Intrusion Detection using an Improved Competitive Learning Lamstar Neural Network

V.Venkatachalam [†] and S.Selvan [††],

[†]H.O.D Dept. of CSE, Erode Sengunthar Engineering College ,Thudupathi,Erode ,Tamilnadu ,India .
†† H.O.D Dept of IT, PSG College of Technology ,Coimbatore,Tamilnadu ,India

Summary

Computer systems vulnerabilities such as software bugs are often exploited by malicious users to intrude into information systems. With the recent growth of the Internet such security limitations are becoming more and more pressing. One commonly used defense measure against such malicious attacks in the Internet are Intrusion Detection Systems (IDSs). Due to increasing incidents of cyber attacks, building effective intrusion detection systems (IDS) are essential for protecting information systems security, and yet it remains an elusive goal and a great challenge. We developed an Intrusion Detection System using LAMSTAR neural network to learn patterns of normal and intrusive activities and to classify observed system activities. we further investigate the time taken for training and testing, generate Confusion matrix with KDD CUP 99 data using simulation tool and compare it with five classification techniques (Gaussian Mixture, Radial Basis Function, Binary Tree Classifier, SOM, and ART). The results indicate that LAMSTAR exhibit high accuracy at the cost of long training time.

Keywords:
Intrusion Detection System, LAMSTAR, Gaussian Mixture, SOM, Radial Basis Function, Binary Tree Classifier, ART.

1.INTRODUCTION

Intrusion detection and prevention generally refers to a broad range of strategies for defending against malicious attacks. Intrusion detection can be categorized into misuse detection and anomaly detection[1]. Misuse typically is a known attack, e.g., a hacker attempting to break into an email server in a way that IDS has already trained. A misuse detection system tries to model normal and abnormal behavior from known attacks. It works by comparing network traffic, system call sequences, or other features of known attack patterns. An anomaly is something out of the ordinary, e.g., abnormal network traffic which is actually caused by unknown attacks. An anomaly detection system models normal behavior and identifies a behavior as abnormal (or anomalous) if it is sufficiently different from known normal behaviors.

We developed an Intrusion Detection System using LAMSTAR neural network to learn patterns of normal and intrusive activities and to classify observed system activities. we further investigate the time taken for training and testing, generate Confusion matrix using simulation tool and compare it with five classification techniques (Gaussian Mixture, Radial Basis Function, Binary Tree Classifier, SOM, and ART)

This paper is organized as follows: section 2 gives some theoretic background about LAMSTAR neural network. Section 3 presents the details about KDD cup 99 dataset used for testing and training the Intrusion Detection system and the cost matrix used to calculate cost per example. Section 4 gives details about the various algorithms and the parameters used to obtain the confusion matrix and cost per example. Section 5 summarizes the obtained results with comparison and discussions. The paper is finally concluded with the most essential points

2. LAMSTAR

A Large Scale Memory Storage and Retrieval (LAMSTAR) network is proposed in [2,3] by combining SOM modules and statistical decision tools. It was specifically developed for application to problems involving very large memory that relates to many different categories (attributes) where some data is exact while the other is fuzzy and where for a given problem some categories might be totally missing [2]. Large Scale Memory Storage and Retrieval (LAMSTAR) network research, which targets large-scale memory storage and retrieval problems. This model attempts to imitate, in a gross manner, processes of the human central nervous system (CNS) concerning storage and retrieval of patterns, impressions, and sensed observations including processes of forgetting and recollection. It attempts to achieve this without contradicting findings from physiological and psychological observations, at least in an input/output manner. Furthermore, it attempts to do so in a computationally efficient manner using tools of neural networks, especially Self-Organizing-Map based

Manuscript received February 5, 2007
Manuscript revised February 25, 2007

(SOM) network modules, combined with statistical decision tools. Its design was guided by trying to find a mechanistic neural network-based model for very general storage and retrieval processes involved. This general approach is related to Minsky's idea that the human brain consists of many agents, and a knowledge link is formed among them whenever the human memorizes an experience. When the knowledge link is subsequently activated, it reactivates the mental agents needed to recreate a mental state similar to the original. The LAMSTAR network employs this general philosophy of linkages between a large number of physically separate modules that represent concepts, such as time, location, patterns, etc., in an explicit algorithmic network.

The LAMSTAR network has been successfully applied in fields of medicine (diagnosis)[4,5,6], engineering (automotive fault detection) and multimedia information systems[7]. Whereas the LAMSTAR design addresses large-scale memory retrieval problems, we use LAMSTAR concepts to processes of storage and retrieval, interpolation and extrapolation of input data, and the use of reward-based correlation-links between modules to detect intrusions. In this modified LAMSTAR network, each Kohonen SOM module represents a class of sub-patterns. The model assumes that the input patterns have been separated into sub-patterns before entering the SOM module . The network is thus organized to assign each neuron to a class of neurons (i.e., one SOM module) that best corresponds to the input sub-pattern. This SOM configuration yields very rapid matching with good error tolerance, and is capable of generalization. Arrays of correlation links (C-links) connect the modules using coefficients determined by the statistical correlations between the various patterns considered. A coordinated activation of neurons between the various modules allows the network to recreate (interpolate) complex patterns and make associations.

3.DATA SET & COST MATRIX

3.1 DATA SET

We trained and tested our system using KDD Cup's 99 dataset [8,9] which covers four categories of attacks: Denial of Service (DoS) attacks:deny legitimate requests to a system, e.g.syn flood, User-to-Root (U2R) attacks: unauthorized access to local super user(root) privileges, e.g.various buffer overflow attacks, Remote-to-Local (R2L) attacks:unauthorized access from a remote machine, e.g. guessing password, and Probing: surveillance and other probing, e.g. port scanning. The 1999 Defence Advanced Research Projects Agency (DARPA) Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. Table I gives the details of KDD CUP 99 data.

Table I : KDD Cup 99 Training and Testing data

Dataset Label	DOS	PROBE	U2R	R2L	Total Attack	Total Normal
Training data	391458	4107	52	1126	494020	97277
Testing data	229853	4166	228	16189	311029	60593

3.1.1 Feature Extractions and Preprocessing

The input data to the neural network must be in the range [0 1] or [-1 1]. Hence preprocessing and normalization of data is required. The kdd-cup format data is preprocessed. Each record in kdd-cup format has 41 features, each of which is in one of the continuous, discrete and symbolic form, with significantly varying ranges. Based on the type of neural nets, the input data may have different forms and so needs different preprocessing. Some neural nets only accept binary input and some can also accept continuous-valued data. In Preprocessor, after extracting kdd-cup features from each record, each feature is converted from text or symbolic form into numerical form. For converting symbols into numerical form, an integer code is assigned to each symbol. For instance, in the case of protocol_type feature, 0 is assigned to tcp, 1 to udp, and 2 to the icmp symbol. Attack names were first mapped to one of the five classes, 0 for Normal, 1 for Probe, 2 for DoS, 3 for U2R, and 4 for R2L.

Two features spanned over a very large integer range, namely src_bytes [0, 1.3 billion] and dst_bytes [0, 1.3 billion]. Logarithmic scaling (with base 10) was applied to these features to reduce the range to [0.0, 9.14]. All other features were boolean, in the range [0.0, 1.0]. Hence scaling was not necessary for these attributes.

3.1.2 Normalization

For normalizing feature values, a statistical analysis is performed on the values of each feature based on the existing data from KDD Cup's 99 dataset and then acceptable maximum value for each feature is determined. According to the maximum values and the following simple formula, normalization of feature values in the range [0,1] is calculated.

If (f > MaxF) Nf=1; Otherwise Nf = (f / MaxF)

F: Feature f: Feature value MaxF: Maximum acceptable value for F Nf: Normalized or scaled value of F

Another simple way to Normalize the data is to use SOM toolbox of the MATLAB software. In this paper the following MATLAB commands were used to normalize the data.

```
sD=som_read_data('kddcup.data')
sD=som_normalize(sD,'var',1:4)
sD=som_normalize(sD,'log',5:6)
sD=som_normalize(sD,'var',7:41)
sD=som_normalize(sD,'var',1:41)
```

3.2 COST MATRIX

A cost matrix (C) is defined by associating classes as labels for the rows and columns of a square matrix: in the current context for the KDD dataset, there are five classes, {Normal, Probe, DoS, U2R, R2L}, and therefore the matrix has dimensions of 5×5. An entry at row i and column j, C(i,j), represents the non-negative cost of misclassifying a pattern belonging to class i into class j. Cost matrix values employed for the KDD dataset are defined elsewhere in [10]. These values were also used for evaluating results of the KDD'99 competition. The magnitude of these values was directly proportional to the impact on the computing platform under attack if a test record was placed in a wrong category. A confusion matrix (CM) is similarly defined, in that row and column labels are class names: a 5×5 matrix for the KDD dataset. An entry at row i and column j, CM(i,j), represents the number of misclassified patterns, which originally belong to class i yet mistakenly identified as a member of class j. Given the cost matrix as predefined in [10] and the confusion matrix obtained subsequent to an empirical testing process, cost per example (CPE) was calculated using the formula,

$$CPE = \frac{1}{N} \sum_{i=1}^{5} \sum_{j=1}^{5} CM(i,j)*C(i,j)$$

where CM corresponds to confusion matrix, C corresponds to the cost matrix, and N represents the number of patterns tested. A lower value for the cost per example indicates a better classifier model. Comparing performances of classifiers for a given attack category is implemented through the probability of detection along with the false alarm rate, which are widely accepted as standard measures. Table II shows the cost matrix used for scoring entries

Table II The cost matrix used for scoring entries

	Normal	Probe	DOS	U2R	R2L
Normal	0	1	2	2	2
Probe	1	0	2	2	2
DOS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

In the confusion matrices above, columns correspond to predicted categories, while rows correspond to actual categories.

The software tool LNKnet, which is a publicly available pattern classification software package [11], was used to simulate pattern recognition and machine learning models. The SOM and LAMSTAR was simulated using JNNS [12] software tool.

3.2.1 Standard metrics for evaluations of Intrusions (attacks)

We evaluated the performance of various IDS systems based on the Detection Rate :detecting normal traffic from attack and recognizing the known attack type False Positive Rate : mis-detecting attack[13] . Table III shows the standard metrics for evaluation of Intrusions

$$\text{Detection rate} : \frac{\text{No. of samples classified correctly}}{\text{No. of samples used for training}}$$

$$\text{False alarm Rate} : \frac{\text{False Positives}}{\text{Total no. of normal connections}}$$

Table III : Standard metrics for evaluations of Intrusions(attacks)

Confusion matrix (Standard Metrics)		Predicted Connection label	
		Normal	Intrusions(Attacks)
Actual Connection label	Normal	True Negative(TN)	False Alarm(FP)
	Intrusions (Attacks)	FalseNegative(FN)	Correctly detected Attacks (TP)

4. ALGORITHMS APPLIED TO INTRUSION DETECTION

4.1 Gaussian Mixture

The Gaussian Mixture classifier[14] can perform better than a Gaussian classifier when classifier distributions are not unimodal Gaussian. Different simulations were performed by changing various parameters like, each class has its own Gaussian mixture, all classes share a single set of tied Gaussian mixtures, diagonal covariance, full matrices covariance, separate variance for each gaussian . The simulation result with parameters ,each class has its own Gaussian mixture and diagonal covariance gives the better cost per example 0.2796. Table IV shows the Confusion Matrix obtained for Gaussian mixture IDS

Table IV Confusion Matrix for Gaussian mixture IDS CPE = .2796

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	59969	423	190	5	6	98.97
Probe	195	3876	95	0	0	93.03
DOS	18015	9002	202822	9	5	88.24
U2R	145	25	1	52	5	22.8
R2l	13950	650	5	30	1554	9.6
%correct	64.99	27.73	99.85	54.16	98.98	

The top left entry in the confusion matrix shows that 59969 of the actual "normal" test examples were predicted to be normal by this entry. The last column indicates that in total 98.97% of the "normal" examples were recognized correctly. The bottom row shows that 64.99% of test examples said to be normal were indeed "normal" in reality. From the last column, we can obtain the average detect rate of 62.52%. The false positive rate for Normal class is 100-64.99 =35.01 %.

4.2 Radial Basis Function

Radial Basis Function classifiers [15] calculate discriminant functions using local Gaussian functions. A total of six simulations were performed using the RBF algorithm .Each simulation used initial clusters created using K-means algorithm: there were 8,16,32,40 ,64 and 75 clusters each in different output classes. Weights are trained using least-square matrix inversion to minimize the squared error of the output sums given the basis function outputs for the training patterns. During training and testing variance are increased to provide good coverage of the data .For each simulation using the RBF, cost per example for the test dataset was calculated. The model with 64 clusters performed best with the cost per example equal to .3801. Table V shows the Confusion Matrix obtained for Radial Basis Function IDS.

Table V Confusion Matrix for RBF IDS CPE = .3801

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	60030	263	290	8	2	99.07
Probe	350	3804	8	3	1	91.31
DOS	37050	20163	172620	15	5	75.10
U2R	190	20	2	16	0	7.01
R2l	7282	7800	200	0	907	5.6
%correct	57.22	11.87	99.71	38.09	99.12	

4.3 SOM

For SOM [16,17] the training algorithm can be summarized in four basic steps. Step 1 initializes the SOM before training. Best matching unit (BMU) is the neuron, which resembles the input pattern most. On Step 2, best matching unit is determined. Step 3 involves adjusting best matching neuron (or unit) and its neighbors so that the region surrounding the best matching unit will represent the input pattern better. This training process continues until all input vectors are processed. Convergence criterion utilized here is in terms of *epochs*, which defines how many times all input vectors should be fed to the SOM for training. The epochs ranging from 100 to 130 gives the better performance cost per example .1996. Table VI shows the confusion matrix obtained for SOM IDS

Table VI Confusion Matrix for SOM IDS CPE =.1996

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	56945	3448	190	8	2	93.98
Probe	1200	2679	280	5	2	64.30
DOS	7964	996	220889	3	1	96.10
U2R	101	68	9	49	1	21.49
R2l	11295	2993	7	0	1894	11.7
%correct	73.47	21.30	99.78	75.38	99.68	

4.4 Binary Tree classifier

The binary decision tree classifier[17] trains and tests very quickly. It can also be used to identify the input features which are most important for classification because feature selection is part of the tree-buliding process. Two different training options were used 1.Expand tree until there are no errors. 2.Stop Expansion Early .Two different testing options were used 1.Full tree for testing, 2.Maximum number of nodes during testing .The simulation with the parameters Expand tree

until there are no errors for training and Full tree for testing gives the best cost per example .1841 . Table VII shows the confusion matrix obtained for the Binary tree classifier IDS

Table VII Confusion Matrix for Binary Tree Classifier IDS CPE=.1841

Predicted Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	58430	1475	678	7	3	96.43
Probe	419	3247	492	6	2	77.94
DOS	7159	995	221694	4	1	96.45
U2R	97	99	1	31	0	13.59
R2l	8063	1000	7054	0	72	0.44
%correct	78.78	47.63	96.42	64.58	92.30	

4.5 ART

Stability and plasticity of ART[18] nets and the capability of clustering input patterns based on the user controlled similarity between them, made such nets more appropriate for using in IDSs, rather than the other types of unsupervised nets including SOM, for classifying network traffic into normal and intrusive attack. Accordingly, we used two types of unsupervised ART nets, ART-1 and ART-2. For ART1 and ART2 the optimum value for the vigilance parameter and number of epochs determines the performance. ART1 with vigilance value of .92, ART2 with vigilance value of .97 and epochs ranging from 90 to 120 gives the better result. Cost per example .1830. Table VIII shows the confusion matrix obtained for ART IDS

Table VIII Confusion Matrix for ART IDS CPE=.1830

Predicted Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	58894	699	998	1	1	97.19
Probe	54	4103	8	0	1	98.48
DOS	4060	2606	223183	3	1	97.09
U2R	57	127	3	41	0	17.98
R2l	12360	1540	459	1	1829	11.29
%correct	78.08	45.21	99.34	89.13	99.83	

4.6 LAMSTAR

Using the LAMSTAR[19,4,5] algorithm, different clusters were specified and generated for each output class. Simulations were run having 2,4,8,16,32,40,64 clusters. Clusters were trained until the average squared error difference between two epochs was less than 1%.

4.6.1 LAMSTAR IDS DESIGN

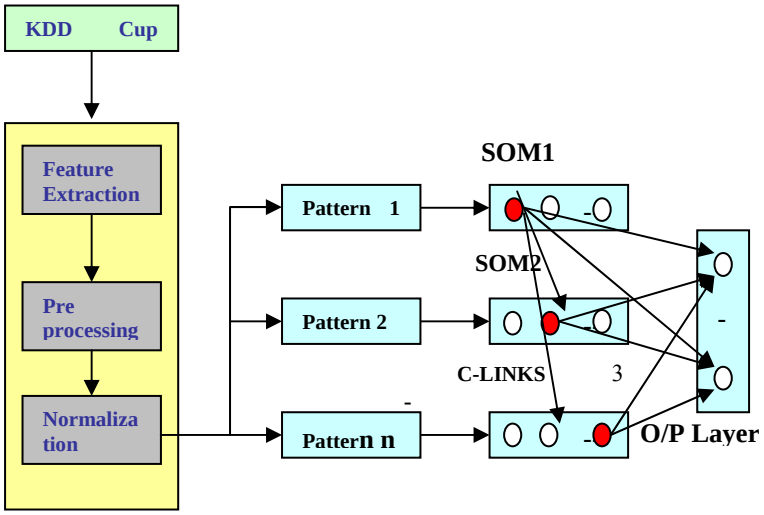

Figure 1 : Modified LAMSTAR architecture

A modified LAMSTAR network used for intrusion detection is as shown in Figure 1. The model reads in KDD cup 99 data sends it first to the feature extraction module which extracts 41 features of the data and sends it to preprocessing module. The preprocessing module converts the 41 features into a standardized numeric representation Normalization block reads the preprocessed data and normalizes the data into a format required by the SOM's. The normalized input pattern was split into sub patterns (basic features 9, content features 13, traffic 9, and others 10)[8]. Each sub pattern given to one SOM module. This SOM configuration yields very rapid matching with good error tolerance, and is capable of generalization.

Between SOM modules, connections are established using correlation links. The correlation links distribute information between various modules. The training data contains 22 attack patterns and normal patterns. The SOM modules are trained using this pattern. The coordinated activation of neurons between the various modules allows the network to detect intrusions.

The input pattern is stored as a real vector x given by:

$$X = \left[x^{1T}, \ldots x^{iT}, \ldots x^{mT} \right]^{T} \quad \text{(1)}$$

To store data concerning the i'th category of the input pattern, each sub-pattern xi is then channeled to the corresponding i'th SOM module. A winning neuron is determined for each input based on the similarity between the input vector xi and weight vectors wi (stored information). For a sub-pattern xi, the winning neuron is determined by the minimum Euclidean distance between xi and wi:

$$||x_i - w^i_{winner}|| = \min || x_i - w^i_{winner} || \ \forall \ k \quad \ldots.(2)$$

where xi - input vector in i'th SOM module
winner -index of the winning neuron
w^i_{winner} - winner weight vector in ith SOM module
k - a number of neurons(stored patterns)in ith SOM module
||-|| - Vector Euclidean distance :

$$||x-w|| = \sum_{i=1}^{n} (w_i - x_i)^2$$

where n - dimension of sub vectors x and w

The SOM module is a Winner-Take-All[12] network where only the neuron with the highest correlation between its input vector and its correspondence weight vector will have a non-zero output. The Winner-Take-All feature also involves lateral inhibition such that each neuron has a single positive feedback onto itself and negative feedback connections to all other units.

$$O^i_j = \begin{cases} 1 & \text{for } ||x_i - w^i_{winner}|| < ||x_i - w^i_j|| \ \forall \ \text{winner} \neq j \\ 0 & \text{otherwise} \end{cases} \quad \ldots.(3)$$

where O^i_j - output of neuron j in ith SOM module
w^i_{winner} - winning weight vector in ith SOM module
winner - index of winning neuron in ith SOM module

The neuron with the smallest error determined is declared the winner and its weights w_{winner} are adjusted using the Hebbian learning law, which leads to an approximate solution:

$$w^i_{winner}(t+1) = w^i_{winner}(t) + \alpha . (xi(t) - w^i_{winner}(t)) \quad \ldots.(4)$$

The adjustment in the LAMSTAR SOM module is weighted according to a pre-assigned Gaussian hat neighborhood function Δ(winner,j):

$$w^i_j(t+1) = w^i_{winner}(t) + \Delta(winner, j). \alpha .(xi(t) - w^i_{winner}(t)) \quad \ldots.(5)$$

Where $w^i_j(t+1)$ - new weight of the neighbour neuron j from winning neuron winner
$\Delta(winner, j)$. –Neighborhood define as gaussian hat
α -Learning rate a slowly decreasing function of time, initial weights are assumed with random values. The learning rate is updated by, $\alpha(t+1) = 0.5\,\alpha(t)$

4.6.2 Training phase

The training of the SOM modules are done as described below

SOM modules are trained with sub-patterns derived from the KDD cup 99 data. Given an input pattern x and for each xi sub-pattern to be stored, the network inspects all weight vectors wi in the i'th SOM module. If any previously stored pattern matches the input sub-pattern within a preset tolerance (error ε), the system updates the proper weights or creates a new pattern in the SOM module. The choice of ε 's value depend on the size of the cluster. The following expression is used to calculate the value of ε:

$$\varepsilon = \underset{x \in cli}{Max} (\text{dist}(x,ci))/10$$ where ci is the cluster center and cli is the cluster i. It stores the input sub-pattern xi as a new pattern, xi = wji, where index j is the first unused kji neuron in i'th SOM module. If there are no more 'free' neurons, the system will fail, which means either the preset tolerance has to be increased to include more patterns in the same cluster of already stored patterns, or more neurons have to be added on the i'th SOM module.

Correlation links C-links among SOM modules are created as follows. Individual neurons represent only a limited portion of the information input. Sub-patterns are stored in SOM's and the correlation links between these sub-patterns are established in such a way that the informations are distributed between neurons in various SOM modules and correlation links. Even if one neuron fails only a little information is lost since the information is spread among SOM's and correlation links. Correlation-link coefficient values C-link are determined by evaluation distance minimization to determine winning neurons, where a win activates a count-up element associated with each neuron and with its respective input-side link. During training sessions, the values of C-links are modified according to the following simple rule (reward)

$$C^{i,l}_{k,j}(\text{new}) = C^{i,l}_{k,j}(\text{old}) - \beta_{\text{reward}}(C^{i,l}_{k,j}(\text{old}) - C_{\text{Max}}) ,$$

for $C^{i,l}_{k,j}(\text{old}) \neq 0$, 1 otherwise ….(**6**)

$C^{i,l}_{k,j}$ - correlation link between k'th neuron in i'th SOM module and l'th neuron in j'th SOM module

β_{reward} - reward coefficient ,initially value is assumed with random values . $\beta_{\text{reward}}(t+1) = .5\ \beta_{\text{reward}}(t)$

To keep link-weights within a reasonable range, whenever the highest of all weights reaches a certain threshold all link-weights to that SOM are uniformly reduced by the same proportion, for example 50%. Additionally, link-weights are never reduced to zero or the connection between the two neurons will be lost permanently. If the correlation link between two sub-patterns already exists, namely, $C_{k,l}^{i,j} > 0$ (a result from previous training), the formula of equation 6 updates (increases) the analyzed C-link. If there are no correlations ($C_{k,l}^{i,j} = 0$), the system creates new C-link with initial value $C_{k,l}^{i,j} = 1$.

4.6.3 Detection phase

The sub-patterns from input pattern is selected and the correlations with stored sub-patterns in each SOM module is examined. For example, one i'th SOM module could have previously stored source IP address, and will correlate any given input i'th sub-pattern and determine if there is a match or not. The Intruder packet is detected by means of its C-links. Once all the winning neurons are determined, the system obtains all correlation-links coefficient values among all SOM modules. The output SOM layer (Figure 1), with which all C-links are inter-connected, will determine whether the input pattern is an intruder packet or a normal packet. This model achieved the lowest cost per example value .1020. Table IX shows the confusion matrix obtained for LAMSTAR IDS

Table IX Confusion Matrix for LAMSTAR IDS CPE=.1027

Predicted / Actual	1	2	3	4	5	%correct
1	60411	140	37	4	1	99.69
2	56	4103	6	0	1	98.48
3	1603	186	228060	3	1	99.21
4	99	54	8	66	1	28.94
5	7519	985	1015	0	6670	41.20
%correct	86.68	75.03	99.53	90.04	99.94	

5. EXPERIMENTAL RESULTS

Best performing instances of all classifiers developed through the KDD testing data set[20] . For a given classifier, its detection rate, false alarm rate, Training time and Testing time performance on a specific attack category was recorded. Simulation results are presented in Table X. Both detection rate, false alarm rate, Training time and Testing time are indicated for each classifier and each attack category. The false alarm rate and detection rate of all the classifiers were recorded. TABLE X shows the comparison of various classifiers.

TABLE X Comparison of Detection Rate, False Alarm Rate, Training Time and Testing Time of various classifiers

	Cost Per Example		Normal	Probe	DOS	U2R	R2L
Gmix	0.2796	DR	98.97	93.03	88.24	22.8	9.6
		FAR	35.01	72.27	0.15	45.84	1.02
		Training Time	40s	15s	60s	5s	10
		Testing Time	28s	10s	45s	5s	12s
RBF	0.3801	DR	99.07	91.31	75.10	7.01	5.6
		FAR	42.78	88.13	0.29	61.91	0.88
		Training Time	41s	14s	55s	5s	11s
		Testing Time	31s	10s	40s	5s	9s
SOM	0.1996	DR	93.98	64.30	96.10	21.49	11.70
		FAR	26.53	78.70	0.22	24.62	0.32
		Training Time	41s	16s	58s	6s	12s
		Testing Time	29s	11s	29s	5s	11s
Binary Tree	0.1841	DR	96.43	77.94	96.45	13.59	0.44
		FAR	21.22	52.37	3.58	35.42	7.70
		Training Time	39s	14s	53s	6s	11s
		Testing Time	30s	12s	29s	6s	9s
ART	0.1830	DR	97.19	98.48	97.09	17.98	11.29
		FAR	21.92	54.79	0.66	10.87	0.17
		Training Time	40s	16s	51s	5s	12s
		Testing Time	28s	13s	29s	4s	8s
LAMSTAR	0.1027	DR	99.69	98.48	99.21	28.94	41.20
		FAR	13.32	24.97	0.47	9.96	0.06
		Training Time	47s	16s	60s	6s	15s
		Testing Time	28s	13s	28s	5s	9s

CONCLUSION

In this paper, we proposed a novel method based on LAMSTAR neural network for intrusion identification. A simulation study was performed to

assess the performance of a set of machine learning algorithms on the KDD 1999 Cup intrusion detection dataset. Simulation results demonstrated that all the algorithms performed well for NORMAL ,DOS and PROBE classes except SOM which shows poor result for PROBE class. For the U2R class SOM and LAMSTAR gives a better performance than the other algorithms For R2L class LAMSTAR gives the better result. Overall LAMSTAR gives better performance at the cost of long training time

REFERENCES

[1] .K.Anup Ghosh et.al, " *Study in Using Neural Networks for Anomaly and Misuse Detection* ", Proceedings of the 8th SENIX Security Symposium, pp 131-142, August 1999, Washington, D.C.

[2] Abirami Muralidharan, J.Patrick Rousche, " *Decoding of auditory cortex signals with a LAMSTAR neural network* ", Neurological Research, Volume 27, pp. 4-10, January 2005

[3] D.Graupe and H. Kordylewski, "*A Large Memory Storage and Retrieval Neural Network for Adaptive Retrieval and Diagnosis* ", International Journal of Software Engineering and Knowledge Engineering, volume 8, pp.115-138, 1998.

[4] D.Graupe, "Principles of Artificial Neural Networks", pp. 191-222, World Scientific Publishing Co. Pte. Ltd., Singapore, 1997.

[5] H. Kordylewski, "*A Large Memory Storage and Retrieval Neural Network for Medical and Engineering Diagnosis/Fault Detection* ", Doctor of Philosophy's Thesis, University of Illinois at Chicago, TK-99999-K629, 1998.

[6] D.Graupe and H. Kordylewski, "*A Large Memory Storage and Retrieval Neural Network for Adaptive Retrieval and Diagnosis* ", International Journal of Software Engineering and Knowledge Engineering, volume 8, pp.115-138, 1998.

[7] S.Chang.et.al, "*An Active Multimedia Information System for Information Retrieval, Discovery and Fusion* ", International Journal of Software Engineering and Knowledge Engineering, volume 8, pp. 139-160, 1998.

[8] http://kdd.ics.uci.edu//databases/kddcup98/kddcup98.html

[9] Srilatha Chebrolu et.al, "*Feature deduction and ensemble design of intrusion detection systems",* Elsevier Journal of Computers & Security" Vol. 24/4, pp. 295-307, 2005.

[10] Itzhak Levin, KDD-99 Classifier Learning Contest LLSoft's Results Overview , "*SIGKDD Explorations. Copyright 2000 ACM SIGKDD"* , Volume 1, Issue 2, pp. 67 -75, January 2000.

[11] www.ll.mit.edu/SST/lnknet/

[12] www-ra.informatik.uni-tuebingen.de/software/JavaNNS/welcome_e.html

[13] Dae-Ki Kang, "*Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation*", Proceedings of the 6th IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY,2005.

[14] Jing Gao et.al, "*A Novel Framework for Incorporating Labeled Examples into Anomaly Detection"*, Proceedings of the Siam Conference on Data Mining, April 2006.

[15] Dima Novikov et.al, "*Anomaly Detection Based Intrusion Detection*" Proceedings of the Third IEEE International Conference on Information Technology: New Generations (ITNG'06), pp. 420-425.

[16] Khaled Labib and Rao Vemuri , " *NSOM: A Real-Time Network-Based Intrusion Detection System Using Self-Organizing Maps* " , *Networks and Security*, 21(1), Oct. 2002.

[17] Richard Lippmann, ***"****Passive Operating System Identification From TCP/IP Packet Headers"* published in the Proceedings of the Workshop on Data Mining for Computer Security (DMSEC), Lincoln Laboratory ,Massachusetts, 2003.

[18] Morteza Amini et.al, "Network-Based Intrusion Detection Using Unsupervised Adaptive Resonance Theory (ART)", Published in the Proceedings of the 4th Conference on Engineering of Intelligent Systems (EIS 2004), Madeira, Portugal, 2004.

[19] Liberios VOKOROKOS et.al, "*Intrusion detection system using self organizing map",* Acta Electrotechnica et Informatica , Vol. 6 No.1, pp.1-6, 2006

[20] Chaker Katar, "*Combining Multiple Techniques for Intrusion Detection",* International Journal of Computer Science and Network Security, VOL.6 No.2B, February 2006.

[18] Morteza Amini et.al, "Network-Based Intrusion Detection Using Unsupervised Adaptive Resonance Theory (ART)", Published in the Proceedings of the 4th Conference on Engineering of Intelligent Systems (EIS 2004), Madeira, Portugal, 2004.

[19] Liberios VOKOROKOS et.al, "*Intrusion detection system using self organizing map",* Acta Electrotechnica et Informatica , Vol. 6 No.1, pp.1-6, 2006

[20] Chaker Katar, "*Combining Multiple Techniques for Intrusion Detection",* International Journal of Computer Science and Network Security, VOL.6 No.2B, February 2006.

Dr. S. Selvan received the B.E. degree in Electronics and Communication Engineering and M.E. degrees in Computer Science. He received his Ph.D degree in Computer Science. His area of specializations are Soft computing, Information Processing, Computer Networks, Digital communication Techniques. He has presented 12 papers in International conference and 21 papers in National Conference . He has published 4 papers in International Journal and 12 Papers in National Journal. Presently working as Professor and Head of the Dept. IT in PSG College of Technology, Coimbatore

V. Venkatachalam rceived the B.E. degree in Electronics and communication Engineering from Bharathiyar University and M.S. degree in software systems from Birla Institute of Technology. He received M.Tech. degree in Computer Science from National Institute of Technology. His Research interest includes Network Security and Pattern recognition.He is currently pursuing his P.hd degree in Network Security. Presently working as Head of the Dept. CSE in Erode Sengunthar Engineering College ,Erode