Int. J. Open Problems Compt. Math., Vol. 2, No. 2, June 2009

Escalate Intrusion Detection using GA - NN

S. SELVAKANI[1] and R.S.RAJESH[2]

[1] Jaya Engineering College, Chennai, Tamilnadu, India.
[2] Dept of CSE, Manonmanium Sundaranar Universtiy, Tirunelveli Tamilnadu, India.
E-mail: sselvakani@hotmail.com

Abstract

Intrusion Detection systems are increasingly a key part of system defense. Various approaches to Intrusion Detection are currently being used but they are relatively ineffective. Among the several soft computing paradigms, we investigated genetic algorithms and neural networks to model fast and efficient Intrusion Detection Systems. With the feature selection process proposed it is possible to reduce the number of input features significantly which is very important due to the fact that the RBF networks can effectively be prevented from over fitting. The Genetic algorithm employs only the eight most relevant features for each attack category for rule generation. The generated rules signal an attack as well as its category and it is end for training to RBF network. The optimal subset of features combined with the generated rules, can be used to analyze the attacks. Empirical results clearly show that soft computing approach could play a major role for intrusion detection. The model was verified on KDD99 demonstrating higher detection rates than those reported by the state of art while maintaining low false positive rate.

Keywords***: Genetic algorithms, Information gain, Mutual Information, Radial Basis Function Networks, Soft Computing***

1 Introduction

Along with bringing revolution in communication and information exchange, Internet has also provided greater opportunity for disruption and sabotage of data that was previously considered secure. While we are benefiting from the

convenience that the new technology has brought us, computer systems are facing increased number of security threats that originate externally or internally. As malicious intrusions into computer systems have become a growing problem, the need for accurately detecting these intrusions has risen. Despite numerous technological innovations for information assurance, it is still very difficult to protect computer systems. Therefore intrusion detection is becoming an increasingly important technology that monitors network traffic and identifies, preferably in real time, unauthorized use, misuse and abuse of computer systems.

A number of approaches based on computing have been proposed for detecting network intrusions. The guiding principle of soft computing is exploiting the tolerance of imprecision, uncertainty, partial robustness and low solution cost. Soft computing includes many theories such as Fuzzy logic, Neural Networks, Artificial intelligence, Information and probabilistic reasoning and Genetic Algorithms. When used for intrusion detection, soft computing techniques are often used in conjunction with rule-based expert systems where the knowledge is usually in the form of if-then rules [3]. Despite different soft computing based approaches having been proposed in recent years, the possibilities of using the techniques for intrusion detection are still under utilized.

In our approach we consider Intrusion detection as a data analysis process. Network behaviors can be categorized into normal and abnormal. Due to the sheer volume of real network traffic, both in the amount of records and in the number of features, it is extremely difficult to process all the traffic information before making decisions. We need to extract the most important data that can be used to efficiently detect network attacks. We use information theory to identify the most relevant features to be used [12]. In this work the initial point is the extraction of the most important piece of information that can be deployed for efficient detection of attacks in order to cope with this problem.

Our idea is to achieve high detection rate by introducing high level of generality when deploying the subset of the most important features of the dataset. As this also results in high false positive rate, we deploy additional set of rules in order to recheck the decision of the rule set for detecting attacks. We deploy genetic algorithm (GA) approach for offline training of the rules for classifying different types of intrusions. Genetic Algorithm field is one of the upcoming fields in computer security and has only recently been recognized as having potential in the Intrusion Detection field.

Neural Networks have been actively applied to IDS. To apply neural networks to real world problem successfully, it is very important to determine the number of hidden nodes in the given problem, because performance hinges upon the structure of the neural networks. Hence we use RBF for learning the rules. We examine the proposed method through experiments with real data and compare the results with those of other methods.

The aim here is to develop an Intrusion Detection System which adapts to the environment. Evolution and learning are the two most fundamental process of

adaptation. Since learning through neural network is a complex, time consuming process, the connection between learning and evolution can be used to decrease the complexity of the problem.

The rest of the paper is organized as follows: Section 2 presents an overview of related works. Section 3 gives overview on attacks within the KDD data set and proposes the deficiencies of the data set. Section 4 gives overview on Genetic Algorithms and the benefits of its using in intrusion detection field. Section 5 discusses the detection rate of our algorithm when applied to the KDD 99 data.

2 Related Works

M A. Chittur extended their idea by using GA for anomaly detection [2]. Random numbers were generated using GA. A threshold value was established and any certainty value exceeding this threshold value was classified as a malicious attack. The experimental result showed that GA successfully generated an empirical behavior model from training data. The biggest limitation of this model was the difficulty of establishing the threshold value which might lead to detect novel or unknown attacks.

J. Gomez et al.[3] proposed a linear representation scheme for evolving fuzzy rules using the concept of complete binary tree structure. GA is used to generate genetic operators for producing useful and minimal structure modification to the fuzzy expression tree represented by chromosomes. The biggest drawback of the proposed approach was that the training was time consuming.

Liao and Vemuri used the K-nearest Vector Machine [10] for profiling computer programs. The KNN classifier was employed with an interesting analogy between classifying text documents and detecting intrusion using the sequences of system calls.

Wang et al. used the evolutionary algorithm [13] for discovering neural networks for intrusion detection. The connections of the network and its weights were encoded with binary bits and evolved simultaneously. Their detection system was evaluated with www log data and showed an accuracy rate of 95%. However, in their experiment, they used their own data set rather than a public bench mark dataset.

Hofmann et al. proposed [7] the evolutionary learning of radial basis function networks for intrusion detection. They targeted a network based IDS. Their evolutionary algorithm performed two tasks simultaneously selecting the optimal feature set and learning the RBFN. The binary bits system was used to encode the 137 possible features of the network packet headers and three components of the RBFN, including the type of basis function, the number of hidden neurons, and the number of training epochs. In the experiments with the network audit data set, the RBFN optimized with the evolutionary algorithm outperformed the normal MLP and the normal RBFN.

Gonzalez et al. proposed an intrusion detection technique based on evolutionary generated fuzzy rules [5]. The conditional part of the fuzzy detection rules was

encoded with binary bits and fitness was evaluated using two factors: the accuracy and the coverage of the rule. The performance was compared to the methods of different genetic algorithms and without the fuzziness of rules using two network audit datasets their own wireless dataset and the knowledge discovery and data mining cup 99 data set.

3 KDD Data Set – Issues and Solutions

Learning algorithms have a training phase where they mathematically learn the patterns in the input data set. The input data set is also called the training set which should contain sufficient and representative instances of the patterns being discovered. A data set instance is composed of features.

In order to promote the comparison of advanced research in the area of intrusion detection, the Lincoln Laboratory at MIT, under DARPA sponsorship, conducted the 1998 and 1999 evaluation of intrusion detection [11]. Based on binary TCP dump data provided by DARPA evaluation, millions of connection statistics are collected and generated to form the training and test data in the classifier Learning contest organized in conjunction with the 5th ACM SIGKDD International conference on Knowledge Discovery and Data mining 1999 [11].

The data set contains 5,000,000 network connection records. A connection is a sequence of TCP packets starting and ending at some well defined moments of time, between which data flows from source IP to a target IP. The training portion of the dataset "kdd-10-percent" contains 494,021 connections of which 20% are normal. The testing data set "corrected" provides a data set with a significantly different statistical distribution that the training data set and contains an additional 14 attacks. It contains 311,029 connections of which 60,593 are normal.

KDD data set is comprised of records. Each record in the data set consists of 41 features [8] where 38 are numeric and 3 are symbolic defined to characterize individual TCP sessions. Each record is also labeled i.e. the information whether it represents an attack or a normal connection is also provided.

As the first step in our work, in order to cope with the speed problem mentioned above, we have used the results obtained in our previous work [12] where we deployed Information Gain based Mutual Information, in order to extract the most relevant features of the data. In this way the total amount of data to be processed is highly reduced. As an important benefit of this arises the high speed of training the system thus providing high refreshing rate of the rule set.

Subsequently, these features are used to form rules for detecting various types of intrusions. This permits the introduction of higher level of generality and thus higher detection rates. The problem that arises with discarding features is a certain increase of false alarm rate.

3 GA Approach

Genetic Algorithms GA are search algorithms based on the principles of natural selection and genetics. GA evolves a population of initial individuals to a population of high quality individuals, where each individual represents a solution to the problem to be solved. Each individual is called chromosome and is composed of predetermined number of genes. The quality of each rule is measured by a fitness function as the quantitative representation of each rule's adaptation. The flow is presented in Fig 1.

The procedure starts from an initial population of randomly generated individuals. Then the population is evolved for a number of generations while gradually improving the qualities of the individuals in the sense of increasing the fitness value as the measure of quality.

During each generation, three basic genetic operators are sequentially applied to each individual i.e. Selection, Cross over and Mutation. Those chromosomes with a higher fitness value are more likely to reproduce offspring. If the new generation contains a solution that produces an output that is close enough or equal to the desired answer then the problem has been solved. If this is not the case, the new generation will go through the same process. This will continue until a solution is reached.

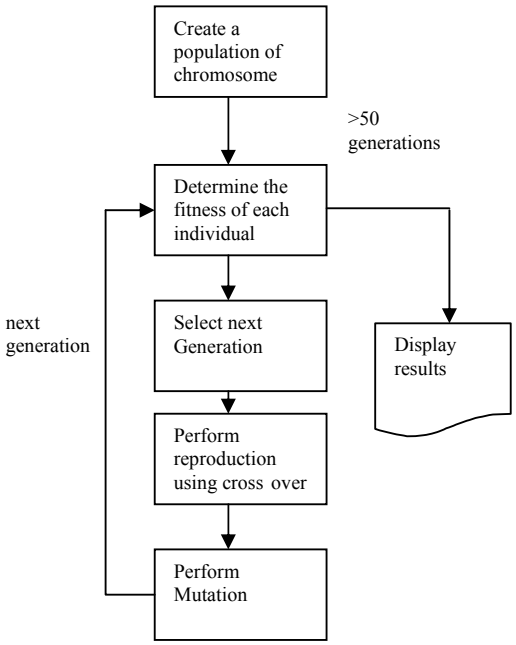

Fig 1: Genetic Algorithm Flow

5 Evaluation Results

The implemented system for intrusion detection consists of a combination of three different parts. The first block represents a feature extraction by using

Information gain [12]. In the continuation the next block represents a rule based system using Genetic Algorithm for known attacks. The last block is the detection of unknown attacks using RBF networks.

We have used an open source machine learning framework WEKA [Waikato Environment for Knowledge Analysis] written at University of Waikato, New zealand. The algorithms can either be applied directly to a data set or called from our own JAVA code. The input data for weka classifiers is represented in .ARFF [Attribute Relation Function Format], consisting of the list of all instances with the values for each instance separated by commas. As a result of data set training and testing, a confusion matrix will be generated showing the number of instances of each class that has been assigned.

The raw data from the KDD 99 is first partitioned into four groups (input data set), DoS attack set, Probe attack set, U2R attack set and R2L attack set. For each attack set different connection record feature set are selected as attributes. The goal of this experiment is to generate a rule which can detect whether a connection is an attack or normal. Each of them forms as "if (condition) then attack" where condition is made up of attributes which belong to feature subset and their values. The proposed method is implemented using the JAVA Language. The Ranked list of features is shown in the Table 1.

Attack type	Ranked List
DOS	5,23,3,33,35,34,24,36,2,39,4,38,26,25,29,30,6,12,10,13,40,41 ,31,37,32,8,7,28,27,9,1,19,18,22,20,21,14,11,17,15,16
Probe	23,29,27,36,4,32,34,40,35,3,30,2,5,41,28,37,33,25,38,26,39,1 0,9,12,11,6,1,8,7,21,19,20,31,22,24,15,13,14,18,16,17
U2R	6,3,13,15,12,14,18,19,16,17,20,4,5,1,2,10,11,7,9,8,35,36,32,3 4,33,40,41,37,39,38,24,25,21,23,22,29,31,30,26,28,27
R2L	3,34,1,6,5,33,35,36,32,12,23,24,10,2,37,4,38,13,16,15,14,8,7, 11,9,29,30,27,28,40,41,31,39,19,20,17,18,25,26,21,22

Table 1: Ranked List of Features

A fundamental step in the design of a GA is the compatible representation of the problem. For the rule generation problem, two representations have been used, each individual in the population (a chromosome) encodes a single rule or each chromosome encodes a set of rules. Both approaches have their pros and cons. We adopt each individual in the population encodes a single rule in this work because of its efficiency resulting from the smaller individuals generated by the approach.

A rule consists of two parts: the antecedent and the consequent. In general, the antecedent part is a conjunction of a number of attribute values. Each categorical attribute in the conjunction is represented by k bits, where k is the number of possible values for the attribute. If a particular value is present then the corresponding bit is set to 1; otherwise, it is zero. Multiple bits may be set to 1 representing a logical OR among the corresponding attribute values. Multiple attributes in the antecedent are chained by the logical AND operator. Numeric attributes are represented by k bits, where k is the number of bins into which the attribute is categorized using equi-width binning.

If the value of the attribute lies within a bin, then the corresponding bit is set to 1; otherwise, it is zero. The consequent part consists of the attack category attribute and its value only. It is represented by m bits, where m is the number of attack categories. If an attack category is present then the corresponding bit is set to 1 and other bits are zero. The consequent part is encoded in the genome of the individual and participates in the evolution process. Our algorithm is run m times, once for each possible value (attack category), generating the best rules for each attack category.

The initial selection for the rules (or individuals in the population) can be done randomly or by using the training dataset to select rules that cover many records. In subsequent iterations of the GA, selection is done based on the fitness value of the rules.

The crossover operation is an important step in a GA. One-point and two-point crossover operations are the most commonly used types. In the one-point crossover operation, the two rules taking part in the operation are cut at a randomly selected point and their left (or right) parts are interchanged. This crossover operation is sensitive to the position of the attributes within the chromosome. For example, the left most and right most attributes are highly unlikely to go in the same partition. We use the two-point crossover operation, in which two cut points are randomly selected in the two rules and the part within the cut point is interchanged. Two-point crossover is independent of the position of the attributes in the rule. The selection of chromosomes that take part in the crossover operation is done by the roulette wheel strategy. The population size is kept constant from one iteration to the next.

Mutation is random flipping of the bits in the rule. It helps avoid getting stuck in local maxima in the search. We use single bit mutation with a low flipping probability.

Each chromosome, representing one learning rule, was evaluated. To evaluate a chromosome, an appropriately sized network was configured for each of the 20 tasks. The following procedure was conducted for each task. For each epoch, the network was shown all the training patterns, and the weights were updated according to the encoded learning rule. The absolute values of connection strengths were capped at 20 to prevent runaway learning rules. The network was presented with each pattern once more, and its outputs were recorded. If the

desired and actual outputs were on opposite sides of 0.5, the response was counted as an error.

An individual (a chromosome) of each population consisted of genes, where each gene represented a certain feature and its values represented the value of the feature. Each GA is trained during 300 generations where in each generation 100 worst performed individuals are replaced with the newly generated ones.

The performance is measured by the fitness function as

$$\text{Fitness} = 3*(\alpha/A)-2*(\beta/B)$$

Where

$\alpha \rightarrow$ number of correctly classified connections

$\beta \rightarrow$ number of incorrectly classified connections

$A \rightarrow$ total number of attacks

$B \rightarrow$ total number of normal connections

High detection rate and low false positive rate result in high fitness value. On the other side low detection rate and high false positive rate results in low fitness value. Greater coefficients at the side of the detection rate give a certain advantage of the detection over false positive rate. The coefficients were chosen after having performed a number of experiments where they assumed different values. This function was used for its simplicity and ease of interpretation.

• The fitness of a chromosome was taken as its average fitness over all twenty tasks, and chromosomes were probabilistically selected for inclusion in the next generation based on their cumulative fitness over generations. The selection mechanism was roulette selection with elitism (that is, the most-fit chromosome was always included in the next generation).

• After the 500th generation, the best chromosome was selected, and the learning rule was encoded using it. The fitness of the learning rule was tested on the 10 datasets that were held back for the testing task. The same process was repeated for ten epochs and the results were analyzed. The above process was repeated with different parameters of genetic algorithm. A two point cross-over and elitist selection was used. The cross over rate was varied from 50% to 80% with an increment of 5% in every step. The mutation rate was varied from 1% to 5% with an increment of 0.5%.

The sample rules are the following ones:

Rule 1:

```
If protocol_type=tcp then
   If service=http then back
  Else if service= private then neptune
   else  if service =finger | telnet then
        if count=1 then land
         if  count >=1&& <=302 then Neptune
```

rule 2:

```
If protocol_type=tcp then
   If service=http then
```

If logged_in=1 then
If dst_host_count =255 then
If dst_host_same=0 then back
Default: normal

Let us now analyze the results obtained from the experiments conducted as explained above. As discussed, we tried the proposed algorithm with different sets of parameter values. The best fitness was 92.4% which was achieved with 55% cross-over rate and 1% mutation rate.

The result of the training process is certain number of best performed rules. We have performed experiments using 50, 75 and 100 best performed rules for detecting attacks.

From Table 2 believing that our system outperforms the best performed model reported in literature. Moreover, our previous statement of reducing false positive rate when deploying additional rule systems for detecting Dos attacks and normal connections is conformed, as the false positive rate has decreased in each of the cases.

Table 2: Performance of the implemented system

No .of rules	Detection Rate in %				False Positive Rate in %			
	DoS	U2R	R2L	Probe	DoS	U2R	R2L	**Probe**
100	86.7	79.2	81.2	86.1	0.9	1.1	1.5	**1.2**
75	81.4	71.3	75.4	83.4	1.9	2.7	2.9	**2.3**
50	**78.3**	**67**	**71**	**82.5**	**2.3**	**3.1**	**3.6**	**2.7**

Ten kinds of network attacks are included in the training set, namely back, land, Neptune, Pod, smurf, teardrop, ipsweep, portsweep, buffer_overflow and guess_passwd. Fifteen kinds of network attacks are included in the testing data set namely perl, xlock, mailbomb, UDPstrom, saint, xlock, back, land, Neptune, pod, smurf, teardrop, ipsweep, portsweep, bufferoverflow and guess_passwd. The test data set is similar with the training data set. The only differences are that the test data set includes some unknown attacks not accruing in the training data set. Detection of attacks involved in the test data set and not occurring in the training data set assesses the potential ability to detect novel attacks.

In the normalization step, we linearly scale each of these features to the range [0.0, 1.0]. Features having smaller integer value ranges like duration [0, 58329], num_compromised [0,255] were scaled linearly to the range [0.0, 1.0]. Two features spanned over a very large integer range, namely src_bytes [0,693375640] and dst_bytes [0, 5203179] were scaled by logarithmic scaling to the range [0.0, 20.4] and [0,15.5]. For Boolean features having values (0 or 1), they were left unchanged.

It should be noted that the test data is not from the same probability distribution as the training data. Moreover the test data includes novel attack types that have not been appeared in the training data.

In the second stage, from the reduced feature subset which is shown in Table 1, rules are formed using the genetic algorithm from the KDD data set and tested on the KDD training set to observe their performance with respect to detection, false alarm rate and missed alarm rate. The only drawback in this is the rules are biased to training data set. The genetic algorithm in the proposed design evaluates the rules and discards the bad rules while generating more rules to reduce the false alarm rate and to increase the intrusion detection. The GA thus continues to detect intrusions and produce new rules, storing the good rules and discard the bad ones.

```
dst_host_srv_count <= 227
|  num_failed_logins <= 0
|  |  rerror_rate <= 0
|  |  |  num_access_files <= 0
|  |  |  |  protocol_type = tcp
|  |  |  |  |  dst_host_same_srv_rate <= 0.11
|  |  |  |  |  |  dst_host_serror_rate <= 0.01
|  |  |  |  |  |  |  dst_host_same_srv_rate <= 0.01: multihop
|  |  |  |  |  |  |  dst_host_same_srv_rate > 0.01: ftp_write
|  |  |  |  |  |  dst_host_serror_rate > 0.01: warezmaster
|  |  |  |  |  dst_host_same_srv_rate > 0.11
|  |  |  |  |  |  is_host_login <= 0: warezmaster
|  |  |  |  |  |  is_host_login > 0: multihop
|  |  |  |  protocol_type = udp: multihop
|  |  |  |  protocol_type = icmp: multihop
|  |  |  num_access_files > 0: ftp_write
|  |  rerror_rate > 0: imap
|  num_failed_logins > 0: guess_passwd
dst_host_srv_count > 227: guess_passwd
```

The summary of the results after RBF training is given as follows:

Correctly Classified Instances	916	99.7821
Incorrectly Classified Instances	2	0.2179 %
Kappa statistic	0.9959	
Mean absolute error	0.0008	
Root mean squared error	0.0269	
Relative absolute error	0.4262 %	
Root relative squared error	9.0025 %	
Total Number of Instances	918	

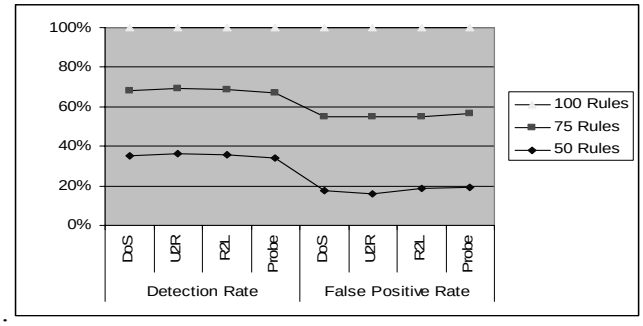

Fig 2: ROC Curve for Intrusion Detection Performance

To show the performance visually, the receiver operating characteristic (ROC) curve, which is a traditional way to represent the performance of the classifier, is used. Fig 2 depicts the ROC curve which illustrates the intrusion detection performance of the proposed method. The DR increases as the false alarm rate does the same. The DR is close to 95 when the false alarm rate is 1.9% to 2%. However the false alarm rate is close to 1%, the DR is only about 70%.The reason is because the number of rules in the rule base is different for each run.

6 Conclusion

In this work a combination of GA based ID for detecting different types of attacks was introduced. As we use only nine features to describe the data, its time of training is considerably reduced, thus providing high refreshing rate of the rule set. However the detection rate is not good for some runs because of the selection of cross over and mutation points in corresponding operations is random. The linear structure of the rule makes the detection process efficient in real time processing of the traffic data. The evaluation of our approach showed that the hybrid method of using discrete and continuous features can achieve a better detection rate of network attacks. In order to increase the detection rate, we select the features that are appropriate for each type of network attacks. That is also an added advantage. The modification of existing RBF networks using heuristic rules has obvious benefits when used in certain situations. The use of knowledge synthesis only makes sense when the available data is insufficient to build a reliable classifier. In such a situation it is advantageous to use heuristic rules to modify an existing RBF network to detect infrequently encountered input vectors that would otherwise be misclassified. However, care must be taken when applying the domain rules.
Further enhancements should be made by the rule learning technique using Radial Basis Network for detecting any unknown attacks.

7 Open Problem

In recent years, intrusion detection has emerged as an important technique for network security. Due to the large volumes of security audit data as well as complex and dynamic properties of intrusion behaviors, to optimize the performance of Intrusion Detection Systems becomes an important open problem. First, detection systems must be more effective, detecting a wider range of attacks with fewer false positives. Second, intrusion detection must keep pace with modern networks increased size, speed, and dynamics. Reducing the number of feature is a greatest task. If the optimal feature should not obtain, then the whole IDS is not efficient one. Hence here we use the decision dependent correlation to solve this open problem. Dynamic representation of rules and the knowledge representation using that rules is still an open problem. To solve this, in the future work we have planned to meliorate the system by training using the Radial Basis Function Networks without the data, only the domain knowledge exists.

References

[1] Bouzida.Y and Cuppens.F, "Detecting known an novel network intrusion", *IFIP/SEC 2006, Int. Information Security Conference Karlstad University*, Sweeden, May 2006, PP.123-129.

[2] Chittur.A, "Model Generation for an Intrusion Detection System using Genetic Algorithms", *High school Honors Thesis, http://www1.cs.columbia.edu/ids/publications/gaids-thesis01.pdf*, accessed in 2006.

[3] Gomez.J, Dasgupta.D, Nasaroui.D and Gonzalez.F, "Complete expression Trees for evolving Fuzzy classifiers system with Genetic Algorithms and Applications to Network Intrusion Detection". *In proceedings of NAFIPS-FLINT Joint Conference, New Orleans*, LA, June 2002, PP.469-474.

[4] Gong.R.H, Zulkernine.M, Anolmaesumi.P, "A software Implementation of a Genetic Algorithm Based approach to Network Intrusion Detection", *Proceedings of the SNPD/SAWN' 05*, Aug 2005, PP.19-27.

[5]Gonzalez.F, Gomez.J, Kaniganti.M, and Dasgupta.D, "An evolutionary approach to generate fuzzy anomaly signatures", *In proceedings 4th Annual IEEE Information Assurance workshop*, West point, NY, June 2003 PP.251-259..

[6] Grosan.C, Abraham.A, Chis.M,"Computational Intelligence for light weight Intrusion Detection systems", *Int.Conference on Applied computing, IADIS'06, San Sebastia*n, Spain, May 2006, PP.538-542.

[7] Hofmann.A and Sick.B, "Evolutionary optimization of radial basis function networks for Intrusion Detection", *In proceedings of Int.Joint.Conf. Neural Networks, Portland*, OR, Vol 1, July 2003, PP.415-420.

[8] KDD cup 1999 data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[9] Lascov.P, Dussel.P, Schafer.C and Riek.K, "Learning Intrusion Detection: Supervised or Unsupervised? ", *CIAP: International conference on image analysis and processing* No.13, cagliari, Italy, Vol.3617, Sep 2005, PP.50-57.

[10] Liao.Y and Vemuri.V.R, "use of k-nearest neighbour classifier for intrusion detection", *Int. J. of Computer Security*, Vol.21, No.5, Oct 2002, PP.439-448.

[11] MIT Lincoln Laboratory DARPA Intrusion Detection Evaluation [Online]. Available: http://www.ll.mit.edu/IST/ideval/index.html.

[12] Selvakani.S, Rajesh.R.S, "Improving ID performance using GA and NN", *International Journal of Computer Aided Engineering and Technology*, Sep 2008, Vol.13, N0.1/2/3.

[13] Wang.L, Yu.G, Wang.G and Wang D, "Method of evolutionary neural network based intrusion detection", *In proceeding of Int.Conf.Info-tech and Info-net*, Beijing, China, Oct 2001, Vol 5, PP.13-18.