Intrusion Detection Based on Rule Extraction from Dynamic Cell Structure Neural Networks

Mansour Sheikhan [1], Amir Khalili [2]
1- Assistant Professor of Electrical Engineering Department, Islamic Azad University, South Tehran Branch, Iran.
Email: msheikhn@azad.ac.ir
2- M.Sc. of Computer Engineering, Islamic Azad University, South Tehran Branch, Iran.
Email: a_khalili@azad.ac.ir

Received: September 2009 Revised: September 2010 Accepted: October 2010

ABSTRACT:
Knowledge embedded within artificial neural networks (ANNs) is distributed over the connections and weights of neurons. So, the user considers ANN as a black box system. There are many researches investigating the area of rule extraction by ANNs. In this paper, a dynamic cell structure (DCS) neural network and a modified version of LERX algorithm are used for rule extraction. On the other hand, intrusion detection system (IDS) is known as a critical technology to secure computer networks. So, the proposed algorithm is used to develop IDS and classify the patterns of intrusion. To compare the performance of the proposed system with other machine learning algorithms, multi-layer perceptron (MLP) with output weight optimization-hidden weight optimization (OWO-HWO) training algorithm is employed with selected inputs based on the results of a feature relevance analysis. Empirical results show the superior performance of the IDS based on rule extraction from DCS, in recognizing hard-detectable attack categories, e.g. user-to-root (U2R) and also offering competitive false alarm rate (FAR). Although, MLP with 25 selected input features, instead of 41 standard features introduced by knowledge discovery and data mining group (KDD), performs better in terms of detection rate (DR) and cost per example (CPE) when compared with some other machine learning methods, as well.

KEYWORDS: Rule extraction, dynamic cell structure, intrusion detection system.

1. INTRODUCTION

In machine learning and data mining research, rule learning has become an important topic. On the other hand, artificial neural networks (ANNs), in spite of their adaptivity and wide range of applications such as pattern classification and time-series prediction, have an important drawback: knowledge embedded within ANNs is distributed over the activations and connections of neurons and is not transparent to users [1-3].

There are many researches investigating the area of rule extraction by different structures of ANNs [4-15]. The researches on rule extraction can be classified into three approaches: decompositional, pedagogical and eclectic.

Analyzing the activation and weights of the hidden layers of ANN is performed in decompositional approach [16-19]. The pedagogical approach treats the ANN as a black box and extract rules by only looking at the input and output activations [1, 20]. Finally, the eclectic approach, which is based on two former approaches, is characterized by any use of knowledge concerning the internal architecture and/or weight vectors in a trained ANN to complement a symbolic learning algorithm [21]. In this paper, the third approach is investigated in which rules are extracted from a dynamic cell structure (DCS) neural network.

Most of the techniques developed thus far for rule extraction are very NN-specific. Two specific rule extraction techniques seemed closely related to this work. One technique, RULEX, was applied first to a constrained multilayer perceptron (MLP) [1] and then to a local-cluster NN [22]. Another technique, LREX, has been used to extract rules from radial basis function (RBF) neural network [23, 24]. In this paper, a modified version of LREX is used for rule extraction from the DCS.

Also, because of the importance of security in information and communication technology (ICT) and critical role of intrusion detection systems (IDSs) in this area, the extracted rules from DCS are used in this paper for attack recognition in computer networks.

The rest of this paper is organized as follows. In Section 2, the basics of IDS are reviewed. The foundation of DCS is described in Section 3. The rule extraction algorithm is introduced in Section 4. The

training data of the system is detailed in Section 5. The foundations of output weight optimization-hidden weight optimization (OWO-HWO), as the training algorithm of the MLPs in this paper, is described in Section 6. The details of empirical results and conclusion are also drawn in Section 7 and Section 8, respectively.

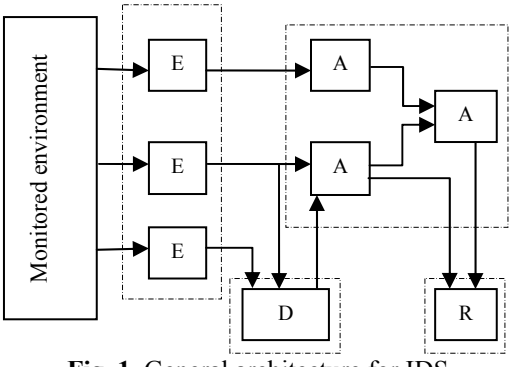

Fig. 1. General architecture for IDS

2. INTRUSION DETECTION SYSTEMS

In 1998, a working group created by Defense Advanced Research Project Agency (DARPA) oriented towards coordinating and defining a common framework in the IDS field. Integrated within Internet Engineering Task Force (IETF) in 2000, and having adopted the new acronym Intrusion Detection Working Group (IDWG), the group defined a general IDS architecture (Fig. 1) [25].

This architecture is based on four types of functional modules: Event (E), Database (D), Analysis (A), and Response (R).

"E" blocks acquire information events to be analyzed by other blocks. Depending on the information source considered, IDS may be either host or network-based [26]. A host-based IDS analyzes events that mainly related to operating system (OS) information. A network-based IDS analyzes network related events: traffic volume, Internet Protocol (IP) addresses, service ports, protocol usage, etc.

"D" blocks intended to store information from "E" blocks for subsequent processing by "A" and "R" boxes.

"A" blocks are processing modules for analyzing events and detecting potential intrusions. Depending on the type of analysis carried out, IDSs are classified as either anomaly-based [27, 28] or misuse-based [29, 30].

"R" blocks generate a response, if an intrusion is detected.

Anomaly-based detectors attempt to estimate the normal or abnormal behaviors of the system to be protected. Misuse-based schemes seek defined patterns within the analyzed data.

The detection techniques in anomaly-based IDS can be classified into three main categories: statistical-based [31], knowledge-based [25] and machine learning (e.g. Bayesian networks [32], Markov models [33], ANNs [27, 34], fuzzy logic [35], and genetic algorithms [36]).

The detection techniques that have been used in misuse-based IDS can also be classified into three main categories: statistical-based [37], knowledge-based [38] and machine learning (e.g. Bayesian networks [39], ANNs [40-43], fuzzy logic [44], genetic algorithms [45], decision trees [46] and hybrid systems [47, 48]).

3. DCS NEURAL NETWORK

The DCS neural network is known as a member of self-organizing maps (SOMs). This neural network, which is implemented in the GEN1 system by National Aeronautics Space Administration (NASA), was originally developed by Bruske and Sommer [49]. This network was a derivative of Fritzke's work [50] combined with competitive Hebbian learning by Martinez [51].

The DCS is designed as a topology representing network that learns the function that describes a map of the input space, represented as Voronoi regions.

The neurons within the NN represent the reference vector (centroid) for each of the Voronoi regions. The connection between the neurons, c_{ij}, is connecting neighboring Voronoi regions through their reference vectors. This reference vector is known as the "best matching unit" (BMU). Given an input, X, the BMU is the neuron whose weights, W, are closest to X. Along with the BMU, the "second BMU" (SBU) is found to adjust nearby neurons within the BMU neighborhood (NBR).

The DCS neural networks consist of two learning rules, Hebbian and Kohonen. Hebbian learning updates c_{ij} between neurons i and j:

$$C_{ij}(t+1)=\begin{cases}1 & ;(\mathrm{i}=\mathrm{BMU})\wedge(\mathrm{j}=\mathrm{SBU})\\ 0 & ;(\mathrm{i}=\mathrm{BMU})\wedge(\mathrm{C_{ij}}<\theta)\ \wedge(\mathrm{j}\in\mathrm{NBR}\setminus\{\mathrm{SBU}\})\\ \alpha C_{ij}(t) & ;(\mathrm{i}=\mathrm{BMU})\wedge(\mathrm{C_{ij}}\geq\theta)\wedge(\mathrm{j}\in\mathrm{NBR}\setminus\mathrm{SBU}\})\\ C_{ij}(t) & ;(i,j\neq BMU)\end{cases} \quad (1)$$

In this equation, the forgetting constant, α, is included to produce a weakening between i and j, if they are not currently the closest to the stimulus, and θ is the edge threshold, a minimum acceptable connection strength in order for the connection to be considered valid. Kohonen learning is used to adjust the weight vectors, W, of the neurons:

$$\Delta\vec{w}_i=\begin{cases}\varepsilon_{\mathrm{BMU}}(\vec{X}-\vec{w}_i(t)) & ;\ (\mathrm{i}=\mathrm{BMU})\\ \varepsilon_{\mathrm{NBR}}(\vec{X}-\vec{w}_i(t)) & ;\ (\mathrm{i}\in\mathrm{NBR})\\ 0 & ;\ (\mathrm{i}\neq\mathrm{BMU})\wedge(\mathrm{i}\notin\mathrm{NBR})\end{cases} \quad (2)$$

where $\varepsilon_{\mathrm{BMU}}$ is the BMU weight adjustment

parameter and ε_{NBR} is the weight adjustment applied to the neighborhood of the BMU.

These two learning rules allow the DCS to change its structure. The ability to add new neurons into the network, as it grows, gives the DCS the potential to evolve into many different configurations.

4. RULE EXTRACTION ALGORITHM

As mentioned in the introduction, a modification of the LERX algorithm by McGarry et al. [23, 24] is used in this work for extracting rules from the DCS. This algorithm was originally used to extract rules from RBF neural network [52].

After training the network, the weights of DCS are used as inputs to the algorithm. The BMU corresponding to each data point is recorded during training and is used as an input to the algorithm, too. The training data is divided into regions based on the BMU. Then for each region, x_{lower} is the smallest value of the independent variable that has a particular BMU and x_{upper} is the largest value of that independent variable that has the same BMU.

These two numbers form bounds for the intervals in the antecedent statement (e.g. ("variable>= x_{lower}" AND "variable<= x_{upper}")). An interval is determined for each of the independent variables and the statements are connected by "AND" to form the full antecedent. The algorithm of rule extraction is as follows:

Input:
Weights of the DCS (centers of Voronoi regions)
Best matching unit for each input
Output:
One rule for each cell of the DCS
Procedure:
Train DCS on the data set
Record BMU for each input
Collect all inputs with common BMU to form cell
For each weight (w_i)
For each independent variable
$x_{lower} = \min\{x \mid x \text{ has BMU} = w_i\}$
$x_{upper} = \max\{x \mid x \text{ has BMU} = w_i\}$
Build rule by:
Independent variable in $[x_{lower}, x_{upper}]$
Join antecedent statements with AND
Dependent variable = category
OR
Dependent variable in $[y_{lower}, y_{upper}]$
Join conclusion statements with AND
Write Rule

5. TRAINING DATA

In 1998, the Lincoln Laboratory at MIT, under the DARPA sponsorship, constructed and distributed the first standard dataset for evaluation of computer network IDS [53]. Afterward knowledge discovery and data mining group (KDD) collected and generated TCP dump data provided by the aforementioned DARPA in the form of train-and-test sets whose 41 features are defined for the connection records [54]. The attacks of KDD fall into one of four categories:

a. Denial of Service (DoS): Attacker tries to prevent legitimate users from using a service.
b. Remote to Local (R2L): Attacker does not have an account on the victim machine, hence tries to gain access.
c. 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.

The KDD dataset consists of three components: "10% KDD", "Corrected KDD", and "Whole KDD" (Table 1).

There are multiple attack types for each main attack category. Table 2 lists the attack categories along with the 22 attack types in the "10% KDD" dataset.

In this work, 49402 records from "10% KDD" are chosen as the training data. This dataset has the same distribution of attacks as "10% KDD" dataset (Table 3). The "Corrected KDD" consists of 14 new unknown attack types. In this work, 31103 records are chosen from "Corrected KDD" with the same distribution of attacks (Table 3).

As mentioned before, each connection in KDD is characterized by 41 features [55] and a label that determines the main category (attacks/normal connection). The description, type, and range of 19 sample features are listed in Table 4.

As shown in Table 4, the features in KDD have different forms (continuous, discrete, and symbolic) with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence, preprocessing is required.

The preprocessing is performed in two steps:

a. Mapping symbolic valued features, such as protocol_type, service, and flag, to integer values.
b. Normalization of feature values.

However, logarithmic scaling (base 10) is applied to three features spanned over a very large integer range, namely duration [0,58329], src_bytes [0,1.3billion] and dst_bytes [0,1.3billion], to reduce the ranges to [0,4.77] and [0,9.11], respectively. Other features are either Boolean, like logged_in, having binary values, or continuous, like diff_srv_rate, in the range of [0,1] and no scaling is needed for these features. So, each of the mapped features are linearly scaled to the range [0,1].

The preprocessed features are the inputs of MLP neural classifier. There are five nodes at the output layer of this classifier. The target values of these nodes

are as follow: 00001 (for Normal pattern), 00010 (for DoS attack), 00100 (for R2L attack), 01000 (for U2R attack), and 10000 (for Probe attack).

6. IMPROVED TRAINING ALGORITHM FOR MLP

To compare the performance of the proposed system with other machine learning algorithms, MLP with OWO-HWO training algorithm is employed with selected input features based on the results of a feature relevance analysis. In this section, the details of OWO- HWO training algorithm and feature ranking method are discussed.

A critical problem in MLP neural networks has been the long training time required. Several fast training techniques, that require the solution of sets of linear equations, have been devised [56, 57].

In output weight optimization-backpropagation (OWO-BP), linear equations are solved to find output weights and backpropagation is used to find hidden weights [58]. Unfortunately, backpropagation is not a very effective method for updating hidden weights [59]. The idea of minimizing a separate error function for each hidden unit is adapted to find the hidden weights and have termed as hidden weight optimization (HWO) [58].

It is noted that in a MLP, if the *j*th unit is a hidden unit, then the net input $net_p(j)$ and the output activation $O_p(j)$ for the *p*th training pattern are:

$$net_p(j) = \sum_i w(j,i).x_p(i) \quad (3)$$

$$O_p(j) = f(net_p(j)) \quad (4)$$

where the *i*th unit is in any previous layer and $w(j,i)$ denotes the weight connecting the *i*th unit to the *j*th unit. For the *k*th output unit, the net input $net_{op}(k)$ for the *p*th training pattern and the output activation $O_{op}(k)$, with the linear property assumption of the output units, are:

$$net_{op}(k) = \sum_i w_o(k,i).O_p(i) \quad (5)$$

$$O_{op}(k) = net_{op}(k) \quad (6)$$

where $w_o(k,i)$ denotes the output weight connecting the *i*th unit to the *k*th output unit.

In order to train a neural network in batch mode, the error for the *k*th output unit is defined as:

$$E(k) = \frac{1}{N_v}\sum_{p=1}^{N_v}[T_p(k) - O_{op}(k)]^2 \quad (7)$$

in which N_v is the number of training patterns.

In this paper, the conjugate gradient approach is used to minimize $E(k)$ [58]. For hidden weight changes, it is desirable to optimize the hidden weights by minimizing separate error functions for each hidden unit. By minimizing many simple error functions, instead of a large one, it is hoped that the training speed and convergence can be improved. The desired hidden net function can be approximated by a current net function plus a net change. That is, for *j*th unit and *p*th pattern, a desired net function can be constructed as [60]:

$$net_{pd}(j) = net_p(j) + Z\delta_p(j) \quad (8)$$

where Z is the learning factor and $\delta_p(j)$ for output units and hidden units are as follows, respectively:

$$\delta_p(j) = f'(net_j).[T_p(j) - O_p(j)] \quad (9)$$

$$\delta_p(j) = f'(net_j).\sum_n \delta_p(n)w(n,j) \quad (10)$$

Similarly, the hidden weights can be updated as:

$$w(j,i) \leftarrow w(j,i) + Z.e(j,i) \quad (11)$$

where $e(j,i)$ is the weight change and serves the same purpose as the negative gradient in backpropagation.

By defining an objective function in terms of mean squared error (MSE) for the *j*th unit as:

$$E_\delta(j) = \sum_{p=1}^{N_v}[\delta_p(j) - \sum_i e(j,i).O_p(i)]^2 \quad (12)$$

and taking the gradient of $E_\delta(j)$ with respect to the weight changes, and setting it to zero, the following linear equations are achieved:

$$\sum_i e(j,i).R_{oo}(i,m) = \frac{-\partial E}{\partial w(j,m)} \quad (13)$$

where

$$R_{oo}(i,m) = \sum_{p=1}^{N_v} O_p(i).O_p(m) \quad (14)$$

The steps of OWO-HWO algorithm are listed as follow:

1. Initialize all weights and thresholds.
2. Increase n by 1 and stop if $n>N_{it}$ (N_{it}=Number of iterations)
3. Apply training pattern and calculate the output activation.
4. Use the conjugate gradient approach to minimize error.
5. If MSE(n) > MSE(n-1)
   $Z \leftarrow Z\downarrow$ % Reduce the value of Z (learning factor)
   Reload the previous best hidden weights
   Go to step 9
6. If $MSE(n) \le MSE(n-1)$
   Accumulate the cross-correlation $R_{\delta o}(m)$ and auto-correlation $R_{oo}(m)$ for hidden units:

$$R_{\delta o}(m) = \sum_{p=1}^{N_v}\delta_p(j).O_p(m)$$

$$R_{oo}(m) = \sum_{p=1}^{N_v} O_p(i).O_p(m)$$

7. Solve linear equations for hidden weight changes:
$$\sum_i e(j,i).R_{oo}(i,m) = R_{\delta o}(m)$$
8. Calculate the learning factor as:
$$Z = \frac{-0.05E}{\sum_j \left[\sum_i \frac{\partial E}{\partial w(j,i)}.e(j,i) \right]}$$
9. Update the hidden weights as:
$w(j,i) \leftarrow w(j,i) + Z.e(j,i)$
10. Go to step 2

Also, feature ranking is an important issue in intrusion detection. Elimination of less significant features lowers the size of ANN and speeds up the computations. The results of using logistic regression to rank the features based on the Chi-square values for different subsets are used in this paper [61]. In this way, the higher the Chi-square value, the higher is the ranking. In Table 5, the ranking results of the Chi-square test on KDD dataset are listed for the 25 most significant features.

7. EMPIRICAL RESULTS

The performance of DCS-based neural IDS is investigated in this section and is compared to MLP with OWO-HWO training algorithm and selected input features.

Based on the results of feature ranking, three experiments are performed by selecting 25, 20 and 15 features as the input vector of MLP, respectively. The MLP in each of these experiments has five linear output neurons (representing 4 attack categories, and 1 normal category). The number of MLP hidden nodes in each of these experiments is selected as 33, 30 and 25, respectively. This selection is based on monitoring the MSE on test data for different values of hidden nodes, e.g. as shown in Table 6 for the case of 20 input features.

The effect of the input feature-vector size reduction on the performance of MLP, when using OWO-HWO as training algorithm, is shown in Fig. 4 for each of the mentioned experiments along with no-feature selection condition. The structure of MLP is shown as [*x y z*] in the legend of figure, representing the number of input, hidden and output nodes, respectively.

Before discussing about the results of experiments, it seems necessary to mention the standard metrics that have been developed for evaluating IDS. Detection rate (DR) and false alarm rate (FAR) are the two most common metrics. DR is computed as the ratio between the number of correctly detected attacks and the total number of attacks, while FAR is computed as the ratio between the number of normal connections that is incorrectly misclassified as attacks and the total number of normal connections.

For the purpose of classifier algorithm evaluation, another comparative measure is cost per example (CPE) [62]. CPE is calculated using the following formula:

$$CPE = \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{m} CM(i,j).C(i,j) \quad (15)$$

Where *CM* and *C* are confusion matrix and cost matrix, respectively. In Eq. (15), *N* represents the total number of test instances, and *m* is the number of classes in classification. *CM* is a square matrix in which each column corresponds to the predicted class, while rows correspond to the actual classes. An entry at row *i* and column *j*, *CM(i,j)*, represents the number of misclassified instances that originally belong to class *i*, although incorrectly identified as a member of class *j*. The entries of the primary diagonal, *CM(i,i)*, stand for the number of properly detected instances. Cost matrix is similarly defined, as well, and entry *C(i,j)* represents the cost penalty for misclassifying an instance belonging to class *i* into class *j*. Cost matrix values employed for the KDD 99 classifier learning contest are shown in Table 7 [54].

The procedure of performance evaluation of the proposed IDS models is depicted in Fig. 3.

The confusion matrices for rule extraction-based approach and also three mentioned MLP classifiers are reported in Table 8 to Table 11, respectively.

The performance of the proposed IDS models is compared with some other machine learning methods, as well (Table 12).

As shown in Table 12, the classification rates of DoS, Probe, and R2L attacks for MLP with 25 selected input-features are better than other reported models. However, DCS offers better classification rate for U2R attack as compared to others. It should be noted that most of the machine learning algorithms have offered an acceptable level of classification rate for DoS and Probe attack categories and demonstrated poor performance on the R2L and U2R categories [65].

DR and CPE of MLP classifier with 25 selected input-features are better than other models, too. However, FAR of the DCS model is competitive as compared to other models.

Table 1. Number of samples in KDD 99 datasets

KDD dataset	Normal	Probe	DoS	U2R	R2L
10%	97277	4107	391458	52	1126
Corrected	60593	4166	229853	70	16347
Whole	972780	41102	3883370	52	1126

Table 2. Attack types and number of their samples in 10% KDD dataset

Category	Type (Number of samples)
Probe	satan (1589), ipsweep (1247), portsweep (1040), nmap (231)
DoS	smurf (280790), neptune (107201), back (2203), teardrop (979), pod (264), land (21)
U2R	buffer_overflow (30), rootkit (10), loadmodule (9), perl (3)
R2L	warezclient (1020), guess_passwd (53), warezmaster (20), imap (12), ftp_write (8), multihop (7), phf (4), spy (2)

Table 3. Size of the training and test datasets

Class	Training samples		Test samples	
	Number	Distribution (%)	Number	Distribution (%)
Normal	9727	19.69	6059	19.48
Probe	411	0.83	417	1.34
DoS	39145	79.24	22985	73.90
U2R	6	0.01	7	0.02
R2L	113	0.23	1635	5.26
Total	49402	100	31103	100

Table 4. Description and value ranges of 19 sample features in KDD dataset

Feature	Description	Type	Value ranges
duration	Duration of the connection (in seconds)	continuous	[0,58329]
protocol_type	Type of the connection protocol	discrete	3 different symbols
service	Service on the destination	discrete	70 different symbols
flag	Status flag of the connection	discrete	11 different symbols
src_bytes	Number of bytes sent from source to destination	continuous	[0,1.3e+9]
dst_bytes	Number of bytes sent from destination to source	continuous	[0,1.3e+9]
wrong_fragment	Number of wrong fragments	continuous	[0,3]
urgent	Number of urgent packets	continuous	[0,14]
hot	Number of "hot" indicators	continuous	0,101]
num_failed_logins	Number of failed logins	continuous	[0,5]
num_compromised	Number of "compromised" conditions	continuous	[0,9]
num_root	Number of "root" accesses	continuous	[0,7468]
num_file_creations	Number of file creation operations	continuous	[0,100]
num_shells	Number of shell prompts	continuous	[0,5]
num_access_files	Number of operations on access control files	continuous	[0,9]
count	Number of connections to the same host as the current connection in the past two seconds	continuous	[0,511]
srv_count	Number of connections to the same service as the current connection in the past two seconds	continuous	[0,511]
dst_host_count	Number of connections having the same destination host	continuous	[0,255]
dst_host_srv_count	Number of connections having the same destination host and using the same service	continuous	[0,255]

Table 5. Chi-square values of the 25 most significant features

Attack type / Feature	Probe	DoS	U2R	R2L
dst_host_diff_srv_rate	3686.3	1334.8	2532.0	1114.1
rerror_rate	2734.5	1016.3	613.4	1016.5
dst_host_srv_rerror_rate	2707.7	967.9	301.1	586.2
srv_rerror_rate	2515.7	805.5	244.9	583.3
dst_ host_rerror_rate	2252.0	732.8	207.8	560.6
diff_srv_rate	1228.3	551.7	39.9	350.1
dst_host_same_srv_rate	793.3	449.2	39.2	311.1
service	588.7	438.8	36.7	249.5
dst_host_srv_count	546.1	433.0	32.6	239.2
logged_in	427.2	363.6	25.1	141.8
dst_host_srv_diff_host_rate	422.3	353.5	25.0	141.3
srv_count	123.4	344.9	15.5	141.2
same_srv_rate	91.8	336.9	15.3	126.1
protocol_type	84.6	328.7	10.7	125.0
num_compromised	70.4	308.4	10.3	116.0
wrong_fragment	68.6	275.6	6.4	99.8
dst_host_same_src_port_rate	65.4	274.0	6.3	78.3
hot	33.9	240.3	6.2	53.1
srv_serror_rate	20.3	188.9	6.2	46.8
dst_host_srv_serror_rate	19.6	129.1	6.2	45.5
is_guest_login	18.2	121.4	3.8	37.1
serror_rate	17.7	102.2	3.4	33.9
src_bytes	8.3	101.5	3.4	27.7
duration	7.6	52.4	2.9	26.1
dst_host_serror_rate	7.4	45.4	2.7	26.0

Table 6. Performance of MLP-based IDS with 20 input features after 50 training epochs

Number of hidden nodes	MSE-training samples	MSE-test samples	Detection rate (%)
10	0.0044	0.0447	97.42
15	0.0038	0.0406	97.75
20	0.0031	0.0407	97.96
25	0.0020	0.0350	99.20
30	0.0020	0.0284	99.58
35	0.0018	0.0291	99.52

Table 7. Cost matrix values for KDD dataset

Predicted / Actual	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

Table 8. Confusion matrix of DCS-based IDS

Predicted / Actual	Normal	Probe	DoS	U2R	R2L
Normal	6031	18	10	0	0
Probe	44	325	48	0	0
DoS	59	33	22893	0	0
U2R	2	2	1	1	1
R2L	657	5	0	1	972

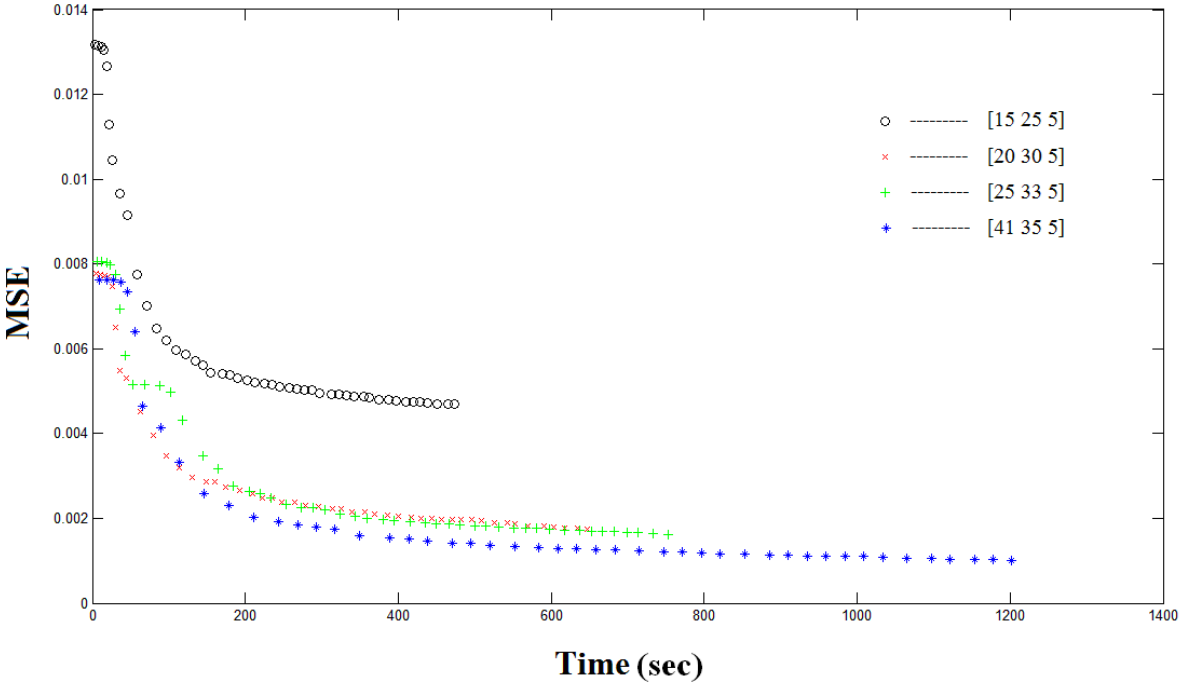

Fig. 2. Performance of MLP-based IDS, using OWO-HWO training algorithm and feature-selection method.

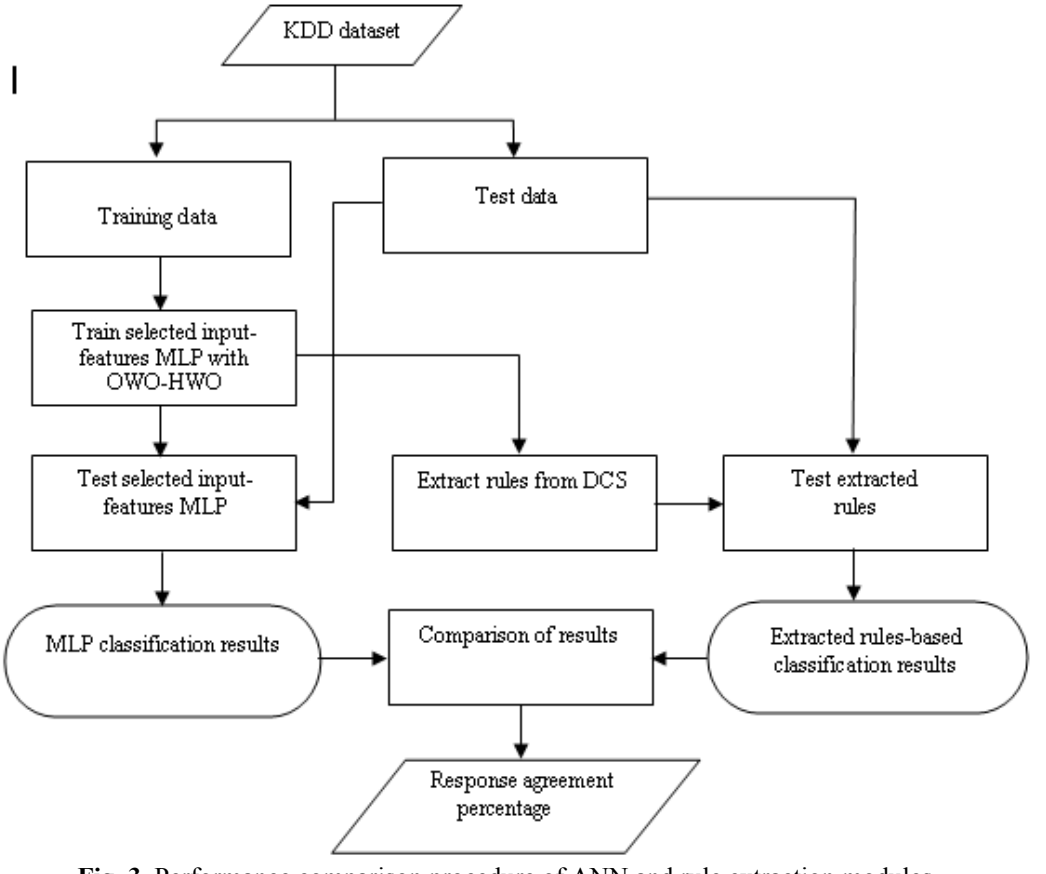

Fig. 3. Performance comparison procedure of ANN and rule extraction modules.

Table 9. Confusion matrix of MLP-based IDS with 25 input features

Actual \ Predicted	Normal	Probe	DoS	U2R	R2L
Normal	6027	8	10	0	14
Probe	7	374	32	0	4
DoS	12	1	22964	0	8
U2R	2	3	1	0	1
R2L	31	2	4	0	1598

Table10. Confusion matrix of MLP-based IDS with 20 input features

Actual \ Predicted	Normal	Probe	DoS	U2R	R2L
Normal	6015	7	16	0	21
Probe	8	371	30	0	8
DoS	5	5	22956	0	19
U2R	2	2	2	0	1
R2L	33	2	15	0	1585

Table 11. Confusion matrix of MLP-based IDS with 15 input features

Actual \ Predicted	Normal	Probe	DoS	U2R	R2L
Normal	6041	10	7	0	1
Probe	9	362	21	0	25
DoS	8	8	22899	0	70
U2R	3	1	2	0	1
R2L	21	0	25	0	1589

Table 12. Performance comparison of different models for intrusion detection

Model	Classification rate					DR	FAR	CPE
	DoS	Probe	R2L	U2R	Normal			
Winner of KDD in 2000 [46]	97.1	83.3	8.4	13.2	99.5	91.8	0.6	0.2331
Runner up of KDD in 2000 [63]	97.5	84.5	7.3	11.8	99.4	91.5	0.6	0.2356
PNrule [62]	96.9	73.2	10.7	6.6	99.5	91.1	0.4	0.2371
ESC-IDS [64]	99.5	84.1	31.5	14.1	98.2	95.3	1.9	0.1579
DCS (proposed)	99.6	77.9	59.4	14.3	99.5	92.7	0.46	0.0959
MLP-25 features (OWO-HWO)	99.9	89.7	97.7	0.0	99.5	99.6	0.53	0.0105
MLP-20 features (OWO-HWO)	99.9	89.0	96.9	0.0	99.3	99.5	0.73	0.0129
MLP-15 features (OWO-HWO)	99.6	86.8	97.2	0.0	99.7	99.2	0.30	0.0142

8. CONCLUSION

In this paper, the performance of rule extracting module from a dynamic cell structure neural network has been investigated in intrusion detection application and compared with fast MLP-based IDS which has used OWO-HWO training algorithm and selected input features.

In this way, a modified version of LERX algorithm has been used for rule extraction from the DCS. The proposed DCS model performs successfully in recognizing hard detectable attacks, such as U2R. The FAR of DCS model is competitive, as compared to other models, too.

The MLP with OWO-HWO training algorithm and 25 selected input features has offered better performance in classifying other attack categories. The DR of this model is higher and its CPE is lower than other models, as well.

REFERENCES

[1] R. Andrews, J. Diederich, and A.B. Tickle, **"A survey and critique of techniques for extracting rules from trained artificial neural networks"**, *Knowledge-Based Systems*, 8, pp. 373-389, (1995)

[2] F. Behloul, B.P.F. Lelieveldt, A. Boudraa, and J.H.C. Reiber, **"Optimal design of radial basis function neural networks for fuzzy-rule extraction in high dimensional data"**, *Pattern Recognition*, 35, pp. 659-675, (2002).

[3] C.J. Mantas, J.M. Puche, and J.M. Mantas, **"Extraction of similarity based fuzzy rules from artificial neural networks"**, *International Journal of Approximate Reasoning*, 43, pp. 202-221, (2006)

[4] G. Towell, and J. Shavlik, **"The extraction of refined rules from knowledge based neural networks"**, *Machine Learning*, 13, pp. 71-101, (1993)

[5] C.W. Omlin, and C.L. Giles, **"Extraction of rules from discrete-time recurrent neural networks"**, *Neural Networks*, 9, pp. 41-52, (1996)

[6] S.H. Huang, and H. Xing, **"Extract intelligible and concise fuzzy rules from neural networks"**, *Fuzzy Sets and Systems*, 132, pp. 233-243, (2002)

[7] G. Bologna, **"Is it worth generating rules from neural network ensembles?"**, *Journal of Applied Logic*, 2, pp. 325-348, (2004)

[8] G. Leng, T.M. McGinnity, and G. Prasad, **"An approach for on-line extraction of fuzzy rules using a self-organizing fuzzy neural network"**, *Fuzzy Sets and Systems*, 150, pp. 211-243, (2005)

[9] E.R. Hruschka, and N.F.F. Ebecken, **"Extracting rules from multilayer perceptrons in classification problems: a clustering-based approach"**, *Neurocomputing*, 70, pp. 384-397, (2006)

[10] K. Odajima, Y. Hayashi, G. Tianxia, and R. Setiono, **"Greedy rule generation from discrete data and its use in neural network rule extraction"**, *Neural Networks*, 21, pp. 1020-1028, (2008)

[11] L.E. Zárate, S.M. Dias, and M.A.J. Song, **"FCANN: A new approach for extraction and representation of knowledge from ANN trained via formal concept analysis"**, *Neurocomputing*, 71, pp. 2670-2684, (2008)

[12] R. Setiono, B. Baesens, and C. Mues, **"A note on knowledge discovery using neural networks and its application to credit card screening"**, *European Journal of Operational Research*, 192, pp. 326-332, (2009)

[13] E. Kolman, and M. Margaliot, **"Extracting symbolic knowledge from recurrent neural networks-a fuzzy logic approach"**, *Fuzzy Sets and Systems*, 160, pp. 145-161, (2009)

[14] H. Kahramanli, and N. Allahverdi, **"Rule extraction from trained adaptive neural networks using artificial immune systems"**, *Expert Systems with Applications*, 36, pp. 1513-1522, (2009)

[15] S. Yu, X. Guo, K. Zhu, and J. Du, **"A neuro-fuzzy-GA-BP method of seismic reservoir fuzzy rules extraction"**, *Expert Systems with Applications*, 37, pp. 2037-2042, (2010)

[16] Y. Hayashi, **"A neural expert system with automated extraction of fuzzy if-then rules"**, *Advances in Neural Information Processing Systems*, vol. 3, pp. 1263-1268, (1991)

[17] C.L. Giles, and C.W. Omlin, **"Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent networks"**, *Connection Science*, 5, pp. 307-328, (1993)

[18] L.M. Fu, **"Rule generation from neural networks"**, *IEEE Transactions on System, Man and Cybernetics*, 28, pp. 1114-1124, (1994)

[19] D.W. Optiz, and J.W. Shavlik, **"Dynamically adding symbolically meaningful nodes to knowledge-based neural networks"**, *Knowledge-Based Systems*, 8, pp. 301-311, (1995)

[20] K. Saito, and P. Nakano, **"Medical diagnosis expert system based on PDP model"**, *Proc. Int. Conf. Neural Networks*, Vol. 1, pp. 255-262, (1988)

[21] E. Keedwell, A. Narayanan, and D. Savic, **"Creating rules from trained neural networks using genetic algorithms"**, *International Journal of Computers*, Systeming Signals, 1, pp. 30-42, (2000)

[22] R. Andrews, and S. Geva, **Rule extraction from local cluster neural nets**, *Neurocomputing*, 47, pp. 1-20, (2002)

[23] K. McGarry, J. Tait, S. Wermter, and J. McIntyre, **"Rule-extraction from radial basis function networks"**, *Proc. Int. Conf. Artificial Neural Networks*, Vol. 1, pp. 613-618, (1999)

[24] K. McGarry, S. Wermter, and J. McIntyre, **The extraction and comparison of knowledge from local function networks**, *International Journal of Computational Intelligence and Applications*, 1, pp. 369-382, (2001)

[25] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, **"Anomaly-base network intrusion detection: techniques, systems and challenges"**, *Computers & Security*, 28, pp. 18-28, (2009)

[26] P. Kabiri, and A.A. Ghorbani, **"Research in intrusion detection and response-a survey"**, *International Journal of Network Security*, 1, pp. 84-102, (2005)

[27] T. Shon, and J. Moon, **"A hybrid machine learning approach to network anomaly detection"**, *Information Sciences*, 177, pp. 3799-3821, (2007)

[28] Z. Chen, H. Wang, B. Yang, L. Wang, and R. Sun, **"A FDRS-based data classification method used for abnormal network intrusion detection"**, *Proc. IEEE 3rd Int. Conf. Natural Computation*, Vol. 2, pp. 375-380, (2007)

[29] Y. Chen, A. Abraham, and B. Yang, **"Hybrid flexible neural-tree-based intrusion detection systems"**, *International Journal of Intelligent Systems*, 22, pp. 337-352, (2007)

[30] R. Chang, L. Lai, W. Su, J. Wang, and J. Kouh, **"Intrusion detection by backpropagation neural networks with sample-query and attribute-query"**, *International Journal of Computational Intelligence Research*, 3, pp. 6-10, (2007)

[31] N. Ye, S.M. Emran, Q. Chen, and S. Vilbert, **"Multivariate statistical analysis of audit trials for host-based intrusion detection"**, *IEEE Transactions on Computers*, 51, pp. 810-820, (2002)

[32] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, **"Bayesian event classification for intrusion detection"**, *Proceedings 19th Ann. Computer Security Applications Conf.*, pp. 14-23, (2003)

[33] M.V. Mahoney, and P.K. Chan, **"Learning nonstationary models of normal network traffic for detecting novel attacks"**, *Proc. 8th ACM SIGKDD*, pp. 376-385, (2002)

[34] R. Beghdad, **"Training all the KDD data set to classify and detect attacks"**, *Neural Network World*, 17, pp. 81-91, (2007)

[35] J. Gomez, and D. Dasgupta, **"Evolving fuzzy classifiers**

for intrusion detection", *Proc. IEEE Workshop Information Assurance*, pp. 68-75, (2002)

[36] D. Song, M.I. Heywood, and A.N. Zincir-Heywood, **"Training genetic programming on half a million patterns: an example from anomaly detection"**, *IEEE Transactions on Evolutionary Computation*, 9, pp. 225-239, (2005).

[37] Y. Liao, and V.R. Vemuri, **"Use of K-nearest neighbor classifier for intrusion detection"**, *Computers & Security*, 21, pp. 439-448, (2002)

[38] D. Novikov, R.V. Yampolskiy, and L. Reznik, **"Artificial intelligence approaches for intrusion detection"**, *Proc. IEEE Conf. Systems, Applications and Technology*, pp. 1-8, (2006)

[39] M.V. Joshi, R.C. Agrawal, and V. Kumar, **"Mining needless in a haystack: classifying rare classes via two-phase rule induction"**, *Proc. ACM SIGMOD Conf. Management of Data*, pp. 91-102, (2001)

[40] V. Golovko, and L. Vaitsekhovich, **"Neural network techniques for intrusion detection"**, *Proc. Int. Conf. Neural Networks and Artificial Intelligence*, pp. 65-69, (2006)

[41] A. Herrero, E. Corchado, P. Gastaldo, F. Picasso, and R. Zunino, **"Auto-association neural techniques for intrusion detection systems"**, *Proc. IEEE Int. Symp. Industrial Electronics*, pp. 1905-1910, (2007)

[42] R. Beghdad, **"Critical study of neural networks in detecting intrusions"**, *Computers & Security*, 27, pp. 168-175, (2008)

[43] M. Sheikhan, Z. Jadidi, and M. Beheshti, **"Effects of feature reduction on the performance of attack recognition by static and dynamic neural networks"**, *World Applied Sciences Journal*, 8, pp. 302-308, (2010)

[44] J.E. Dickerson, J. Juslin, J. Koukousoula, and J.A. Dickerson, **"Fuzzy intrusion detection"**, *Proc. IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) Int. Conf.*, Vol. 3, pp. 1506-1510, (2001)

[45] Y. Lin, K. Chen, and X. Liao, **"A genetic clustering method for intrusion detection"**, *Pattern Recognition*, 37, pp. 924-927, (2004)

[46] B. Pfahringer, **"Winning the KDD 99 classification cup: bagged boosting"**, *SIGKDD Explorations*, 1, pp. 65-66, (2000)

[47] K. Shah, N. Dave, S. Chavon, S. Mukherjee, A. Abraham, and S. Sanyal, **"Adaptive neuro-fuzzy intrusion detection system"**, *Proc. IEEE Int. Conf. Information Technology: Coding and Computing*, Vol. 1, pp. 70-74, (2004)

[48] M.S. Abadeh, J. Habibi, and C. Lucas, **"Intrusion detection using a fuzzy genetic–based learning algorithm"**, *Journal of Network and Computer Applications*, 30, pp. 414-428, (2005)

[49] J. Bruske, and G. Sommer, **"Dynamic cell structures"**, *Proc. Neural Information Processing Systems*, pp. 497-504, (1994)

[50] B. Fritzke, **"Growing cell-structures-a self-organizing network for unsupervised and supervised learning"**, *Neural Networks*, 7, pp. 1441-1460, (1994)

[51] T.M. Martinez, **"Competitive Hebbian learning rule forms perfectly topology preserving maps"**, *Proc. Int. Conf. Artificial Neural Networks*, pp. 427-434, (1993)

[52] M. Darrah, B. Taylor, and M. Webb, **"A geometric rule extraction approach used for verification and validation of a safety critical application"**, *Proc. 18th Annual Florida Artificial Intelligence Research Society Conf.*, Vol. 3, pp. 624-627, (2005)

[53] MIT Lincoln Lab., Information Systems Technology Group, **"The 1998 intrusion detection off-line evaluation plan"** (http://www.11.mit.edu/IST/ideval/docs/1998/id98-eval-11.txt, Mar. 1998).

[54] **"1999 KDD cup competition"** (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2007).

[55] W. Lee, S.J. Stolfo, and K.W. Mok, **"Mining in a data-flow environment: experience in network intrusion detection"**, *Proc. 5th ACM SIGKDD*, pp. 114-124, (1999)

[56] M.A. Sartori, and P.J. Antsaklis, **"A Simple method to derive bounds on the size and to train multilayer neural networks"**, *IEEE Transactions on Neural Networks*, 2, pp. 467-471, (1991)

[57] K. Rohani, M.S. Chen, and M.T. Manry, **"Neural subnet design by direct polynomial mapping"**, *IEEE Transactions on Neural Networks*, 3, pp. 1024-1026, (1992)

[58] H.H. Chen, M.T. Manry, and H. Chandrasekaran, **"A neural network training algorithm utilizing multiple sets of linear equations"**, *Conference Record of the 30th Asilomar Conference on Signals, Systems and Computers*, pp. 1166-1170, (1996)

[59] P. Werbos, **"Backpropagation: past and future"**, *Proc. Int. Conf. Neural Networks*, pp. 343-353, (1988)

[60] R.S. Scalero, and N. Tepedelenlioglu, **"A fast new algorithm for training feedforward neural networks"**, *IEEE Transactions on Signal Processing*, 40, pp. 202–210, (1992)

[61] A. Tamilarasan, S. Mukkamala, A.H. Sung, and K. Yendrapalli, **"Feature ranking and selection for intrusion detection using artificial neural networks and statistical methods"**, *Proc. Int. Joint Conf. Neural Networks*, pp. 4754-4761, (2006)

[62] R. Agrawal, and M.V. Joshi, **"PNrule: a new framework for learning classifier models in data mining (a case-study in network intrusion detection)"**, IBM Research Division Report No. RC-21719, (2000)

[63] I. Levin, **"KDD classifier learning contest: LLSoft's results overview"**, *SIGKDD Explorations*, 1, pp. 67-75, (2000)

[64] A. Nadjaran Toosi, and M. Kahani, **"A novel soft computing model using adaptive neuro-fuzzy inference system for intrusion detection"**, *Proc. IEEE Int. Conf. Networking, Sensing and Control*, pp. 834-839, (2007)

[65] M. Sabhnani, and G. Serpen, **"Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set"**, *Journal of Intelligent Data Analysis*, 6, pp. 1-13, (2004)