu \leq c \\ 1 & c < \mu \end{cases} \\
 low(\mu, a, c) &= 1 - high(\mu, a, c) \\
 medium(\mu, a, c) &= \begin{cases} high(\mu, b - d, b) & if(\mu \leq b) \\ low(\mu, b, b + d) & if(b < \mu) \end{cases}
 \end{aligned}$$

Qualities are to be scrutinized in the anomaly detection domain. For example, the number of different destination IP addresses in the last 2 seconds is considered. A typical rule that may read like “IF the variation of destination addresses during the last 2 seconds was high THEN an unusual situation exists” is applied to this example. Association rules are indited using Apriori algorithm, as discussed in Section 2. Fuzzy logic is employed to build a generic

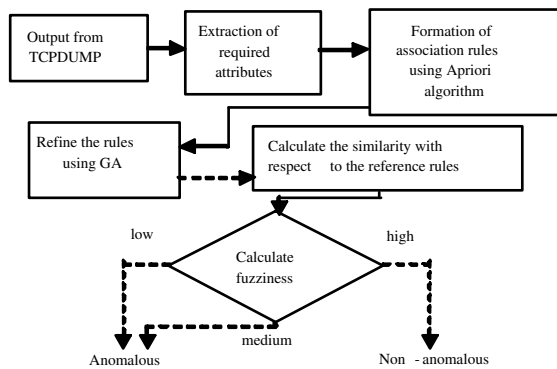


Figure 12: Network based FLAD - Testing

Anomaly detector. To detect anomaly, fuzzy association rule sets are mined from the TCPDUMP data and a similarity is arrived at, on comparison with the sets mined from the normal data [12]. An alarm is raised on encountering a relatively low similarity value when mapped onto fuzzy sets. Training is consummated by forming association rules using the Apriori algorithm (as in Section 2), which are further refined using Genetic algorithm.

The process of mapping the similarity distance into fuzzy sets is assisted by Fuzzy logic, facilitating an unambiguous classification of the nature of Input samples. It has been observed that Genetic algorithms can be successfully applied to tune the membership functions of the fuzzy sets used by our anomaly detector [29]. The control flow of the Fuzzy logic based Network anomaly detection system is exemplified in Figure 12. The steps involved are:

- 1) Network traffic is collected, aided by TCPDUMP.
- 2) Output is reduced to obtain the required parameters - Protocol, Source and Destination addresses, Source and Destination port numbers.
- 3) Choose protocol as the axis attribute and form frequent association rules. (The axis attribute is selected as the protocol and frequency association rules are compiled.)
- 4) The rules are refined with the fitness function as guidelines i.e., support value \geq minimum support value.
- 5) The similarity between the current rule set and the reference rule set is computed as per the above formula.
- 6) Fuzziness is calculated. If it maps to

High- Declare non intrusive

Medium- Declare intrusive

Low- Declare intrusive

For the experiments reported here, the minimum support and confidence values considered are 0.0 and 0.3 respectively. This can be varied for experimental convenience. The discussion and comparison are in Section 5.

5 Experiments and Results

5.1 Host Based Anomaly Detection

Command histories of 26 users collected from the Purdue University Laboratories, are used for the experimental analysis. Several novel approaches that are independent of the command histories and operating environment have been initiated [30]. In order to benchmark the approach, the conventional way of maintaining a basic list of most frequently used commands is executed in the Linux environment. Since the system is trained with each user command sample, the user specific commands are added to the basic list. The consequent enhancement of the user behavior characterization process further accentuates the performance of the Anomaly detection system. The frequency of occurrence of these commands in the user commands' sample is subsequently determined.

The system is trained with the command histories of each user. On testing another user's command sample, the occurrence of an anomaly is denoted. The performance of the system is shown in the Figures 13 and 14. Parallel to the increasing command history size, a concurrent increase in the detection rate along with a decrease in false alarm rate has been validated. The negligible time taken for training and testing is regarded as the principal advantage of applying Genetic algorithms to the Anomaly detection domain. From the above plots it can be inferred that the detection rate and the false positive rates are saturated beyond a threshold of the command history size. The detection capacity cannot be improved after a certain level. This is because sporadically employed commands with minimal contribution to the detection process also populate the command history.

The following features of the GAAD based detection model has been deduced from the results:

- 1) Knowledge of new attacks and system vulnerabilities is superfluous, since detection is achieved by observing deviation in user behavior patterns.
- 2) Anomaly detection can be performed in real-time. Amount of log data processed to detect anomalies is low.
- 3) A low false alarm rate.
- 4) A drift in the user's behavior is identified and captured by continuously learning the user behavior.
- 5) A decentralized manner is adopted in the operation of GADD, wherein the algorithm is run in each node independently.

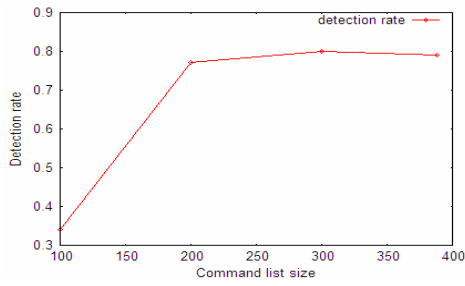


Figure 13: Host based GAAD - Detection rate

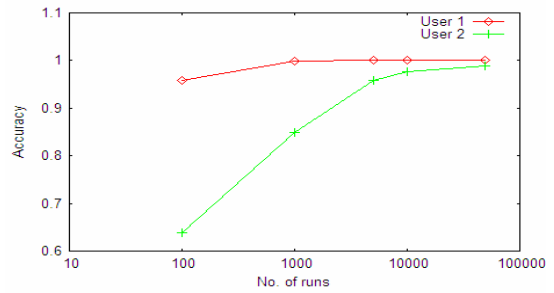


Figure 16: Detection accuracy

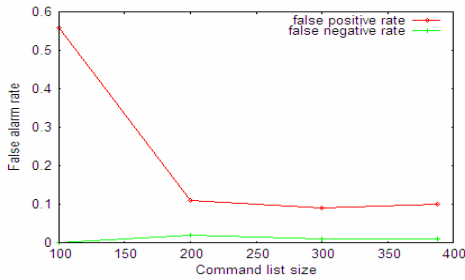


Figure 14: Host based GAAD - Performance evaluation

A host based Neural network anomaly detector (NNAD), with a neural network structure consisting of 100 input neurons, 100 hidden neurons (single layer) and 10 output neurons was designed. Command samples similar to those for the GADD system, are given as input to the system. The time taken for training, detection and false alarm rate with respect to variations in the number of input and hidden neurons are plotted in Figure 15. In Neural network based anomaly detector, a decline in performance due to the enormous amount of time taken to train the system is perceived to be the main disadvantage, construed from the above plots. However, in comparison to the system based on Genetic algorithms, more accurate results and lesser false alarms are obtained.

It can be inferred from plot 16 that the detection ac-

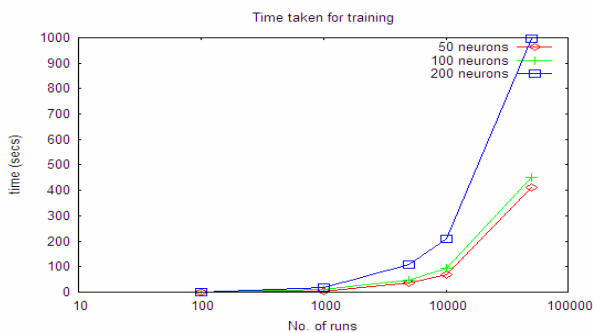


Figure 15: Host based NNAD - Time vs. number of input neurons

curacy reaches almost unity for 50,000 runs. Thus, at the cost of training time, accuracy can be enhanced by extensive training. From the above graphs, it can be understood that a plot of the time taken for training is a parabolic curve $y = kx^2$. Thus for applications where time is not a constraint and offline training is possible, the advantages of Neural network can be exploited. It was noted that the pre-processing time amounts to be the same for both GAAD and NNAD systems. With the number of training epochs, a corroborative increase in the accuracy of detection using Neural networks has been detected. The exhibited high false alarm rate of GAAD system can be reduced by training the system with a large set of command sample, thereby equaling its performance to that of NNAD system, with a concomitant reduction in processing time.

5.2 Network Based Anomaly Detection

The evaluation of this approach will be based on data sets provided by the Defense Advanced Research Projects Agency (DARPA) IDS evaluation in 1999 [19], obtained as TCPDUMP binaries, and collected from MIT Lincoln Laboratories. The experiments' requirements are not met by the 1999 DARPA data sets, with respect to hostile traffic because the range of vulnerabilities has evolved substantially since 1999. Experimentation based exclusively on such relatively old attacks could lead to unpractical results in current environments, which are exposed to more sophisticated threats. Moreover, controversial comments on these datasets have been listed in the literature [21]. Nevertheless, abandoning the idea of using the DARPA data sets is inexpedient, because variations in the accuracy of anomaly detection is profound only when there a considerable deviation from the existing traffic patterns, and these can be easily found in the DARPA datasets. This accuracy is relatively immutable in case of new classes of attacks. The value of these datasets is amplified by its popular acceptance and their extensive usage by the research community. Furthermore, comparability and reproducibility of the experiments is ensured by the use of these data sets, whereas the use of intrinsic synthetic data fails.

The system is trained by collecting network traffic



Figure 17: Network based GAAD - Similarity calculation

pumped from the TCPDUMP output of the DARPA dataset for every 10 and 20 seconds. Pre-processing of input is accomplished by extracting the necessary parameters such as protocol type, source & destination addresses, and source & destination port. The time exhausted for the training and testing of GAAD system is negligible, when compared to that of the NNAD system as depicted in Figure 18. The detection and false alarm rates are depicted in Figure 17, above by comparing intrusive and non-intrusive samples with respect to the trained data. Intrusive samples of 10 and 20 seconds are generated randomly and are tested against the trained data. Samples below the threshold value of 0.5 (similarity) are considered intrusive. Those samples between 1 and 20 whose similarity is 0 are denoted as false negatives. Despite requiring relatively longer time for training in a network NNADS, an accuracy of 100% with very minimum false alarms has been confirmed. NNADS proves to be efficient in applications where time is not critical with no false positives.

In Fuzzy based anomaly detection, similarity is calculated based on the Genetic algorithm system and the values are mapped onto fuzzy sets. The same network traffic is provided as input for the testing and training phase. While a crisp value of anomalies is presented by GA, its fuzziness is mapped by FLAD. Even though anomalies are detected more accurately with negligible false positives, the false negatives purported by this system are higher than the other systems. The samples from 31 to 40 are intrusive as the fuzzy values remain at 0. From Figure 19, it can be inferred that numerous false negative alarms generated by the system. This can be reduced by extensive training.

To summarize, Table 1 proves that anomaly detection using Neural network exhibit more complexity in terms of space and time as compared to Genetic algorithms & Fuzzy logic. However, the performance in terms of detection and false alarm rate is better in Neural networks and hence proves to be a better pick in accuracy specific scenarios.

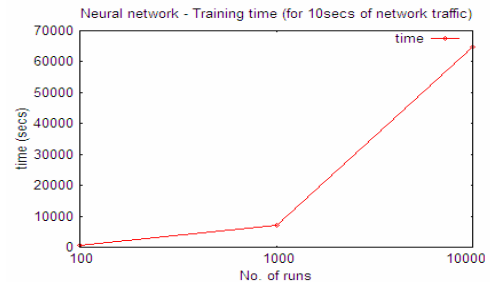


Figure 18: Network based NNAD - Time taken for training



Figure 19: Network based FLAD - Fuzzy mapping

6 Conclusion and Future Work

Various financial transactions that occur as a direct consequence of Electronic Commerce have made anomaly detection a crucial issue in computing systems. The inadequacies of Misuse detection in detecting all types of anomalies due to the advent of new attacks and system vulnerabilities has been indubitably established, and consequently the development of Anomaly detection has assumed paramount importance. However, high false alarm rate in anomaly detection due to the vagaries in a user's behaviour has to be addressed before implementing it commercially. False alarm rates are greatly reduced by soft computing, thus improving the detection rate due to extensive training. For the same input, different behavior is exhibited by various techniques of soft computing. A trade off between time taken for training & testing and the accuracy of detection is professed by the results of the work.

Host based GAAD system is implemented by regarding commands as the basic units (genes), which consume less time and space. However, the accuracy of the NNAD system is found lacking in them, apropos of detection. A back propagation model is used for implementing NNAD, since it provides optimum weight updation due to the back propagation of error. Nevertheless a greater time complexity is evident, in comparison to its counterparts.

Table 1: Comparison of Performance of anomaly detection using various soft computing techniques

Type		Time complexity	Space complexity	Detection rate (in %)	False alarm rate (in %)
Host based	NNADS	$O(nm)$	$O(iho)$	100	Negligible
	GAADS	$O(c)$	$O(1)$	88	12
Network based	NNADS	$O(nm)$	$O(iho)$	99.14	1.86
	GAADS	$O(r)$	$O(r)$	93.05	6.95
	FLADS	$O(r)$	$O(r)$	91.75	8.25

n : number of neurons; m : number of runs; c : number of input commands; i : input neurons; h : hidden neurons; o : output neurons

Network based GAAD system is designed by forming rules based on network traffic. During the testing phase, new rules are formulated. Anomaly is detected by comparing these two rule sets, and determining similarities if present. More space is required to store these rules. Since, only 10 seconds of the network traffic is considered, the processing time is negligible. Contrarily, in NNADS time increases parabolically to the increase in number of runs, although a greater accuracy is offered by the system. In fuzzy based system, the space and time complexities are the same as in GAAD system, but the detection accuracy is less. It is also constructed by forming rules using the Apriori algorithm and testing for similarity. Similarity values are then mapped onto fuzzy sets. Establishing the pros and cons of these soft computing techniques, it could be concluded that depending upon the requirement, any of the methods discussed in this work can be exploited.

The work could be further improved as stated below. The system has a known fallacy; it was ported on Linux environment, which could characterize users' Command samples or Network packets. This could possibly be extended to other environments to support GUI behavior characteristics of the users. As some of the methods present slightly higher false positives rate, methods involving support vector machines and rough sets can be investigated upon for reduction of these false alarms. Anomaly detection system can be coupled with misuse detection to achieve better detection rate. The recent approaches of evolutionary algorithms could replace soft computing and its advantages could be exploited.

References

- [1] A. Abraham, and R. Jain, "Soft computing models for network intrusion detection system," *Soft Computing in Knowledge Discovery: Methods and Applications*, ch.16, Studies in Fuzziness and Soft Computing, Springer-Verlag, May 2004.
- [2] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of The 20th International Conference on Very large Databases*, pp. 487-499, Santiago, Chile, Sep. 1994.
- [3] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner, *State of the Practice of Intrusion Detection Technologies*, Carnegie Mellon Software Engineering Institute, SEI Technical Report, CMU/SEI-99-TR-028, 2000.
- [4] R. G. Bace, *Intrusion Detection*, Macmillan Technical Publishing, 2000.
- [5] B. Balajinath and S.V. Raghavan, "Intrusion detection through learning behavioral model," *Computer Communications*, vol. 24, no. 12, pp. 1202-1212, July 15, 2001.
- [6] A. Bivens, M. Embrechts, C. Palagiri, R. Smith, and B. Szymanski, "Network-based intrusion detection using neural networks," *Proceedings of Artificial Neural networks In Engineering*, vol. 12, pp. 10-13, St. Louis, Missouri, Nov. 2002.
- [7] S. M. Bridges, and R. B. Vaughn, "Fuzzy Data mining and Genetic algorithms applied to Intrusion Detection," *Proceedings of the 23rd National Information Systems Security Conference*, Baltimore, MD, Oct. 2000.
- [8] Y. Chen, A. Abraham, and B. Yang, "Hybrid flexible neural-tree-based intrusion detection systems," *International Journal of Intelligent Systems*, vol. 22, no. 4, pp. 337-352, Apr. 2007.
- [9] A. Chittur, *Model Generation for Intrusion Detection System using Genetic Algorithms*, High School Honors Thesis, Ossining High School, Ossining, NY, Nov. 27, 2001.
- [10] M. Crosbie, and E. Spafford, "Applying genetic programming to intrusion detection," *Proceedings of AAAI Fall Symposium*, pp. 1-8, AAAI Press, Nov. 1995.
- [11] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, Wiley-IEEE Press, 3rd ed., Dec. 2005.
- [12] F. German, S. Bridges, and R. Vaughn, "An improved algorithm for fuzzy data mining for intrusion detection," *Proceedings of the North American Fuzzy Information Processing Society Conference (NAFIPS 2002)*, New Orleans, LA, June 27-29, 2002.
- [13] A. K. Ghosh, A. Schwartzbard, and M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection," *USENIX Workshop on Intrusion Detection and Network Monitoring*, pp. 51-62, California, USA, 1999.

- [14] J. Gomez, F. Gonzalez, and D. Dasgupta, "An immuno-fuzzy approach to anomaly detection," *Proceedings of the 12th IEEE International Conference on Fuzzy Systems*, vol. 2, pp 1219-1224, Memphis, TN, USA, May 2003.
- [15] F. Karray, and C. W. D. Silva, *Soft Computing and Intelligent Systems Design, Theory, Tools and Applications*, pp. 560, Addison Wesley, Pearson Education Limited, Essex, England, Sep. 2004.
- [16] H. G. Kayacik, A. N. Z. Heywood, and M. I. Heywood, "A hierarchical SOM-based intrusion detection system," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 4, pp. 439-451, June 2007.
- [17] D. S. Kim, H. N. Nguyen, and J. S. Park, "Genetic algorithm to improve SVM based network intrusion detection system," *Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, vol. 2, pp. 155-158, 2005.
- [18] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," *IEEE Security and Privacy Symposium*, pp. 120-132, Oakland, Canada, 1999.
- [19] R. P. Lippmann, et al., "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," *Proceedings of the DARPA Information Survivability Conference and Exposition*, IEEE Computer Society Press, 2000.
- [20] W. Lu, and I. Traore, "Detecting new forms of network intrusion using genetic programming," *IEEE Evolutionary Computing*, vol. 3. pp. 2165- 2172, Dec. 8-12, 2003.
- [21] M. V. Mahoney, and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection," *Proceedings of Recent Advances in Intrusion Detection (RAID)*, pp. 220-237, 2003.
- [22] S. Mukkamala, and A. H. Sung, *Soft Computing Techniques for Intrusion Detection*, Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, George Mason University, ONR and AFSOR Sponsored Workshop on Intrusion Detection, Fair Fax, 2003.
- [23] T. Ozyer, R. Alhajj, and K. Barker, "Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening," *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 99 - 113, Jan. 2007.
- [24] M. M. Pillai, J. H. P. Eloff, and H. S. Venter, "An approach to implement network Intrusion detection using Genetic algorithm," *Proceedings of the annual South African Institute of Computer Scientists and Information Technologists conference (SAIC-SIT)*, pp. 221-228, Stellenbosch, SA, Unisa Press, 1-58113-982-9, Oct. 2004.
- [25] J. Ryan, M. J. Lin, and R. Miikkulainen, "Intrusion detection with neural networks," *Advances in Neural Information Processing Systems*, vol. 10, pp. 943-949, The MIT Press, Denver, CO, 1998.
- [26] S. T. Sarasamma, Q. A. Zhu, and J Huff, "Hierarchical kohonen net for anomaly detection in network security," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 35, no. 2, pp. 302-312, Apr. 2005.
- [27] N. Srinivasan, and V. Vaidehi, "Anomaly detection in a distributed environment using neural networks on a cluster," *IASTED International Conference on Communication, Network, and Information Security (CNIS 2005)*, Phoenix, USA, Nov. 2005.
- [28] C. H. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection," *Pattern Recognition*, vol. 40, no. 9, pp. 2373-2391, Sep. 2007.
- [29] W. Wang, and S. M. Bridges, "Genetic algorithm optimization of membership functions for mining fuzzy association rules," *Proceedings of the 7th International Conference on Fuzzy Theory & Technology*, pp.131-134, Atlantic City, NJ, Feb. 27-Mar. 3, 2000.
- [30] M. Wang, C. Zhang, and J. Yu, "Native API based windows anomaly intrusion detection method using SVM," *Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 1, pp. 514- 5191, Xi'an Jiaotong University, China, June 05-07, 2006.
- [31] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 28-44, 1973.

N. Srinivasan received a BE in Electrical and Electronics Engineering in 2000 and ME in Computer Science and Engineering in 2002 from the University of Madras. He is a PhD candidate at the Department of Information Technology of the Madras Institute of Technology, Anna University, India. Since 2002, he has been involved in several research projects focusing on Data and Network security, Theoretical Computer science.

V. Vaidehi received her BE in Electronics and Communication Engineering from College of Engineering, Guindy, ME in Applied Electronics and PhD from Madras Institute of Technology, Chennai. She was a recipient of academic exchange fellowship of Association of Commonwealth Universities. She has carried out funded projects on Tracking Algorithm for ship borne RADARS ?funded by LRDE; GPS signal simulator funded by Ministry of Information Technology; University Micro satellite funded by ISRO; Semantic Intrusion Detection System funded by Xambala Inc. Currently she is a Professor and Head of Department of Electronics Engineering, Madras Institute of Technology, Chennai. Her areas of interests are Network security, Parallel processing and Embedded systems.