Anomaly Intrusion Detection System Using Hierarchical Gaussian Mixture Model

M. Bahrololum and M. Khaleghi

Iran Telecommunication Research Center, PO Box 14155-3961, Tehran, Iran.

Summary

Intrusion Detection Systems have been widely used to overcome security threats in computer networks and to identify unauthorized use, misuse, and abuse of computer systems. Anomaly-based approaches in Intrusion Detection Systems have the advantage of being able to detect unknown attacks; they look for patterns that deviate from the normal behavior.

In this paper we proposed Hierarchical Gaussian Mixture Model (HGMM) a novel type of Gaussian Mixture which detects network based attacks as anomalies using statistical preprocessing classification. This method learns patterns of normal and intrusive activities to classify that use a set of Gaussian probability distribution functions. The use of Maximum likelihood in detection phase has used the deviation between current and reference behavior. HGMM is evaluated by dataset KDD99 without any special hardware requirements. We compare it with six classification techniques; Gaussian Mixture, Radial Basis Function, Binary Tree Classifier, SOM, ART and LAMSTAR to verify its feasibility and effectiveness. Experimental results show that this method is able to reducing the missing alarm, and can accurately predict probable attack behavior in IDS.

Keyword: Intrusion Detection System, Hierarchical Gaussian Mixture Model, Anomaly detection.

1. Introduction

In recent years, intrusion detection technologies are indispensable for network and computer security as the threat becomes a serious matter year by year. Therefore Intrusion Detection Systems (IDSs) inspect all inbound and outbound network activities and identify suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise the system [1].

IDSs are categorized into misuse detection and anomaly detection systems [2]. Misuse detection systems detect known attacks using pre-defined attack patterns and signatures, e.g., a hacker attempting to break into an email server in a way that IDS has already trained. Anomaly detection systems detect attacks by observing deviations from the normal behavior of the system. It works by comparing network traffic, system call sequences, or other features of known attack patterns. An anomaly is something out of the ordinary.

Various artificial intelligence techniques have been utilized in anomaly IDS; Data clustering is a common technique for statistical data analysis such as IDS, including machine learning [3], data mining, pattern recognition and neural networks [4].

We developed an Intrusion Detection System using Hierarchical Gaussian Mixture Model (HGMM) to learn patterns of normal and intrusive activities and to classify observed system activities. This method learns statistical regularities from packets that each group of the input sets is modeled the best possible probability distribution by a set of Gaussian probability distribution functions which are called Gaussian Mixtures. It is compared with six classification techniques; Gaussian Mixture, Radial Basis Function, Binary Tree Classifier, SOM, ART and LAMSTAR [5].

Manuscript received August 5, 2008.
Manuscript revised August 20, 2008.

The used data in our experiments originats from MIT's Lincoln Lab; a well known dataset. It was developed for intrusion detection system evaluations by DARPA and is considered a benchmark for IDS evaluations [6]. We perform experiments to classify the network traffic patterns according to 5-class taxonomy. The five classes of patterns in the DARPA data are normal, probe, denial of service, user to super-user, and remote to local.

This paper is organized as follows: Section 2 briefly introduces Intrusion Detection System. Section 3 gives some theoretic background about GMM and HGMM. Section 4 presents the details about KDD cup 99 dataset used for testing and training the Intrusion Detection system. Section 5 summarizes the obtained results with comparison; an evaluation experiment shows the performance of HGMM.

2. Intrusion Detection System

An intrusion detection system (IDS) monitors network traffic for suspicious activity and alerts the system or network administrator against malicious attacks. In some cases the IDS may also respond to anomalous or malicious traffic by taking action such as blocking the user or source IP address from accessing the network.

There are several ways to categorize an IDS [1]:

2-1. Misuse detection vs. Anomaly detection

In misuse detection, the IDS analyses the information it gathers and compares it to large databases of attack signatures. Essentially, the IDS looks for a specific attack that has already been documented. Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against. In anomaly detection, the system administrator defines the baseline, or normal, state of the network traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and look for anomalies.

2-2. Network-based vs. Host-based

In a network-based system, or NIDS, the every individual packet flowing through a network is analyzed. The NIDS can detect malicious packets that are designed to be overlooked by a firewall simplistic filtering rules. In a host-based system, the IDS examines at the activity on each individual computer or host.

3- Hierarchical Gaussian Mixture Model

In general, intrusion detection using Hierarchical Gaussian Mixture Model (HGMM) is the process of identifying the abnormal packets in the network. There are two phase in the process of HGMM. The first is the *training phase* that reference templates are generated by HGMM. So, the attacks have to be trained to the system. The features have been extracted from the sample data that is provided by traffic, in order to obtain the data for statistical modeling. Next phase is *detection phase*. During the detection phase, the input packet's deviation from the stored reference models is calculated and recognition decision is made as to which model suits that packet.

In this phase, it is evaluated the performance of various intrusion detection method based on the Detection Rate (DR) which is the number of samples classified correctly to the number of test set.

The two phases are shown in Fig 1.

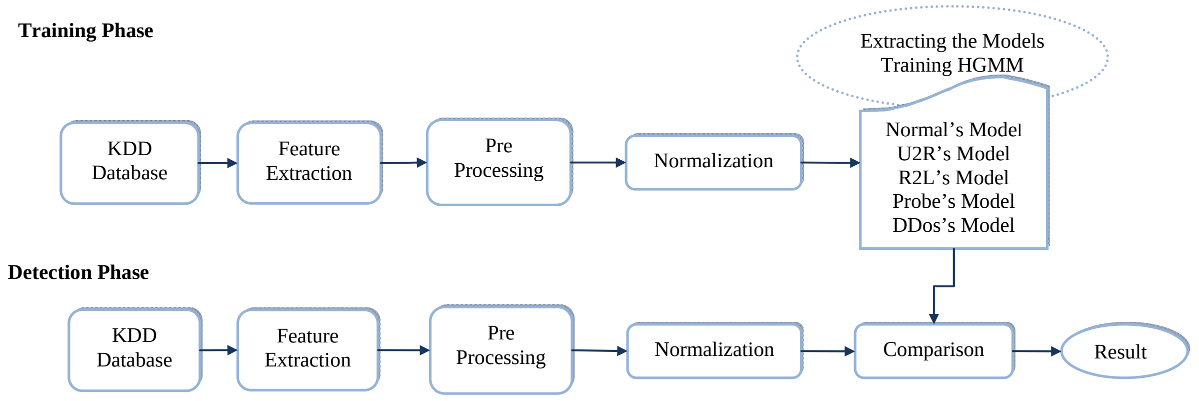

Figure1: Training and Detection phase in a HGMM

3.1. Overview of Gaussian Mixture Model

Mixture models are a type of density model that comprise of a number of component functions, usually Gaussian. The distribution of feature vectors was extracted from packets in the network.

A Gaussian Mixture Model (GMM) [7] is used to construct a Bayesian classification procedure on the observations and leads to the system behavior model. Parameters of mixture model are used by the Expectation Maximization (EM) algorithm. The next step consists of evaluating detection packets related to new system activities to detect deviations of the current from the reference behaviors.

A Gaussian mixture density is a weighted sum of M component densities, as is depicted in Fig. 2 and given by equation (1):

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} w_i b_i(\vec{x}) \qquad (1)$$

Where $\vec{x}$ is a D-dimensional random vector, $b_i(\vec{x})$, $i = 1, \dots, M$, are the component densities and w_i, $i = 1, \dots, M$, are the mixture weights. Each component density is a D-dimensional variable Gaussian function of the form,

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(\vec{x}-\vec{\mu}_i)'\Sigma_i^{-1}(\vec{x}-\vec{\mu}_i)\right\} \qquad (2)$$

In this formula, $\overline{\mu}_i$ is mean vector and Σ_i is covariance matrix.

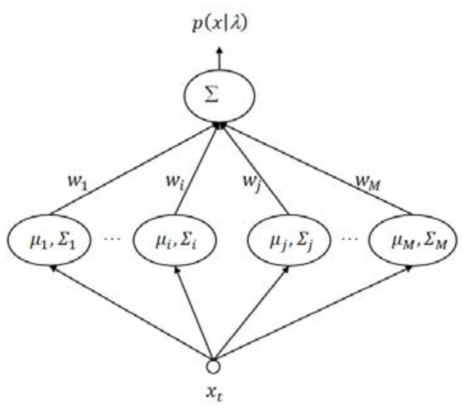

Figure 2: Shows the overview of the structure of GMM.

The mixture weights satisfy the constraint:

$$\sum_{i=1}^{M} w_i = 1 \qquad (3)$$

The complete Gaussian mixture density is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities. These parameters are collectively represented by the notation:

$$\lambda = \{w_i, \vec{\mu}_i, \Sigma_i\} \qquad i = 1, \dots, M \qquad (4)$$

For intrusion detection, each classification of attacks is represented by a GMM and is referred to by model λ. The GMM can have different forms depending on the choice of covariance matrices. The model can have one covariance matrix per Gaussian component (nodal covariance), one covariance matrix for all Gaussian components in a model (grand covariance), or a single covariance matrix shared by all models (global covariance). The covariance matrix can also be full or diagonal. Full covariance matrices are usually deemed unnecessary by the assumption of statistical independence for the mixture components. This allows simplification by using only the diagonal of the covariance matrix. In this paper, nodal and diagonal covariance matrices are used for our models.

The aim of ML estimation is to find the model parameters which maximize the likelihood of the GMM, given the training data. For a sequence of T training vectors $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_T\}$ the GMM likelihood can be written as

$$p(X|\lambda) = \prod_{t=1}^{T} p(\vec{x}_t|\lambda) \qquad (5)$$

The basic idea of the EM algorithm is beginning with an initial model λ, to estimate a new model $\bar{\lambda}$ if the probability of $(X|\bar{\lambda})$ is greater than or equal to $(X|\lambda)$, $p(X|\bar{\lambda}) \geq p(X|\lambda)$. The new model then becomes the initial model for the next iteration and process is repeated until some convergence threshold is reached.

On each EM iteration, the following reestimation formulas are used which guarantee a monotonic increase in the model's likelihood value:

Mixture weights:

$$\overline{w}_i = \frac{1}{T}\sum_{t=1}^{T} p\left(i|\vec{x}_t, \lambda\right) \qquad (6)$$

Means:

$$\bar{\mu}_i = \frac{\sum_{t=1}^{T} p\left(i \middle| \bar{x}_t, \lambda\right) \bar{x}_t}{T\bar{w}_i} \tag{7}$$

Variances:

$$\bar{\delta}_i = \frac{\sum_{t=1}^{T} p(i|x_t, \lambda) x_t^2}{T\bar{w}_i} - \bar{\mu}_i^2 \tag{8}$$

The posteriori probability for class *i* is given by:

$$p(i|x, \lambda) = \frac{w_i b_i(x_t)}{\sum_{k=1}^{M} w_k b_k(x_t)} \tag{9}$$

The goal of attack model training is to estimate the parameters of the GMM, λ, which in some sense best matches the distribution of the training feature vectors. Two critical factors in training a GMM is selecting the order of the mixture (M) and initializing the model parameters prior running to the EM algorithm.

3-1. K-Means Algorithm

K-means algorithm [8],[9] was originally designed for vector quantization codebook generation. It is an unsupervised clustering algorithm, which represents each cluster with its mean. Assuming a set of vectors $X = \{\bar{x}_1, \bar{x}_2, \bar{x}_3, ..., \bar{x}_T\}$ which is divided into *M* clusters represented by their mean vectors $\{\bar{\mu}_1, \bar{\mu}_2, \bar{\mu}_3, ..., \bar{\mu}_M\}$, the objective of the K-means algorithm is to minimize the total distortion given by equation (10):

$$\textit{Total distortion} = \sum_{t=1}^{T} \sum_{i=1}^{M} \|\bar{x}_t - \bar{\mu}_i\| \tag{10}$$

K-means follows an iterative approach to meet the objective. In each successive iteration, it redistributes the vectors in order to minimize the distortion. Although originally meant for codebook generation, it can be adapted to train GMM. The procedure is outlined as follow:

(a) To initialize, *M* random vector from the training set are selected as the means of *M* clusters.

(b) Each vector $\bar{x}_t, 1 \le t \le T$ is assigned to cluster *j*, if, $\|\bar{x}_t - \bar{\mu}_j\| < \|\bar{x}_t - \bar{\mu}_k\|, \forall k \neq j, 1 \le j, k \le M$.

(c) The new mean of a cluster is obtained by calculating the mean of all the vectors assigned to that particular cluster.

(d) The weights are determined by calculating the proportion of the vectors assigned to the cluster and the covariance matrix is the covariance matrix of the assigned vectors.

Steps (b) and (c) are repeated till the clusters are stable, i.e., the distortion is minimized. When the clusters are stable, the weights and covariance matrix can be found as described in step d. It is to be noted that in each iteration, *K*-means estimates the means of all the *M* clusters.

3.2. Training of the HGMM

The Hierarchical Gaussian Mixture Model can be considered as a general GMM. All Gaussian components in local GMM belong to a root node that is also represented by a Gaussian component. During training, the observation vectors of each local Gaussian component are pooled to estimate the new parameter of the root nodes, and then these new estimates also help to adjust the parameters of components that have no sufficient observations.

The structure of HGMM is shown in Figure 3. With the local GMM density, the fine structure of local area can be described more precisely. Thus, the HGMM can describe the distribution with higher precision

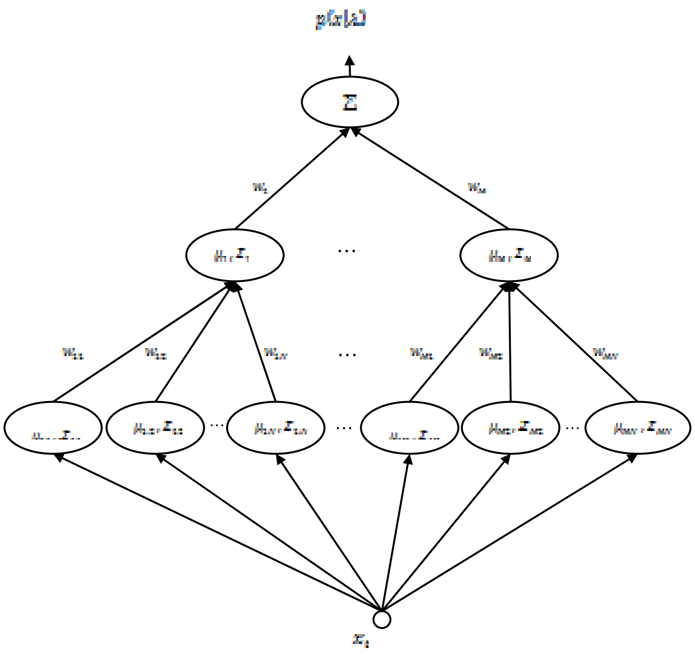

Figure 3: Overview of the structure of HGMM

When training the HGMM, we still use the EM algorithm to perform maximum likelihood training. The specifics of the training are as follows. Given the training vectors $x = \{x_1, x_2, \ldots, x_T\}$, we first determine the probabilistic alignment of the training vectors into each component of HGMM. That is, for the *i*th component of first hierarchy, we compute:

$$p(i|x_t) = \frac{w_i b_i(x_t)}{\sum_{i=1}^{M} w_i b_i(x_t)} \quad (11)$$

And for each Gaussian Mixture of ith component

$$p_i(j|x_t) = \frac{w_{ij} b_{ij}(x_t)}{\sum_{k=1}^{K} w_{ik} b_{ik}(x_t)} \quad (12)$$

K is the number of local mixture. w_{ij} is the local mixture weight which satisfy the constraint $\sum_{j=1}^{K} w_{ij} = 1$.

Where

$$b_{ij}(\bar{x}) = \sum_{j=1}^{K} w_{ij} \frac{1}{(2\pi)^{D/2} |\Sigma_{ij}|^{1/2}} exp\left\{\frac{-(\bar{x} - \overline{\mu_{ij}})' \Sigma_{ij}^{-1} (\bar{x} - \overline{\mu_{ij}})}{2}\right\} \quad (13)$$

Cascade these two formulas, we get

$$p(i,j|x_t) = p(i|x_t) p_i(j|x_t) \quad (14)$$

And then we can compute the statistics for the weight, mean and variance parameters for each component.

$$w_{ij} = \sum_{t=1}^{T} p(i,j|x_t) \quad (15)$$

$$w_i = \sum_{t=1}^{T} p(i|x_t) \quad (16)$$

$$\mu_{ij}(x) = \frac{1}{w_{ij}} \sum_{t=1}^{T} p(i,j|x_t) x_t \quad (17)$$

$$\mu_i(x) = \frac{1}{w_i} \sum_{t=1}^{T} p(i|x_t) x_t \quad (18)$$

$$\delta_{ij}(x^2) = \frac{1}{w_{ij}} \sum_{t=1}^{T} p(i,j|x_t) x_t^2 \quad (19)$$

$$\delta_i(x^2) = \frac{1}{w_i} \sum_{t=1}^{T} p(i|x_t) x_t^2 \quad (20)$$

4. Intrusion Detection Experiments

The first requirement of each IDSs is a set of input data for processing and determining the security level. We trained and tested our system using KDD Cup's 99 dataset [10],[11].

4-1. Database

The 1998 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. The 1999 KDD intrusion detection contest uses a version of this dataset [6].

Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a LAN simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

Each connection is labeled as either normal, or as an attack of a specific type that mention in below.

Table 1 gives the details of KDD CUP 99 data.

Table 1: Attacks in KDD99's database

Classification of Attacks	Attack Name
Denial of Service	Neptune, Smurf, Pod, Teardrop, Land, Back, Apache2, Udpstorm, Process-table, Mail-bomb
Remote to User	Guess-password, Ftp-write, Imap, Phf, Multihop, Warezmaster, Warezclient, Snmpgetattack, Named, Xlock, Xsnoop, Send-mail
User to Super User	Buffer-overflow, Load-module, Perl, Rootkit, Xterm, Ps, Http-tunnel, Sql-attack, Worm, Snmp-guess
Probing	Port-sweep, IP-sweep, Nmap, Satan, Saint, Mscan

In KDD99, Attacks fall into four main categories [11]:

- Probing: is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities; surveillance and other probing, e.g., port scanning.
- DOS: Denial of Service; is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine, e.g. Syn-flood.
- U2R: User to root exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system; unauthorized access to local super user (root) privileges, e.g., various “buffer overflow” attacks.
- R2L: A remote to user (R2L) attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine’s vulnerability to illegally gain local access as a user; unauthorized access from a remote machine, e.g. guessing password.

4-2. Feature Extraction

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. Stolfo et al. used domain knowledge to add features that look for suspicious behavior in the data portions, such as: length (number of seconds) of the connection; type of the protocol, e.g. tcp, udp, etc; network service on the destination, e.g., http, telnet, etc; number of data bytes from source to destination and vice versa; number of connections to the same host as the current connection in the past two seconds; number of operations on access control files; number of file creation operations; number of outbound commands in an ftp session; the number of failed login attempts; these features are called "content" features. A complete listing of the set of features defined for the connection records is given in [10].

For using HGMM preprocessing and normalization of data is required.

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.

Three features spanned over a very large integer range, namely length [0, 60000], 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, 4.78], [0.0, 9.14] and [0.0, 9.14] respectively. All other features were boolean, in the range [0.0, 1.0].

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.

4-3. Decision System

For attack identification, a group of attacks $\{1,2,\dots,S\}$ is represented by GMM’s $\lambda_1,\lambda_2,\dots\lambda_S$. The objective is to find the attack which has the maximum a posterior probability for a given input.

Formally,

$$\hat{S} = \arg\max_{1\le k\le S} p(\lambda_k|x) = \arg\max_{1\le k\le S} \frac{p(x|\lambda_k)p(\lambda_k)}{p(x)} \quad (21)$$

Where the second equation is due to Bays’ rule.

Nothing that $p(x)$ is the same for all attacks, the classification rule simplifies to

$$\hat{S} = \arg\max_{1\le k\le S} p(x|\lambda_k) \quad (22)$$

Using logarithms and the independence between observations, the attack identification system computes

$$\hat{S} = \arg\max_{1\le k\le S} \sum_{t=1}^{T} \log p(x_t|\lambda_k) \quad (23)$$

In which $p(x_t|\lambda_k)$ is given in 1.

4-4. Experimental Result

The raw dataset was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. We choose almost 310,000 records from this database as the test set. Each connection record consists of about 100 bytes.

Our experiment implemented by Visual C++

The number of attacks in train and Test set for each of the elements is according to table 2.

Table 2: KDD Cup 99 Training and Testing Set

Attack Type	DDOS	Probe	U2R	R2L	Total Attack	Normal
Training Set	391458	4107	52	1126	494020	97277
Testing Set	229853	4166	2636	13781	311029	60593

Table 3 shows the confusion matrix obtained for HGMM IDS.

Table 3: Confusion Matrix for Hierarchical Gaussian Mixture Model IDS

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%Correct
Normal	53405	222	65	40	6861	88.14
Probe	15	4138	7	5	1	99.33
DOS	228	276	229349	0	0	99.78
U2R	14	0	0	2531	91	96.01
R2L	2378	2	1	9	11391	82.66
%Correct	95.30	89.22	99.97	97.91	62.09	

Table 4 gives the experimental results of intrusion detection using HGMM. The results clearly reveal that this method is able to achieve much higher detection rates which are obtained with negligible false detection rate in the KDD99 dataset detecting the unknown attacks.

Notably the experimental results shows the calculated DR factors for HGMM method in most attacks are higher than the one in other mentioned methods. Particularly this factor is pretty higher for R2L and U2R in our presented method.

Table 4: Comparison of Detection Rate of various classifiers (GMix, RBF, SOM, Binary Tree, ART, LAMASTAR & HGMM)

		Normal	Probe	DOS	U2R	R2L
Gmix	DR	98.97	93.03	88.24	22.8	9.6
	FAR	35.01	72.27	0.15	45.84	1.02
RBF	DR	99.07	91.31	75.10	7.01	5.6
	FAR	42.78	88.13	0.29	61.91	0.88
SOM	DR	93.98	64.30	96.10	21.49	11.7
	FAR	26.53	78.70	0.22	24.62	0.32
Binary Tree	DR	96.43	77.94	96.45	13.59	0.44
	FAR	21.22	52.37	3.58	35.42	7.70
ART	DR	97.19	98.48	97.09	17.98	11.3
	FAR	21.92	54.79	0.66	10.87	0.17
LAMSTAR	DR	99.69	98.48	99.21	28.94	41.2
	FAR	13.32	24.97	0.47	9.96	0.06
HGMM	DR	88.14	99.33	99.78	96.01	82.66
	FAR	4.70	10.78	0.03	2.09	37.91

5. Conclusion

In this paper we proposed a novel method based on Hierarchical Gaussian Mixture Model for intrusion detection mechanism. HGMM is an effective model for detecting computer attacks of unknown patterns. Experimental results in standard dataset prove that this method is able to achieve accuracy much better than the six classification techniques; Gaussian Mixture, Radial Basis Function, Binary Tree Classifier, SOM, ART and LAMASTAR.

References

[1]. H. Lin, C. Jiang, H. Huang, "A Secure and Efficient Model for Network Defensive Systems", ICS Dec 2006, Taiwan.

[2]. 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.

[3]. S. Mukkamala, G. Janoski, A. Sung, “Intrusion detection using neural networks and support vector machines” Proceedings of the 2002 IEEE International Joint Conference on Neural Networks, PP. 1702 – 1707.

[4]. A. K. Ghosh, “Learning Program Behavior Profiles for Intrusion Detection”. *USENIX* 1999.

[5]. V. Venkatachalam, S. Selvan, “Intrusion Detection using Improved Competitive Learning Lamstar Neural Network”, IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.2, February 2007.

[6]. Lincoln Laboratory, Massachusetts Institute of Technology (MIT), 1998-2000. DARPA Intrusion Detection Evaluation.

[7]. D.A. Reynolds, T.F. Quatieri, and R.B. Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models", Digital Signal Processing, Pages 19-41, 2000.

[8]. G. Singh, A. Panda, S. Bhattacharyya, and T. Srikanthan, “Vector Quantization Techniques for GMM Based Speaker Verification”, ICASSP 2003.

[9]. O.M. San, V.N. Huynh, and Y. Nakamori, "An Alternative Extension Of The K-Means Algorithm for Clustering Categorical Data", Int. J. Appl. Math. Comput. Sci. Vol. 14, No. 2, 241–247, 2004.

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

[11]. 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.