Molded Intrusion Detection enforcing RBF

Journal:	*Transactions on Neural Networks*
Manuscript ID:	TNN-2009-P-1651
Manuscript Type:	Paper
Date Submitted by the Author:	10-Sep-2009
Complete List of Authors:	Kandeeban, Selvakani; Francis Xavier Engineering College, Dept Of Computer Applictaions
Keywords:	Radial Basis Function, KDD99 Cup Set, Encoding, Correlation Measure

Molded Intrusion Detection enforcing RBF

Selvakani S Kandeeban
Professor, Department of Computer Applications
Francis Xavier Engineering College
Tirunelveli, Tamilnadu, India
sselvakani@hotmail.com

R.S. Rajesh
Reader, Dept of Computer Science & Engineering
Manonmanium Sundaranar University
Tirunelveli, Tamilnadu, India
rs_rajesh@yahoo.co.in

Abstract—**Computers and other technologies are automating processes and increasing the ability to communicate. The rapid growth has brought greater benefits, as well as introducing new problems and challenges. The potential damage to computer networks keeps on increasing due to the growing reliance on the Internet and more extensive connectivity. Intrusion detection systems (IDSs) have become an essential component of computer security to detect attacks which occur despite the best preventative measures. A problem with current intrusion detection systems is that they have many false positive and false negative events. In this approach, effort has been made to learn new attacks and to detect previously learned attacks in a network data stream, and in autonomously improving its analysis over time. This detection method consists of the following three important phases: the feature reduction by mutual Information Correlation, framing the rule set based on genetic algorithm and the training done by Neural Network - Radial Basis Function network, with out data but the domain knowledge exists. Experiments were done by SPSS and WEKA. Experimental results found that Genetic Algorithm is highly successful in detecting known attacks, neural networks are found to be more effective to detect unknown attacks. The proposed methods outperform previous work in detecting both known and new attacks. In this research work KDD 99 data sets have been used without modification.**

Keywords- Feature extraction, Correlation Measure, DNA Encoding, Radial Basis Function, KDD99 Cup Set.

I. INTRODUCTION

Advances in technology and communication infrastructure have also greatly increased the ability to share information faster and more efficiently. The importance of computer is further enhanced by increased usage of the internet. The rapid growth has brought great benefits, as well as introducing new problems and challenges. Since the development of computers, computer security was always a concern. The concern over security grew as computers starting being networked. Threats change as fast as both technology and business change. Adaptation and improvisation are the key features of a security system. No hardware or software element can ever be immune from security weaknesses.

The modern internet user engages in a number of applications that require secure transactions when transferring information and funds which can be everyday internet banking, shopping or attaining access to password protected websites etc…. As the number of secure transactions increases so does the number of sophisticated attacks on systems that store precious corporate and personal data.

While firewalls gives some protection, but they does not provide full protection and still it is needed to be complimented by an intrusion detection system. The purpose of intrusion detection is to help computer systems prepare for identification and prevent any possible attacks.

The remainder of the paper is organized as follows. Section II deals with the computer security issues and the problem statement. Related works are also discussed. Section III provides the methodology of the work. Experiments and results are reported in Section IV and Conclusions are drawn in Section V.

II. COMPUTER SECURITY ISSUES

A. *Security Issues*

First, Annual reports from the Computer Emergency Response Team (CERT/CC, 2005) indicate a significant increase in the number of computer security incidents. Fig 1 depicts the rise in computer security incidents with six incidents reported in 1988 and 1, 37,529 in 2003 [1].

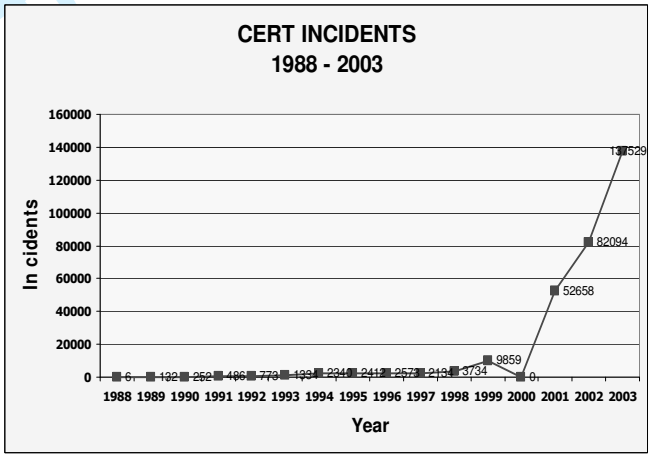

Figure 1. CERT Reported incidents per Year

This report showed that in 1995 there were approximately 250 attempted break-ins into federal computer systems in USA. Out of these attacks there were 64% (about 160) was successful. And it is estimated that the number of attacks will be doubled every year. Not only are these attacks becoming more numerous but also becoming more sophisticated.

There is still lot of things to be done before Intrusion Detection Systems are working satisfying enough. Here are the biggest issues:

• The biggest problem with Intrusion Detection Systems is that they are reactive, not proactive.

• In anomaly detection there is a problem when there is a small change in a user's behaviour. This happens sometimes, and will the Intrusion Detection System then alert this as an attack/misuse?

• Anomaly intrusion detection systems are still not working satisfying enough in a dynamic environment where there are big changes in the user behaviour.

• The need for more effective systems that can detect close up to 100% of attack methods without the high rate of false-positives.

• The need for security components that are resilient, and that can respond intelligently to attacks and have countermeasures.

• Current intrusion detection systems have limited response mechanisms that are inadequate given the current threat[2]. While intrusion detection system research has focused on better techniques for intrusion detection, intrusion response remains principally a manual process.

B. Problem Statement

Most IDS perform monitoring of a system by looking for specific "signatures" of behavior. However, using current methods, it is almost impossible to develop comprehensive-enough databases to create alarm of attacks. This is due to three main reasons. First, these signatures must be hand-coded. Attack signatures which are already known are coded into a database, against which the IDS uses to check current behavior. This system may be imagined as being very rigid. Second, because there is a theoretically infinite number of methods and variations of attacks, an infinite size database would be required to detect all possible attacks. This, of course, is not feasible. Also, any attack that is not included in the database has the potential to cause great harm. Another problem is that current methods are likely to raise many false alarms.

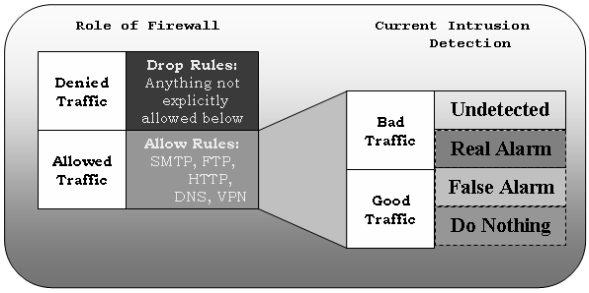

Figure 2. Firewall and Current IDS

This work offers a thorough analysis of current IDS models. Fig 2 explains clearly the difference between the firewall and the current IDS. The benchmark for this problem provided by the Defense Advanced Research Projects Agency (DARPA) and the International Knowledge Discovery and Data Mining Group (KDD) have been studied and investigated. This benchmarks and the experience of prior researchers are utilized to create IDS that is capable of learning attack behavior and is able to identify new attacks without system update.

C. Background Study

Anderson [1] first proposed that audit trails should be used to monitor threats in 1980. The importance of such data had not been comprehended at that time and all the available system security procedures were focused on denying access to sensitive data from an unauthorized source.

Chittur [4] extended this idea by using GA for anomaly detection. 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 accurate empirical behavior model from training data. The biggest limitation of this approach was the difficulty of establishing the threshold value, possibly leading to a high false alarm rate when used to detect novel or unknown attacks.

Gomez [5] proposed a linear representation scheme for evolving fuzzy rules using the concept of complete binary tree structures. GA is used to generate genetic operators for producing useful and minimal structure modification to the fuzzy expression tree represented by chromosomes. This approach however, required time consuming training.

Balajinath and Raghavan [3] used GA to learn individual user behavior. Active user behavior is predicted by GA based on past observed user behavior and used to detect intrusion. In both approaches, the training process is time consuming and they can only be used to detect anomalous behaviors at the host level.

Shazzad *et al. [*8] proposed a hybrid features selection method by combining Correlation based Features Selection (CFS), SVMs, and GAs. GA is used to generate subsets of features from the given features set, which is then evaluated by CFS and SVMs to pick the best features set. Shazzad *et al.* combined three different approaches and they were able to reduce the number of features from 41 to 12 for the DARPA dataset. However, they did not reach an optimal solution in terms of the number of features because GA might not evolve towards a good solution and it requires many parameters. Moreover, their approach is complicated (using three different approaches), making the modification and enhancement process difficult.

Kim *et al.* [7] proposed a features selection method identical to the previous method, but they used GA techniques to obtain the optimal features set and the optimal parameters for a kernel function of SVMs. This method suffers from the same problems that we mentioned earlier in terms of the heavy computation cost and optimal solution possibility.

So far the literature survey revealed the challenges already addressed as follows:

- Large number of False Negatives
- Large number of False Positives
- Alerts do not contain enough information
- Not finding higher level new attack patterns
- Less speed due to the inclusion of more features

- Have a poor real time performance
- Are difficult to extend with new techniques

To solve these unsolved problems, in this research methodology, three phased molded Intrusion detection mechanism was provided so as to minimize the false positives and to increase the performance. Features were reduced based on Correlation and optimized rules were developed using Genetic Algorithm and new attack patterns were detected by learning the patterns from the Neural Network Radial Basis Function Networks as shown in Fig 3.

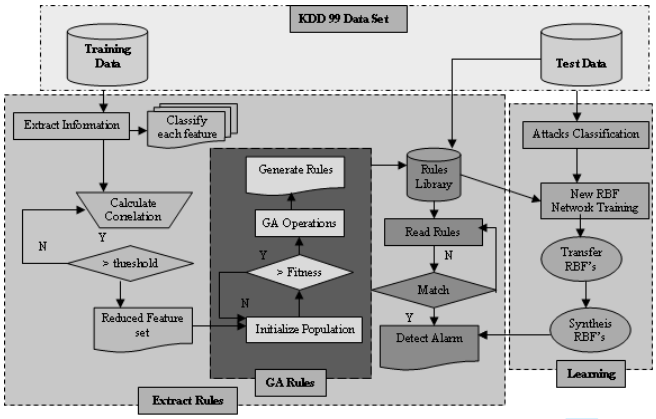

Figure 3. System Flow

D. KDD99 Cup Set

A proper dataset must be obtained to facilitate experimentation. In our experimental environment, it was difficult to obtain real-life datasets due to limitations of network size and limited external access. Unfortunately, usable datasets are rarely published as they involve sensitive information such as the network architecture, security mechanisms, and so on. Thus, in this work, we rely on the publicly available KDD Cup 99 intrusion detection dataset.

The KDD Cup 99 [6] intrusion detection data set is comprised of several files which account for the various connections detected on a host located on a simulated military network. The intrusion logs were generated in 1998 as part of the DARPA Intrusion Detection Evaluation Program. The nine weeks' worth of raw TCP dump data got split into a seven-week training data set and the remaining logs were turned into test data. 22 different types of attacks were identified.

Attacks fall into one of four categories:

- *Denial of Service (dos):* Attacker tries to prevent legitimate users from using a service.
- *Remote to Local (r2l):* Attacker does not have an account on the victim machine, hence tries to gain access.
- *User to Root (u2r):* Attacker has local access to the victim machine and tries to gain super user privileges.
- *Probe:* Attacker tries to gain information about the target host.

Each record consists of 41 attributes and one target. The target value indicates the attack name.

- The data has 4,898,431 records in the dataset.
- 3,925,650 (80.14%) records represent attacks that fall into one of the four mentioned above categories.
- Total 22 attacks were identified.
- 972,781 (19.85%) records of normal behavior were found.

III. MOLDED INTRUSION DETECTION

A. Feature Selection using Correlation Measure

Feature extraction in knowledge and date engineering is the process of identifying and removing as much of the irrelevant and redundant information as possible. Regardless of whether a machine learning algorithm attempts to select features itself or ignores the issue, feature extraction prior to learning can be beneficial. Reducing the dimensionality of the data reduces the size of the hypothesis space and allows algorithms to operate faster.

In this work, we present a new approach that effectively removes irrelevant features from the ranked feature list based on the mutual information between each feature and the decision variable. We obtain the ranked lists of features by using a simple forward selection hill climbing search, starting with an empty set and evaluating each feature individually and forcing it to continue to the far side of the search space. Redundant features are removed through the pair wise decision dependent correlation analysis.

The methodology is summarized as follows:

1) *Generate feature set R from the ranked list of features*
2) ***For each*** *feature for each type of attack, calculate the mutual information between the feature* X_i *and the decision Y,* $I(Y;X_i)$
3) *Updating relevant features set R by comparing the mutual information* $I(Y_i;X_i)$
4) ***if*** $I(Y;X_i) \geq \delta x$ ***then*** $R \leftarrow R + \{ X_i \}$
5) *where* δx *is the threshold which is user defined*
6) *Create working Set W by copying R*
7) *Set goal Set G = null*
8) ***While*** $e(G) < \delta_2$ *do*
9) ***If*** *W = null* ***then*** *break*
10) *Choose* $X_k \in W$ *that subjects to*
11) *Mutual information where*
    a. $I(Y;X_k) \geq I(Y;X_l)$ *for all* $l \neq k$, $X_l \in W$
12) *Correlation Measure*
    a. $Q_y(X_k,X_n) \leq Q_y(X_m,X_n)$ *for all* $m \neq k$, $X_n \in G$
    b. $W \leftarrow W - \{X_k\}$
    c. $G \leftarrow G + \{X_k\}$
13) ***End*** *Loop*
14) *Obtain a feature subset from the intersection of all the attacks subset*

The goal of the feature selection algorithm is to select the minimum set of features that are strongly related to the desired variable and have least redundancy among them. It consists of two functional modules. The first one focuses on removing irrelevance. It depends on a user defined threshold δ, to determine which feature is relevant to the final decision. In this part of the algorithm, irrelevant features are removed from

the original feature set. The second part focuses on eliminating redundancy from the features to be selected.

For each loop, the feature X_k is chosen which satisfies two conditions simultaneously. The first one is that the feature X_k should be the most relevant one compared with the rest of features in the working set. The second one is that feature X_k should have the least correlation with all the features in the working set W.

The goal of the feature selection algorithm is to select the minimum set of features that are strongly related to the desired variable and have least redundancy among them. It consists of two functional modules. The first one focuses on removing irrelevance. It depends on a user defined threshold δ, to determine which feature is relevant to the final decision. In this part of the algorithm, irrelevant features are removed from the original feature set. The second part focuses on eliminating redundancy from the features to be selected.

For each loop, the feature X_k is chosen which satisfies two conditions simultaneously. The first one is that the feature X_k should be the most relevant one compared with the rest of features in the working set. The second one is that feature X_k should have the least correlation with all the features in the working set W.

The next step is to obtain a rule set using these features for Intrusion Detection using Genetic Algorithm.

B. *Framing rules using Genetic Algorithm*

By analyzing the dataset, rules will be generated in the rule set. These rules will be in the form of an 'if then' format as follows. The pseudo code for the generation of rule set using Genetic algorithm is as follows:

```
1) Generation t := 0, Experience pool e:=NULL, done:=false
2) Initialize population p(t)
3) WHILE (!done)  DO
4)       FOR each rulebase r in population p(t)
5)              e := Evaluate(r)
6)              r := CreateAndDeleteRules(r, e)
7)              FOR i: 1 to MAX_EPISODES
8)                     e := Evaluate(r)
9)                     r := DistributeReward(r, e)
10)             END FOR
11)             Evaluate(r)
12)      END FOR
13)      done := CheckStoppingCondition(p(t))
14)      IF (!done)
15)             p(t+1) := Select(p(t))
16)             p(t+1) := Crossover(p(t+1))
17)             t := t + 1
18)      END IF
19) END WHILE
```

The first aspect is to learn the set of rules that makes up the solution. This involves searching through the rule space for good rules and GA, a stochastic search procedure based on the principles of natural evolution, is suitable for tackling this aspect. However, during the search process, GA does not guarantee that the rules in the solution are mutually exclusive. In this situation, the strength of the rule can be used to distinguish the 'good' rules from the 'bad' rules. The strength of a newly created rule may be set to a default value, but this should be modified according to its performance, which in turn is measured by the reward from the evaluator.

Each rule is an *if-then* clause, which contains a "condition" and an "outcome". The features are connected using the logical AND operations and compose the "condition" part of a rule. The feature "Attack name" is used in the "outcome" part, which indicates the classification of a network record when the "condition" part of a rule is matched. The fitness function will assign a fitness value for each rule. The fitness value will be a predetermined value.

C. *Training without data using RBF*

The KDD rule set is divided into four types U2R, R2L, Dos and Probe Training and Testing Set. The network was trained and the accuracy was examined with the training set. The hRex algorithm was used to assign hidden units to the output classes. The modified RBF network had its hidden – to – output weights and biases recalculated as shown in Fig 4. The modified network was retested on the training data set and the accuracy noted which is given as follows:

1) Given the number of the hidden nodes (centers) J is chosen, the learning algorithm is formulated as follows:

- *a. Find the positions of centers $\{\boldsymbol{w}_j\}$. This can be done by the following procedure:*
- *b. Choose randomly J instances $\boldsymbol{x}_j$ and use them as the positions of the centers $\{\boldsymbol{w}_j\}$*
- *c. All the remainders of the instances (training patterns) are assigned to a class j of the closest centre $\boldsymbol{w}_j$, and the locations of each center are calculated again using for example k-nearest neighbor method.*
- *d. The above steps are repeated until the locations of the centers stop changing.*

2) Calculate the output from each hidden neuron as a function of a radial distance from the input vector to the radial center. Calculated distance between the center and the input vector is passed through a non-linear 2 mapping Gaussian function.

3) Weights $\{b_{jk}\}$ for the output layer are calculated using methodologies as in MLP, using back propagation. The output from the output node can be expressed as

$$Z_k = \frac{\sum_{j=1}^{J} b_{jk} y_j}{\sum_{j=1}^{J} y_j} \qquad (1)$$

where b_{jk} – the weight on the connection from the hidden node j to the output node k, y_j - the output from the hidden node j

4) Calculate the error between the network's output and the target output and if the error of the network's output is more than the desired limit then the number of the hidden units arechanged and all the steps are repeated again.

The RBF neural networks can generate well to new patterns only if they stem from the same distribution of input patterns. If the task is very different they do not generalize well.

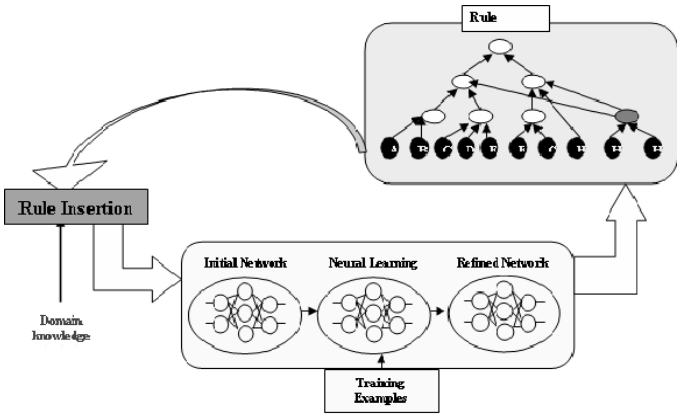

Figure 4. RBF training Knowledge

With additional rules these results could be improved. The existing rules cover a relatively simple input to output space mapping that was common to most machine behaviour (generic) and was easy to produce from the knowledge acquisition process. Overall, an improvement of 25% was made.

IV. EXPERIMENTS AND RESULTS

Experiments were done using SPSS [9] statistical analysis packages for feature extraction and Weka2 (Waikato Environment for Knowledge Analysis) has been [10] used for performing Genetic algorithm and Neural Network RBF Function. It is a powerful open-source Java-based machine learning workbench that can be run on any computer that has a Java run time environment installed. Weka brings together many machine learning algorithms. Weka requires input data to be in the form of Attribute-Relation File Format (ARFF).

The Training data set is separated in two subsets:

- Data containing only normal traffic and data containing only attacks.
- The subset with normal traffic contains 97,277 records. We randomly select 10% of the first subset.
- The subset containing only attacks has 3, 96,744 records. This subset is divided in three subsets containing:
  - ❖ 2,80,790 records with Smurf attack data
  - ❖ 1,07,201 records with Neptune attack
  - ❖ All remaining attack types with 8,752 records.
- The two totals of normal traffic and attacks are joined together in a training set with 14,904 records.

In this experiment, we then assign the training and test sets based on the distribution of the attack instances.

A. *Analysis of Results*

For Information gain and correlation based attribute selection, the selected attributes have a ranking. For example, the DoS attack detection results shown in Table I indicate that attribute 5 is more important than attribute 23 which is also more important than attribute 3, and so forth.

TABLE I. RANKED LIST OF FEATURES

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

Based on the above results, the best nine features for each attack have been selected which is shown in TABLE II.

TABLE II. SELECTED FEATURES FOR DIFFERENT CATEGORIES OF ATTACKS

Attack type	Selected Features
DOS	Count, service, dst_bytes, logged_in, dst_bytes_same_src_port_rate, srv_count, protocol_type, dst_host_count, src_bytes
Probe	Src_bytes, service, dst_bytes_diff_srv_rate, dst_bytes, rerror_rate, count, flag, dst_host_srv_diff_host_rate, same_srv_rate
U2R	Service, root_shell, dst_host_srv_count, duration, num_file_creations, dst_host_count, dst_host_same_src_port_rate, srv_count, dst_host_srv_diff_host_rate, src_bytes
R2L	Service, dst_host_srv_count, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, is_guest_login, srv_count, dst_bytes, dst_host_count, count, src_bytes

Based on the above results for DoS attacks, two hundred rules appears to be an adequate population size. Using this population size, the performance of the GA after 50 iterations and for all attack categories is given in TABLE III. It is seen that the performance does not increase significantly by increasing the number of iterations. Our GA outperforms the winning result for probe and U2R, and is not significantly worse for R2L and DoS attack categories. These results highlight the advantage of using a reduced and relevant feature set. The performance is comparable to the winning performance and at the same time being more efficient.

TABLE III. PERFORMANCE RESULTS FOR ALL ATTACKS AFTER 50 ITERATIONS AND USING 200 RULES

	Percentage Correct			
	DoS	***Probe***	***U2R***	***R2L***
Training Data Set	98.55	96.52	97.88	97.79
Testing Data Set	98.66	98.74	97.96	96.43

The experimental results conclude that the proposed method yielded good detection rates when using the generated rules to classify the training data itself.

Fig 5 shows how the value of the fitness function changes as the GA progresses. The top line represents the fitness (or quality of solution) of the best individual in the population.

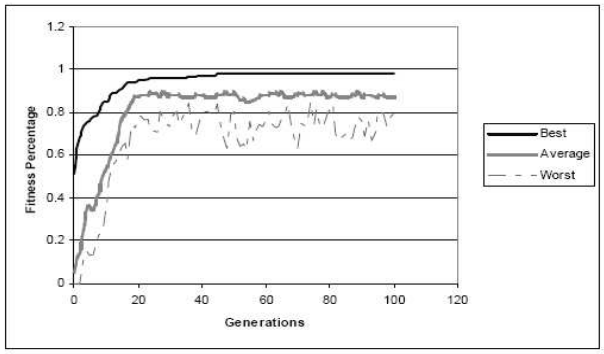

Figure 5. The evolution Process

Some of the rules which are extracted for DoS attacks is listed below:

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

Rule 3:

If Protocol_type =udp then
if service = private then
if Src_bytes = 28 then
if dst_bytes =0 then
if Logged_in =0 then Teardrop

TABLE IV. EVALUATION RESULT ON DOS USING RBF NETWORK

=== Evaluation result ===		
Scheme: RBFNetwork		
Relation: dos		
Correctly Classified Instances	917	99.8911 %
Incorrectly Classified Instances	1	0.1089 %
Kappa statistic	0.998	
Mean absolute error	0.0004	
Root mean squared error	0.0191	
Relative absolute error	0.1953 %	
Root relative squared error	6.3565 %	
Total Number of Instances	918	

"Correctly classified Instances" informed about the number of instances that were correctly classified and 100% is the best. Here it was 99.89%. The number of instances that were incorrectly classified is the failure percentage. Here the total number of instances was 918 and the instance incorrectly classified was 1 which was shown in the TABLE IV.

TABLE V. CONFUSION MATRIX FOR DOS ATTACKS

a	b	c	d	e	f	← Classified as
77	0	0	0	0	0	a = back
0	7	0	0	0	0	b = land
0	0	251	0	0	0	c = neptune
0	1	0	16	0	0	d = pod
0	0	0	0	566	0	e = smurf
0	0	0	0	0	0	f =teardrop

A Confusion matrix as shown in TABLE V is used to represent the result of testing. The advantage of using this matrix is that it not only tells us how many got misclassified but also what misclassifications occurred.

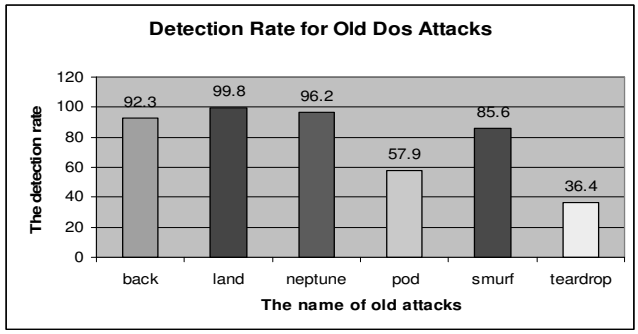

Figure 6. Detection rate for old DoS attacks

Out of the Six Old Dos Attacks, only 3 attacks namely back attack, land attack, and Neptune attacks have more than 90% detection rate as depicted in the bar diagram Fig.6. However, other attacks have the detection rate of less than 50%. The pod attack, smurf attack, and teardrop attack are the attacks mainly using ICMP. The packets using ICMP are included as smaller part in the DARPA Training Set than the DoS attacks using UDP or TCP. Thus, the information about the packets using ICMP cannot sufficiently influence the correlation, adopts the concept of the information entropy. Similarly, the information of the neptune attack is not enough to be reflected because most packets in neptune attack are destined to the telnet port, but the large number of packets destined to the telnet port are also contained in the data of negative class.

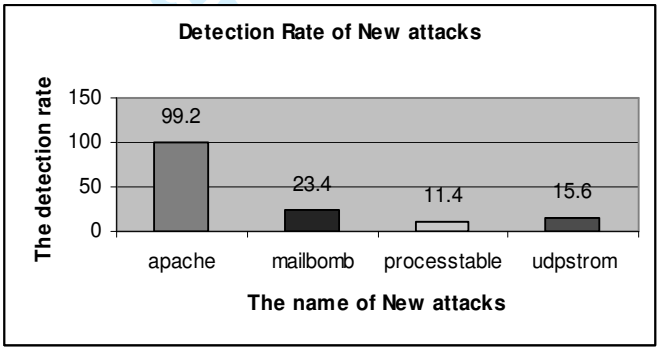

Figure 7. Detection rate for new DoS attacks

The fig 7 represents the detection rates of the new kinds of DoS attacks. As shown in the graph, the apache-2 attack is

detected for 100% detection rate in spite of new kinds of attack. This is because the patterns of the encoded data for apache-2 attack have similar patterns to the old DoS attacks. However the other attacks such as the mailbomb attack, process table attack, and UDP storm are rarely detected because the patterns of the encoded data are very different from the patterns of the old DoS attacks.

B. Limitations and Future Work

- Due to the probable existence of interdependency among three or more features, the learning algorithm may pick up some inaccuracy in its classification.
- Even though all these requirements are currently supported through manually generated heuristic rules which can be further upgraded to become even more autonomous and intelligent.
- For DoS, U2R and Probe attacks, it has shown better results, however R2L attacks remain difficult to identify.
- The generated rules were biased to the training data set. This issue may be resolved by carefully selecting either the number of generations in the training phase.

Future enhancements will surely resolve all the above said limitations.

Although some new attacks were predicted successfully, this does not mean that all new attacks would be detected. This is because the different attributes that are used for evaluation do not correspond to the whole raw traffic. Much more research should be performed in this direction to find out an optimistic approach that may summarize all the needed traffic without information loss. This is so restrictive since it is not so obvious to detect new attacks if we consider a new environment with new applications that are not considered during the learning step. However, we propose to always build a learning data base where all known intrusions and normal traffic are taken into account in the environment that is considered.

V. CONCLUSION

Selecting a good feature is very important because it gives a significant contribution to the intrusion detection system in terms of accuracy of detection. Nowadays, most of the researchers use only the selected features in their research without mentioning the reason of selecting the feature in predicting the intrusion activity. The previous researchers also did not mention the influence of the selected features to the system developed in their research. Understanding the relation and influence of the feature before using them may help to reduce the possibilities of selecting unnecessary feature which may give an impact in detecting the intrusion activity especially fast attack since fast attack is used in the initial stage of an attack where attackers use it to begin their attack inside the network. Therefore identifying the attacker earlier may help the administrator to overcome further damage caused by the attacker. In this research, we manage to reveal the influence of the feature in predicting the detection of the intrusion especially fast attack using correlation approach.

The main conclusion of the presented research is the production of rules for encoding DNA Genetic algorithm. We are pursuing the creation of rules to detect complex network intrusions to maximize the utility of the system, and to produce a dynamic rule base capable of detecting new attack signatures. Nine network features including both categorical and quantitative data fields were used when encoding and deriving the rules. A simple but efficient and flexible fitness function is used to select the appropriate rules.

The use of knowledge synthesis only makes sense when the available data is sufficient 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. Unless the domain rules can cover a large proportion of the synthesized input to output space it is unlikely that all the necessary centre positions will be moved into appropriate locations. This effect will reduce the effectiveness of the new centers. Fortunately, because of the local characteristics of the RBF network the existing centers will be relatively unaffected and should still be able to classify with the same accuracy before knowledge synthesis occurred. The existing rules cover a relatively simple input to output space mapping that was common to most machine behavior and was easy to produce from the knowledge acquisition process.

Overall Intrusion Detection Systems are the most useful asset in system security if it is implemented and configured correctly.

REFERENCES

[1] Anderson. J. P. “Computer Security Threat Monitoring and Surveillance.” Technical Report, James P Anderson Co., Fort Washington, Pennsylvania, 1980.

[2] Bace, Rebecca, “Intrusion Detection.” Macmillan Tech. Pub. Indianapolis, IN, 2000.

[3] Balajinath.B, Raghavan.S.V., “Intrusion Detection through Learning Behaviour model” , Int. J. Of Computer Communications, Vol.24, No.12, PP.1202-1212.

[4] 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, 2006.

[5] Gomez.J, Dasgupta.D., “CIDS: An agent based Intrusion Detection system”, Computers and Security, Vol.24, No.5, PP.386-398, 2005.

[6] KDD Cup 1999 Data.: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[7] Kim, D.S., Park, J.S., “Network-based Intrusion Detection with Support Vector Machines.” Lecture Notes in Computer Science, Vol. 2662, Springer-Verlag, Berlin Heidelberg , 747–756, 2003.

[8] Shazzad.K.M., Dong Seong kim, “Toward modeling Light weight Intrusion Detection system through Correlation Based Hybrid Feature Selection” , In SKLOIS Conference on Information Security and Cryptology, CISC 2005, Beiging, China, PP.279-289, 2005.

[9] SPSS 11.0 © SPSS Inc, www.spss.com

[10] WEKA software, Machine Learning, http://www.cs.waikato.ac.nz/ml/weka/, The University of Waikato, Hamilton, New Zealand.