International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2015): 6.391

Primary Factor Investigation for Decreasing the Computational Complexity of LAMSTAR DDoS

Dhanasekaran R[1], Balaji P[2]

[1]Assistant Professor of Computer Applications, K.S.Rangasamy College of Arts and Science (Autonomous), Tiruchengode, India

[2]Assistant professor of Computer Applications, K.S.Rangasamy College of Arts and Science (Autonomous), Tiruchengode, India

Abstract: ***Nowadays the security is very important for our Computer system networks. Here we use Distributed Denial of Services (DDoS) and it act as a ''second line of defense'' in a protected network and looking for threats and its data recorded in a computer. We developed LAMSTAR DDoS a neural network used to know methods of normal and intrusive activities, to classify observed system activities and compared the performance of LAMSTAR DDoS with other classification techniques using 5 classes of KDDCup99 data. LAMSTAR DDoS shows performance is better at a rate of high Computational Complexity, Training time and testing time, when compared to other classification techniques (Binary Tree classifier, RBF classifier, and Gaussian Mixture classifier). We reduce the Complexity of LAMSTAR DDoS by defining the dimension of the data using principal component analysis which in turn the training and testing time gets reduced with performance is same.***

Keywords: Binary Tree Classifier, Gaussian Mixture, Distributed Denial of Service, LAMSTAR, Radial Basis Function

1. Introduction

Computer security has become a critical issue with the rapid development of business and other transaction systems over the Internet. Intrusion detection is to detect intrusive activities while they are acting on computer network systems. There are two major intrusion detection techniques: misuse detection and anomaly detection [1]. Misuse detection discovers attacks based on the patterns extracted from known intrusions. Anomaly detection identifies attacks based on the deviations from the established profiles of normal activities. Activities that exceed thresholds of the deviations are detected as attacks. Misuse detection has low false positive rate, but cannot detect new types of attacks. Anomaly detection can detect unknown attacks, under a basic assumption that attacks deviate from normal behavior.

We developed a Distributed Denial of Service using LAMSTAR neural network to learn patterns of normal and intrusive activities, to classify observed system activities and other classification techniques using 5 classes of KDDCup99 data. LAMSTAR DDoS gives better performance at the cost of high Computational Complexity, Training time and testing time, when compared to other classification techniques (Binary Tree classifier, RBF classifier and Gaussian Mixture classifier). We further reduced the computational complexity of LAMSTAR DDoS by reducing the dimension of the data using principal component analysis which in turn reduces the, training and testing time with almost the same performance.

2. LAMSTAR

A Large Scale Memory Storage and Retrieval (LAMSTAR) network is proposed by combining SOM modules and statistical decision tools [2]. LAMSTAR 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. Large Scale Memory Storage and Retrieval (LAMSTAR) network research, which targets large-scale memory storage and retrieval problems. This model attempts to imitate the processes of the human Central Nervous System (CNS) concerning storage and retrieval of patterns and its impressions, and sensed observations including processes of forgetting and recollection. It attempts to achieve this without contradicting findings from physiological and psychological observations 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 (SOM) network modules are combined with statistical decision tools. And 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 [3] of linkages between a large number of physically separate modules that represent concepts, such as time, location, patterns, etc., in an explicit algorithmic network.

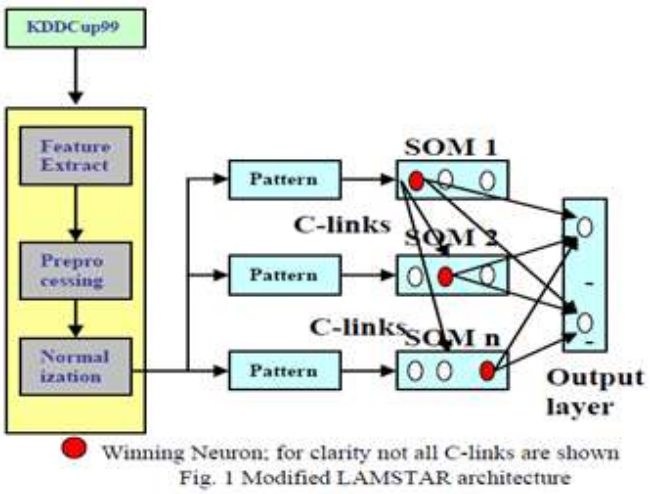

Fig. 1 Modified LAMSTAR architecture

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 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 [8]. 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. LAMSTAR DDoS Design

A modified LAMSTAR network used for intrusion detection [18] is as shown in Fig. 1. The model reads in KDDCup99 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). Each sub pattern is given to one SOM module. This SOM configuration yields very rapid matching with good error tolerance, and it 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[17].

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

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

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

$$||x^i - w^i{}_{\text{winner}} = min: ||X^i - W^i{}_k \;||\forall \; k \quad (2)$$

Where x - input vector in i'th SOM module winner -index of the winning neuron

$W^i{}_{\text{Winner}}$ - winner weight vector in ith SOM module

k - a number of neurons (stored pattern) in ith SOM module

||-|| - Vector Euclidean distance

$$||X - W\;|| = \sum_{i=1}^{i=n} (\, W_{t-}\, X^t \,)^2$$

Where n - dimension of sub vectors x and w

The SOM module is a Winner-Take-All [9] 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 features also involves lateral inhibition such that each neuron has a single positive feedback onto itself and negative feedback connections to all other units.

$$O^i{}_{j} = \begin{cases} 0 \\ 1 \text{ for } \| X^t - W^i\text{winner}\| < \|x^i - w^i{}_j\| \end{cases}$$

¥ winner ≠ 0 otherwise

Where

$O^i{}_j$ - output of neuron j in ith SOM Module

$W^i\text{winner}$ - Winning weight vector in the ith SOM Module

Winner - index of the winning neuron is the 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,

$$W^i\text{winner}(t+1) = W^i\text{winner}(t)_0 + \alpha(x^t(t) - W^i\text{winner}(t)) \quad (4)$$

α - Learning rate a slowly decreasing function of time and initial weights are assumed with random values. The learning rate is updated by, α (t+1) = 0.5α (t).

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

$$\boldsymbol{W}^i{}_j(t+1) = \boldsymbol{W}^i{}_j(t) + \Delta(\text{winner}, j).\alpha(\boldsymbol{x}^i(t) - \boldsymbol{W}^i{}_j(t) \quad (5)$$

Where $\boldsymbol{W}^i{}_j(t+1)$ - new weight of the neighbor neuron j from winning neuron.

Δ (winner, j) – neighborhood defined as Gaussian hat.

Training Phase

The training of the SOM modules are done as described below SOM modules are trained with sub-patterns derived from the KDDCup99 data. Given an input pattern x and for each x sub-pattern to be stored, the network inspects all weight vectors win the I'th SOM module. If any previously 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 depends on the size of the cluster. The following expression is used to calculate the value of ε

$$\varepsilon = \text{MAX}_{x\varepsilon.cli}.\ \text{dist}(x,ci) / 10 \quad (6)$$

Where ci is the cluster center and cli is the cluster i. It stores the input sub-pattern x as a new pattern, **x = w**, where index j is the first unused k 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 and 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 information's 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 between 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)

$$C_{k,j}^{i,l}(\text{new}) = C_{k,j}^{i,l}(old) - \beta_{reward}(C_{k,j}^{i,l}(old) - C \text{ Max), for } C_{k,j}^{i,l}(\text{old}) \neq 0,1 \quad (7)$$

Where $C_{k,j}^{i,l}$ - Correlation link between k'th neuron in the i[th] SOM module and the l'th neuron in j'th SOM.

β reward- reward coefficient, initially value is assumed with some random values. β reward(t+1) =.5. β reward (t).

To keep the link-weights within a reasonable range, whenever the highest of all weights reaches a certain threshold all link- weights to that the same proportion, for example 50%, uniformly reduces SOM 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, i result from previous training), the formula of equation 6 updates (increases) the analyzed C-link. If there are no correlations (C_{kj} = 0), the system creates new system C-link with initial value k" I = 1.

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 [15], 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 (Fig.1), with which all C-links are inter-connected, will determine whether the input pattern is an intruder packet or a normal packet.

Data Set & Cost

The KDDCup99 intrusion detection [9] datasets based on the 1999 DARPA initiative, which provides designers of intrusion detection systems (DDoS) with a benchmark on which to evaluate different methodologies. Our system has been trained and tested using KDDCup99 dataset [8],[10] which covers 22 attack types in the training data which are classified into 5 classes Denial of Service (DoS) attacks: deny legitimate requests to a system, e.g. CYN flood, User-to-Root (U2R) attacks: unauthorized access to local super user(root) privileges, e.g. various buffer overflow attacks, Remote-to- Local (112L) attacks: unauthorized access from a remote machine, e.g. guessing password, and Probing: surveillance and other probing, e.g. port scanning.

The 1999 Defense Advanced Research Projects Agency (DARPA) Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs [11]. The objectives are used to survey and evaluate research in intrusion detection. A standard set of data is to be audited, which includes a variety of intrusions simulated in a military network environment are provided. Table 1 gives the details of KDDCup99 data.

Table 1: KDDCUP99 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

4. 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 KDDCup99 format data is preprocessed. Each record in KDDCup99 format has 41 features, each of one, which is in 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 inputs and some can also accept continuous-valued data. In Preprocessor, after extracting KDDCup99 features from each record, each feature is converted from text or symbolic form into numerical form [12]. For converting symbols into numerical form, an integer code is assigned to each symbol. For example, in the case of protocol type features, 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 to Normal, 1 to Probe, 2 to DoS, 3 to U2R, and 4 to 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.

5. Normalizations

For normalizing feature values, a statistical analysis is performed on the values of each feature based on the existing data from KDDCup99 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) (8)

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

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

```
sD=som read-data('KDDCup99.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)
```

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 KDDCup99 dataset, there are five classes, {Normal, Probe, DDOS, 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 **KDDCup99** defined also in [11]. These values were also used for evaluating results of the **KDDCup99** competition. T 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 **KDDCup99** 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 [11] and the confusion matrix obtained subsequent to an empirical testing process, cost per example (CPE) was calculated using the formula,

$$\text{CPE}=\frac{1}{N}\sum_{i=1}^{5}\sum_{j=1}^{5}CM(i,j)*C(i,j) \qquad (9)$$

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 2 shows the cost matrix used for scoring entries.

Table 2: 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

The confusion matrices in the above, columns correspond to predicted categories, while rows correspond to actual categories. The software tool LNK net, which is a publicly available pattern classification software package, was used to simulate pattern recognition and machine learning models. The LAMSTAR was simulated using JNNS [13] software tool.

STANDARD METRICS FOR EVALUATIONS OF INTRUSIONS (ATTACKS)

We evaluated the performance of various DDoS systems based on the Detection rate: detecting normal traffic from attack and recognizing the known attack type False Alarm rate: mis-detecting attack [14]. Table 3 shows the standard metrics for evaluation of intrusions.

Table 3: Standard Metrics for Evaluations of Intrusions

Confusion Matrix (Standard Metrics)		Predicted Connection Label	
		Normal	Intrusions (Attacks)
Action Connection Label	Normal	True Negative (TN)	False Alarm (FP)
	Intrusions (Attacks)	False Negative (FN)	Currently Detected Attacks (TP)

6. Conclusion

An approach for detecting network intrusions using LAMSTAR Neural Network is proposed in this paper. The performance of LAMSTAR DDoS evaluated using KDDCup99 data and compared with three other classifiers. Simulation results demonstrated that all the algorithms performed well for NORMAL, DDoS and PROBE classes except Binary Tree, which shows poor result for PROBE class. For the U2R and R2L class LAMSTAR gives a better performance than the other algorithms. The performance of LAMSTAR DDoS is obtained at the cost of high training and testing time due to computational complexity. The computational complexity can be reduced by, training and testing time, Principal Component Analysis [16] was applied to KDDCup99 data and 13 important features were selected out of 41 features in the KDDCup99 data. Experimental results with 13 features show significant reduction in training and testing time due to the reduction in computation while keeping the detection rate and false alarm rate almost the same.

References

[1] A.K.Ghosh, A.Schwartzbard, "Study in Using Neural Networks for Anomaly and Misuse Detection", in Proc. 8th USENIX 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 scale memory (LAMSTAR) neural network for medical diagnosis", in Proc. 19th Annual International Conference of the IEEE, Volume 3, Issue 30, Oct-2 Nov 1997 Page(s): 1332 – 1335.
[7] S.K.Chang, D.Graupe, K.Hasegawa, H.Kordylewski, "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] Teuvo Kohonen , "The Self Organizing Map", in Proc. IEEE, Volume 78, No. 9, pp 1464 – 1480, September 1990.
[9] Srilatha Chebrolu, Ajith Abraham, Johnson P.Thomas, "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", Vol. 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.html.
[13] Dae-Ki Kang, "Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation", in Proc. 6th IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 2005.
[14] D. Nguyen, A. Das, G. Memik, and A. Choudhary , "Reconfigurable Architecture for Network Intrusion Detection Using Principal Component Analysis" In Proc. ACM/SIGDA 14th international symposium on Field programmable gate arrays , pp. 235 – 235, 2006.
[15] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier", In Proc. IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM'03), pp 172–179, Nov. 2003.
[16] I. T. Jolliffe, "Principal Component Analysis", Springer Verlag, NewYork, NY, third edition, July 2002.
[17] Jing Gao, Haibin Cheng, Pang Ming Tan, "A Novel Framework forIncorporating Labeled Examples into Anomaly Detection", in Proc. ofthe Siam Conference on Data Mining, April 2006.
[18] Dima Novikov, Roman V. Yampolskiy, Leon Reznik, "Anomaly Detection Based Intrusion Detection" in Proc. of the Third IEEE International Conference on Information Technology: New Generations (ITNG'06), pp. 420-425, 2005.

Author Profile

Dr. R. Dhanasekaran received his Bachelor of Science degree in Computer Science from Bharathiyar University in the year 2003 and Master degree in Computer Applications from Bharathiyar University in the year 2007. He has received his Ph.D in the field of Computer Networks from Anna University in the year 2017. He has published 2 papers in International Journal. He also attends 4 National and 1 International Conferences.

Mr. P. Balaji received his Bachelor of Science degree in Computer Science from Periyar University in the year 2004 and Master degree in Computer Applications from Bharathiyar University in the year 2007. He has received his M.Phil in Computer Science in the year 2013. He attends 2 National and 1 International Conferences.