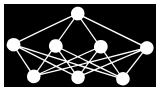

INTELLIGENT NETWORK-MISUSE-DETECTION-SYSTEM USING NEUROTREE CLASSIFIER

B. Muthukumar[*], *S.S. Sivatha Sindhu*[†], *S. Geetha*[‡], *A. Kannan*[§]

Abstract: Intrusion detection systems (IDSs) are designed to distinguish normal and intrusive activities. A critical part of the IDS design depends on the selection of informative features and the appropriate machine learning technique. In this paper, we investigated the problem of IDS from these two perspectives and constructed a misuse based neurotree classifier capable of detecting anomalies in networks. The major implications of this paper are a) Employing weighted sum genetic feature extraction process which provides better discrimination ability for detecting anomalies in network traffic; b) Realizing the system as a rule-based model using an ensemble efficient machine learning technique, neurotree which possesses better comprehensibility and generalization ability; c) Utilizing an activation function which is targeted at minimizing the error rates in the learning algorithm. An extensive experimental evaluation on a database containing normal and anomaly traffic patterns shows that the proposed scheme with the selected features and the chosen classifier is a state-of-the-art IDS that outperforms previous IDS methods.

Key words: *intrusion detection system, misuse detection, genetic algorithm, neural network, decision tree, neurotree*

Received: November 27, 2014 **DOI:** 10.14311/NNW.2015.25.027
Revised and accepted: June 6, 2015

1. Introduction

1.1 Motivation and misuse detection scenario

One of the biggest hurdles to having a secure and safe network is the large amount of human expertise and domain knowledge required to manage and at the same time non-availability of suitable personnel to man them. A mechanism that partially automates the network security management procedures, thus reducing the

[*]Balasubramaniam Muthukumar, Department of Information Technology, Syed Ammal Engineering College, Ramanathapuram, India

[†]Siva S. Sivatha Sindhu, Security Associate, Shan Systems, New Jersey, USA

[‡]Subbiah Geetha – Corresponding author, School of Computing Science and Engineering, VIT Campus Chennai, India, E-mail: geethabaalan@gmail.com

[§]Arputharaj Kannan, School of Information Science and Technology, Anna University, Chennai, India

total dependence on human experts alone is required. Moreover, the mechanism should be intelligent enough to cope with unseen events and should contain a learning strategy to keep itself updated so that human intervention can be minimized. With this as the objective, research on building intelligent IDS was carried out by many researchers [12, 14, 16, 25, 26]. Intelligent IDS is a dynamic defensive system that is capable of adapting to dynamically changing traffic patterns and is present throughout the network rather than only at its boundaries, thus helping to catch all types of attacks. Humans have intuitive characteristics of sensing the incorrect patterns that differ from normal and this is what differentiates a man from a machine. This is the most important property of humans which is not available easily in the computer based systems and hence such intelligence must be incorporated into IDS. Soft-computing techniques impart artificial intelligence to IDS to make them self-functioning as much as possible. These techniques utilize tolerance for tackling ambiguity, uncertainty, partial truth and approximation to achieve robustness at low solution cost. An IDS using soft-computing technique has been proposed by Cannady [6] in which he used artificial neural network for classification. Moreover, there are many other works that use soft-computing techniques for intrusion detection including Genetic Algorithm (GA) [13], Support Vector Machine (SVM), Naive Bayes (NB) [4], Decision Tree (DT) [22, 28, 33] for the discovery of useful knowledge in order to detect intrusive activities. The proposed work develops an advanced intelligent system using ensemble soft-computing techniques for intrusion detection. Integration of soft-computing techniques like Neural Network (NN) [14], GA and DT has lead to the discovery of useful knowledge to detect and prevent intrusion on the basis of observed activity happening in a network. The hybridization [2, 5, 16, 19, 23, 24, 32] of these learning and adaptation techniques overcomes the limitations of individual soft-computing techniques and achieves synergetic effects for intrusion detection.

1.2 Solution strategy and contribution

A misuse detection system tries to discover intrusions by searching for unique patterns or signatures of known attacks. This detection system forms rules based on the known attacks which are stored in knowledge bases for making further inferences. They employ soft-computing techniques like DT algorithm to form the rules which are to be used for deductive inference. As most of the existing misuse detection systems [2, 17, 19, 22] are generally incapable of adapting to the change of circumstances, they report high false alarm rate. However, algorithms based on NN [3, 9, 12] have the capability of capturing the dynamics of network traffic behavior. Unlike traditional statistical models, NNs are data driven, nonparametric weak models (as they take long time to compute a stable output) and they let the data speak for themselves. Also, NNs are universal function approximators that can handle any nonlinear system even without prior assumption about the data. So NNs are less susceptible to the model misspecification problem than most of the parametric models and are more powerful in describing the dynamics of network behaviour than traditional statistical models. In addition, NN methods scale up much better than linear statistical models as the size and complexity of the learning task grows. These characteristics of NN based algorithms make

them more suitable and effective for misuse detection. In IDS, the network profiles may change over time. To handle such dynamic profiles learning algorithms are required to closely track the network behaviour and adapt to the dynamically changing scenario. Therefore, this work proposes an IDS by integrating NN and DT for multi-class categorization i.e., neurotree. The proposed neurotree algorithm generates rules automatically without manual intervention and the rules thus generated possess both generalization and comprehensibility. The generated rules are used to detect a network traffic pattern as either normal or intrusive.

1.3 Structure of the paper

The rest of this paper is organized as follows: Section 2 surveys the related works on intrusion detection, evaluates other related works and presents the need for the proposed model. Section 3 provides the functional framework of neurotree. In Section 4, we detail the design of the neurotree detection methodology which is used for performing the misuse detection. Section 4 also explains the feature extraction algorithm employed, the proposed neurotree algorithm and the performance metrics. Section 5 focuses on the experimental analysis of neurotree detection paradigm with the Neural Network Ensemble (NNE) and extended C4.5 detection model. Section 6 explains the different phases of experiments that have been conducted and also presents the experimental results and analysis of the obtained using preprocessed Knowledge Discovery and Data mining (KDD) dataset. Finally, Section 7 draws conclusion of the work.

2. Literature review

Shun [27] presented a NN-based IDS for detecting internet-based attacks on a computer network. They used NN to identify and predict current and future attacks. Feed-forward NN with the back propagation training algorithm was employed to detect intrusion. Sarasamma et al. [25] proposed a novel multilevel hierarchical Kohonen net to detect intrusions in network. Randomly selected data points from KDD Cup '99 dataset were used by them to train and test the classifier. The process of learning the behavior of a given program by using evolutionary NN based on system-call audit data was proposed by Han et al. [12]. The benefit of using evolutionary NN by them is that it takes lesser amount of time to obtain better NNs than when using conventional approaches. This is because they evolve structures and weights of the NNs simultaneously. They performed the experiment with the KDD Cup '99 intrusion detection evaluation data. Thomas et al. [30] addressed the problem of optimizing the performance of IDS using fusion of multiple sensors. In their approach, trade-off between detection rate and false alarms highlighted that the performance of the detector is better when the fusion threshold is determined according to the Chebyshev inequality. A NN supervised learner was designed by them to determine the weights of individual IDS depending on their reliability in detecting a certain attack. The final stage of their data dependent fusion architecture is a sensor fusion unit which does the weighted aggregation in order to make an appropriate decision. The major limitation with their approach is that it requires large computing power. Linda et al. [17] presented an IDS-NNM

- intrusion detection system using NN based Modeling for detection of anomalous activities. The major contributions of their approach are (i) use and analyses of real network data obtained from an existing critical infrastructure; (ii) the development of a specific window based feature mining technique; (iii) construction of training dataset using randomly generated intrusion vectors and (iv) the use of a combination of two NN learning algorithms namely the Error-Back Propagation and Levenberg-Marquardt, for normal behavior modeling. Stefanos et al. [16] presented a NN classifier ensemble system using a combination of NN which is capable of detecting network attacks on web servers. Their system identifies unseen attacks and categorizes them. The performance of their proposed NN in detecting attacks from audit dataset is fair with success rates of more than 78% in detecting novel attacks and suffers from high false alarms rates.

Comprehensibility, i.e., the ability to explain the learned knowledge is essential in terms of usage in reliable applications like IDS. The existing NN based IDS discussed in the literature lack comprehensibility and this is incorporated by means of extended C4.5 decision tree in the proposed system. Also, a variation in activation function is proposed in order to reduce the error rate thus increasing the detection performance.

An intrusion detection based on the AdaBoost algorithm was proposed by Weiming et al. [32]. In their algorithm, decision stumps were used as weak classifiers and decision rules were provided for both categorical and continuous features. They combined the weak classifiers for continuous attributes and categorical attributes into a strong classifier. The main advantage of their approach is that relations between these two different types of features were handled naturally, without any type conversions between continuous and categorical attributes. Yasami et al. [34] presented a host based IDS using combinatorial of k-means clustering and ID3 decision tree learning algorithms [28] for unsupervised classification of abnormal and normal activities in computer network. The k-means clustering algorithm is first applied by them to the normal training data and it is partitioned into k clusters using Euclidean distance measure. DT was constructed on each cluster using Iterative Dichotomiser 3 (ID3) algorithm. Anomaly score's value from the k-means clustering algorithm and decision rules from ID3 were extracted. Resultant anomaly score value was obtained by them using a special algorithm which combines the output of these two algorithms.

Unlike existing DT based IDS discussed above, the generated rules fired in this proposed work are more efficient in classification of known and unknown patterns, because the proposed system uses a neurotree that incorporates NN to preprocess the data in order to increase the generalization ability. Since the existing DT based approaches discussed in the literature lack generalization ability and they are less effective and inaccurate to classify unseen pattern.

3. Functional framework of neurotree detection paradigm

The system framework of neurotree detection paradigm proposed in this work is shown in Fig. 1. It consists of two phases namely pre-processing phase and detection

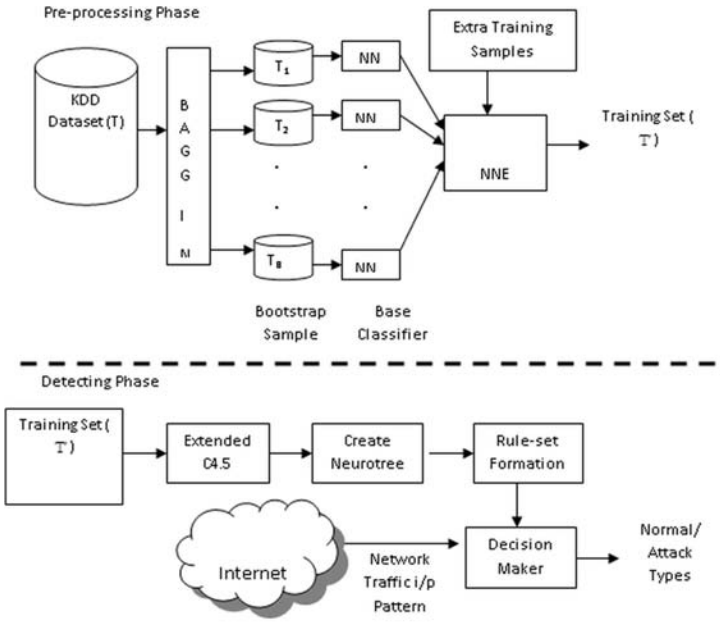

Fig. 1 *Functional framework of neurotree detection paradigm.*

phase. The pre-processing phase generates a training set and creates a trained NNE. The detecting phase constructs a neurotree and classifies a network traffic pattern as normal or attack types.

4. Design of the proposed system

The proposed system has three phases:

- Phase I — Preprocessing of dataset,
- Phase II — Feature extraction,
- Phase III — Classifier construction.

4.1 Rationale for the choice of preprocessing of dataset

The need for data preprocessing can be seen from the fact that redundant data may often confuse the classification algorithm, leading to the discovery of inaccurate or ineffective knowledge. The preprocessing module of the proposed system performs the following functionalities:

i. Performs redundancy check and handles null/missing values,

ii. Converts categorical data to numerical data.

4.2 Rationale for the choice of weighted sum GA as feature extraction

4.2.1 Need for genetic feature extraction

The proposed work for misuse detection approach is based on multi-feature vector decisional framework. The time complexity of the proposed intrusion detection is linear with the size of data points and the number of features. Also, it is very crucial if IDS is operated in real time. Too many features increase the training time and in most cases all these features are not required. Thus, the extraction of useful features for IDS forms the critical part of classification process [6, 8, 20]. The KDD dataset includes 41 network traffic features (Tab. I) and it is not known precisely which of these features are significant in classification of network traffic patterns. If there exists a paradigm or a mathematical formulation which extracts independent features or the complex relationship among features, then IDS task becomes easy. Since there is no such straightforward technique available, a wrapper approach based on genetic feature extraction algorithm is proposed that selects features based on the performance of IDS. This feature extraction approach is based on cascading the weighted sum GA and neurotree. The predictive accuracy of the neurotree is used as a metric in extracting significant features i.e., the importance of the features selected by GA is evaluated based on neurotree.

4.2.2 Algorithm for feature extraction

Algorithm: Feature set extraction using weighted sum GA.
Input: Network traffic pattern (all features), Number of generations (2000), Population size (100), Crossover probability ($P_c = 0.7$), Mutation probability ($P_m = 0.001$).
Output: Set of selected features.
Genetic_Feature_Extraction()

1. Initialize the population randomly with chromosomes of length 41. Each gene value in the chromosome can be '0' or '1'. A bit value of '0' represents that the corresponding feature is not included in chromosome and '1' represents that the feature is included.
2. Initialize the weights $W_1 = 0.7$, $W_2 = 0.3$, N (total number of records in the training set), P_c and P_m.
3. For each chromosome in the new population:
   a. Apply uniform crossover with a probability P_c,
   b. Apply mutation operator to the chromosome with a probability P_m,
   c. Evaluate fitness = $W_1 \times$ Accuracy + $W_2 \times (1/\text{Count\_of\_Ones})$.
4. If (Current_fitness–Previous_fitness < 0.0001) then exit.
5. Select the top best 60% of chromosomes into new population using tournament selection.
6. If number of generations is not reached, go to line 3.

4.2.3 Chromosome representation

Each network traffic pattern is represented as a vector of 41 features, which are the signatures of the respective network behavior. Every chromosome in the population has 41 genes. Each feature is linked with one bit in the chromosome. If the i-th bit is 1, then the i-th feature is selected and used in classification of pattern for intrusion detection, otherwise, that feature is not selected. Each chromosome thus represents a different subset of features. A sample chromosome is shown below:

10011110000000000001111110101011111000111

4.2.4 Initial population

The initial population is generated randomly. The number of 1's for each individual is generated randomly, to form different subset of features. Then, the 1's are randomly placed in the chromosome.

4.2.5 Weighted sum fitness evaluation

The aim of weighted sum fitness evaluation is to use fewer features to attain similar or better performance. Fitness of a chromosome is evaluated based upon the accuracy from the validation dataset and number of features present in a chromosome. Accuracy is calculated using the formula (TP + TN)/(P + N), where TP (True Positive) and TN (True Negative) are the number of records correctly classified in normal and abnormal classes respectively. Positive – P and Negative – N are the total number of records in normal and abnormal classes respectively. Each feature subset contains a list of features. If two subsets attain the same performance, while having different number of features, the subset with fewer features have been chosen. Among accuracy and number of features, accuracy is the key concern, so more weightage is given to accuracy ($W_1 = 0.7$) than the number of features ($W_2 = 0.3$) to be selected. The fitness function is obtained by combining the above terms:

$$\text{fitness} = 0.7 \times \text{Accuracy} + 0.3 \times (1/\text{Count\_of\_Ones}), \tag{1}$$

where "Accuracy" is the classification rate that an individual achieves on validation dataset and "Count_of_Ones" is the number of ones in the chromosome. The number of ones ranges from 1 to 41 where 41 is the length of the chromosome. Here, the second term assumes values in the interval 0 (If none of the feature is selected) to 12.3 (If all the features are selected i.e., $0.3 \times 41 = 12.3$). Among the 41 bits in the chromosome, it is ensured that at least one of the bits in the chromosome is 1 – i.e., there is no chromosome with all bits set to zeroes i.e., indirectly making sure that at least one feature is required for classification of normal and anomalous pattern. In general, higher accuracy implies higher fitness. Also, fitness increases if less number of features are used i.e., if less number of 1's are present in a chromosome. A point to be noted is that chromosome with higher accuracy would outweigh chromosome with lower accuracy, independent of number of features present.

S.No.	FeatureName	Description	Type
1	Duration	length (number of seconds) of the connection	Continuous
2	Protocol_type	type of the protocol, e.g. tcp, udp, etc.	Discrete
3	Dst_bytes	number of data bytes from destination to source	Continuous
4	Flag	normal or error status of the connection	Discrete
5	Land	1 if connection is from/to the same host/port; 0 otherwise	Discrete
6	Wrong_fragment	number of "wrong" fragments	Continuous
7	Urgent	number of urgent packets	Continuous
8	Hot	number of "hot" indicators	Continuous
9	Num_failed_logins	number of failed login attempts	Continuous
10	Logged_in	1 if successfully logged in; 0 otherwise	Discrete
11	Num_compromised	number of "compromised" conditions	Continuous
12	Root_shell	1 if root shell is obtained; 0 otherwise	Discrete
13	Su_attempted	1 if "su root" command attempted; 0 otherwise	Discrete
14	Num_root	number of "root" accesses	Continuous
15	Num_file_creations	number of file creation operations	Continuous
16	Num_shells	number of shell prompts	Continuous
17	Num_access_files	number of operations on access control files	Continuous
18	Num_outbound_cmds	number of outbound commands in an ftp session	Continuous
19	Is_host_login	1 if the login belongs to the "host" list; 0 otherwise	Discrete
20	Is_guest_login	1 if the login is a "guest"login; 0 otherwise	Discrete
21	Count	number of connections to the same host as the current connection in the past two seconds	Continuous
22	Serror_rate	% of connections that have "SYN" errors	Continuous
23	Rerror_rate	% of connections that have "REJ" errors	Continuous
24	Same_srv_rate	% of connections to the same service	Continuous
25	Diff_srv_rate	% of connections to different services	Continuous
26	Srv_count	number of connections to the same service as the current connection in the past two seconds	Continuous
27	Diff_srv_rate	% of connections to different services	Continuous
28	Srv_count	number of connections to the same service as the current connection in the past two seconds	Continuous
29	Srv_serror_rate	% of connections that have "SYN" errors	Continuous
30	Srv_rerror_rate	% of connections that have "REJ" errors	Continuous
31	Srv_diff_host_rate	% of connections to different hosts	Continuous
32	Dst_host_count	count of connections having the same destination host	Continuous
33	Dst_host_srv_count	count of connections having the same destination host and using the same Service	Continuous
34	Dst_host_same_srv_rate	% of connections having the same destination host and using the same Service	Continuous

S.No.	FeatureName	Description	Type
35	Dst_host_diff_srv_ rate	% of different services on the current Host	Continuous
36	Dst_host_same_src_ port_rate	% of connections to the current host having the same src port	Continuous
37	Dst_host_srv_diff_ host_rate	% of connections to the same service coming from different hosts	Continuous
38	Dst_host_serror_ rate	% of connections to the current host that have an S0 error	Continuous
39	Dst_host_srv_serror _rate	% of connections to the current host and specified service that have an S0 error	Continuous
40	Dst_host_rerror_rate	% of connections to the current host that have an RST error	Continuous
41	Dst_host_srv_rerror _rate	% of connections to the current host and specified service that have an RST error	Continuous

Tab. I *List of features available in KDD Cup '99 dataset.*

4.2.6 Crossover and mutation operators

Crossover operator explores the combinations of current chromosome while mutation operator generates new chromosome. There are 41 features present in the traffic pattern and these features may be independent or dependent on each other. If dependent features are away from each other in the chromosome, it is possible that single point crossover may destroy the schemata. To overcome this difficulty, uniform crossover is used. In uniform crossover, bits are randomly copied from the first or from the second parent chromosome depending on the value of mask. A mask is generated randomly with length equal to the length of the chromosome used for crossover. The mask determines which bits are copied from one parent and which bits from the other parent. Mutation inverts a bit in the population with a probability P_{m}. The role of mutation operator is to restore the lost genetic material. The parameters P_{c} and P_{m} are adjusted to achieve good results for the experiments conducted.

4.2.7 Selection operator

Selection operator selects chromosome from population of individuals for next generation. The proposed GA utilizes tournament selection to select the fittest chromosome. Tournament selection selects subgroup of chromosomes from the initial population where individuals within each subgroup compete against each other.

4.3 Rationale for the choice of neurotree as detection methodology

The proposed neurotree detection algorithm uses a bagging approach which generates multiple training datasets from the original KDD dataset and then trains multiple Back Propagation Neural Network (BPNN) named NNE using these multiple datasets. The predicted results produced by these NNs are combined based

on a new algorithm called voting algorithm. Then, the trained NNE is employed to generate a new training set by replacing the desired class labels of the original training examples, with those output from the trained NNE. Some extra training examples are also generated from the trained NNE and are added to the new training set. Finally, an enhanced C4.5 DT is grown from the new training set. The fusion of the improved NNE and enhanced C4.5 is used to detect the intrusions. From the experiments carried out in this work, it has been observed that this enhanced C4.5 provides higher detection rates on unseen samples and known samples.

4.3.1 Neural network ensemble

In the first step, it collects data randomly from the KDD dataset for each NN and constructs a training dataset using bagging approach. Bagging and boosting [2] are the two widely applied techniques for combining multiple classifiers in order to improve the prediction accuracy. These techniques aggregate multiple hypotheses produced by the same learning algorithm (NN) invoked over different distributions of KDD dataset. They generate a classifier with a smaller error on the dataset as it combines multiple hypotheses which individually have a huge error. In bagging [2] if a single classifier is unstable i.e., if it has high variance, the aggregated classifier (NNE) has a smaller variance than a single base classifier. Boosting generates a series of NNs whose training datasets are determined by the performance of the previous networks. Training instances that are wrongly predicted by the previous networks play a major role in the training of the later networks. This results in high error rate if the initial NN trained produces false prediction results. Also, bagging is more suitable if the induced classifier is unstable. As NN and DT are unstable classifiers (since these classifiers generate different classification accuracy as the number of records in the training dataset varies) bagging approach has been employed in the proposed work. Bagging creates a training dataset by sampling with replacement n times from the dataset containing n records. The created dataset has some duplicated records and some of the records are not selected from original dataset. These unpicked records are placed in the testing set. As the KDD dataset has large number of records, about 36.8% of the original records are placed in the testing set.

Consider the case in which there are n records in the dataset then the probability of particular record being picked is $1/\ n$. Therefore, the probability of record not being picked is $1-(1/n)$. As these records are picked n times then the chance that particular record not being picked is $(1-1/n)n \approx e-1 = 0.368$ for $n > 20$, where e is the base of natural logarithms (2.7183). From this, it can be concluded that around 36.8% of the original unique records are placed in the testing set and about 63.2% of unique records are placed in the training set. Some of the records are repeated in training set and the total size of the generated training set is the same as that of the original dataset.

In the second step, each of the NN is trained using training dataset generated to identify the network pattern based on feature vector. Some extra training examples are also generated from the trained NNE and are added to the training set. In the final stage, the trained NNE is employed to generate a new training set through re-

placing the desired class labels of the original training examples, with those output from the trained NNE. The network pattern is identified based on the predicted output from each of the NN using voting algorithm. The voting algorithm chooses the class label which receives the most number of votes as the final output of the ensemble.

4.3.2 Extended C4.5

DT induction algorithms have been applied in various fields. Some of the DT algorithms are ID3, C4.5 and C5.0 [1, 7, 21]. C4.5 is an extension of the basic ID3 algorithm. The proposed system utilizes enhanced C4.5 which is an improvement over C4.5. The ID3 and C4.5 algorithms utilize the information theoretic approach in classifying a network traffic pattern. The decision tree is initially created from the preclassified dataset. Each instance is defined by the values of the attributes. In this proposed work, there are 17 attributes which include protocol_type, service, flag etc as shown in Tab. II. A decision tree consists of nodes, edges and leaves. A node of a decision tree identifies an attribute by which the instance is to be partitioned. Every node has a number of edges, which are labeled according to a potential value of edges and a probable value of the attribute in the parent node. An edge links either two of the nodes in a tree or a node and a leaf. Leaves are labeled with class labels for classification of the instance. In DT based classification [21, 22], information gain is calculated for each of the attribute. The best attribute to divide the subset at each stage is selected using the information gain of the attributes. The instances are divided according to the values of these attributes. If the value of attributes is nominal then a branch for each value of the attribute is formed, but if it is numeric a threshold value is determined and two branches are created. This procedure is recursively applied to each partitioned subset of the instances. The procedure ceases when all the instances in the current subset belong to the same class. The concept of information gain tends to favor attributes that have a large number of values. For example, if there are set of records T and an attribute X that has a distinct value for each record, then Info*(X, T)* is 0, thus Gain*(X, T)* is maximal. To overcome this, an extended C4.5 algorithm is used in this work which employs gain ratio instead of information gain which takes into account the potential information from the partition itself. To categorize an unknown instance, the detection algorithm starts at the root of the DT and follows the branch indicated by the result of each test until a leaf node is arrived. The name of the class at the leaf node is the resulting classification.

4.3.3 Proposed neurotree algorithm

Algorithm: Neurotree algorithm.
Input: Network audit data $T = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ from MIT Lincoln Labs, extra data ratio η, trials of bootstrap sampling B, number of records in the training set n where x_i denote the list of attributes and y_i their corresponding class labels on Neural Network Ensemble NNE
Output: Set of Rules.

1. Train the NN from T via Bagging. Call the procedure NNE = (T, NN, B).
2. Initialize generated training set $T' = \phi$.
3. Process the original training set with the trained NNE and classify an instance x_i by counting votes for which NNE (x_i) represents the class with most votes:

   for ($i = 1$ **to** n)
   begin

   i. Replace the class label (y_i) with those output from the NNE $y_i' = \text{NNE}(x_i : (x_i, y_i) \in T)$.
   ii. Add the new samples generated to the generated training set $T' = T' \cup (x_i, y_i')$.

   end
4. Generate extra training data from the trained NNE:

   for ($j = 1$ **to** $\eta \cdot n$)
   begin

   i. Randomly generate attribute vector x_i' = Random() and feed it into NNE and add the outcome y_i' to the attribute vector as the last label. $y_i' = \text{NNE}(x_i')$.
   ii. Append the attribute vector (x_i', y_i') to the generated training set $T' = T' \cup (x_i', y_i')$.

   end
5. Read records from the NNE generated training set T' .
6. Tokenize each record and store it in an array.
7. Find the probability of occurrence for each value for each class.
8. Find the entropy

$$I(P) = -p_1 \log_2 p_1 + p_2 \log_2 p_2 + \cdots + p_n \log_2 p_n. \tag{2}$$

9. Calculate the information gain

$$\text{Gain}(X, T') = \text{Info}(T') - \text{Info}(X, T'). \tag{3}$$

10. Compute

$$\text{GainRatio}(X, T') = \text{Gain}(X, T')/\text{SplitInfo}(X, T'), \tag{4}$$

    where SplitInfo(X, T') is the information due to the split of on the basis of the value of the categorical attribute X. Thus $(T_1', T_2', \ldots, T_m')$ is the partition of induced by the value of attribute X.
11. Construct extended DT with the highest GainRatio attribute as the root node and values of the attribute as the arc labels.

12. Repeat steps 7 to 11 until categorical attributes or the leaf nodes are reached.
13. Derive rules following each individual path from root to leaf in the tree.
14. The condition part of the rules is built from the label of the nodes and the labels of the arcs: the action part is the classification (Normal or specific attack types).

Procedure Bagging (T, NN, B) returns NNE

for ($i = 1$ **to** B)
begin

1. Generate new training set of size n with replacements for each B trials, T_i= bootstrap sample from T.
2. Generate a classifier NN_i for each training set. Call procedure NeuralNetwork (T_i).
3. Form NNE classifier by aggregating the B classifiers.
4. Return trained NNE.

end

Procedure NeuralNetwork (TrainingSet T_i) returns NN

begin

1. Get input file T_i for training.
2. Read records from T_i.
3. Initialize weights and bias to random values.
4. Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer. Here, hidden layer is the layer between input and output layer.
5. Calculate the error at the outputs using ER, where ER = ($W_1\times$ FP + $W_2\times$ FN), FP is false positive rate, FN is false negative rate, W_1 and W_2 are their respective weight values.
6. Use the output error to compute error value for hidden layers.
7. Use the calculated error value to compute weight adjustments.
8. Apply the weight adjustments.
9. Repeat until the error value doesn't change in consecutive 5 iterations.
10. Return trained NN.

end

4.3.4 Performance measurement indices

Performance of IDS is evaluated using the following indices:

- Detection Rate (DR) — Ratio between number of anomaly (normal) correctly classified and total number of anomaly (normal).
- Error Rate (ER) — Ratio between number of anomaly (normal) incorrectly classified and total number of anomaly (normal).
- False Positive (FP) — Classifying normal class as an anomaly class.
- False Negative (FN) — Classifying anomaly class as a normal class.

These are good indices of performance, as they measure what percentage of intrusions the system is able to detect and how many incorrect classifications are made in the course of action.

Other metrics include

- Mean Absolute Error (MAE) — Average over the verification sample of the absolute values of the differences between forecast and the corresponding observation.
- Root Mean Squared Error (RMSE) — Quadratic scoring rule which measures the average magnitude of the error. Measures the difference between forecast and corresponding observed values, are each squared and then averaged over the sample. Finally, the square root of the average is taken.
- Relative Absolute Error (RAE) — Similar to the relative squared error in the sense that it is also relative to a simple predictor, which is just the average of the actual values.
- Relative Squared Error (RSE) — Calculates the total squared error and normalizes it by dividing the total squared error of the simple predictor.

5. Experimental analysis

The proposed model has been implemented using Java and the procedure used to implement the neurotree algorithm proposed in this work is as follows: The input to the system is given as an Attribute Relation File Format (ARFF) file. The dataset is created using the name specified in "relation". The attributes are specified under "attribute" correspondingly specifying the type of attribute and instances specified under "data" are retrieved from the ARFF file and then they are used for training by the classifier. This procedure is followed for test set also. The classifier has been trained and evaluated using the preprocessed dataset, formed from the KDD dataset [15].

5.1 Test objectives

1: To investigate the impact of proposed feature extraction algorithm on the performance of IDS.

2: To identify the set of misuse sensitive features and to find out the discriminative power of selected features.

3: To evaluate the impact of proposed neurotree classification algorithm on the proposed framework.

4: To investigate the detection capability of neurotree in two different test scenarios.

i: Dataset with five partitions namely Normal, DoS, U2R, R2L and Probe classes.

ii: Dataset with 23 partitions containing Normal and specific attack type classes such as Neptune, Back, Smurf, Buffer_overflow etc.

5: To investigate the impact of training and test dataset on detection accuracy.

5.2 Influence of feature vector

In this experiment, different feature vectors have been selected by the feature selection algorithms to see the influence on detectability of the proposed approach. Initially, to extract relevant features using proposed method, dataset with 41 attributes of feature vector and 22 different attack types is considered. These relevant features extracted are written into a file with their corresponding data values for each and every class. The above procedure is repeated with the other feature extraction algorithm and the results have been stored for comparison. The results obtained using the proposed feature extraction has been found to be better when compared to the existing algorithms. This satisfies objective 1 and objective 2.

5.3 Influence of neurotree classifier

To test the performance of the proposed method, the training and test dataset consists of network traffic patterns from KDD dataset [15]. It formed a network traffic database containing diverse data with 22 different attack types and normal class with selected feature vector to satisfy objective 4-scenario 2. Another database is formed in which class labels present in the existing dataset is replaced by DoS, R2L, Probe and U2R class accordingly with selected feature vector in order to satisfy objective 4-scenario 1. Cross-validation has been applied to the dataset formed for estimating the generalization error based on re-sampling and to estimate how accurately the intrusion detection paradigm performs in real time.

Experiments have been conducted using stratified 10-fold cross-validation [11]. In stratified 10-fold cross-validation, the sample of data instances are split into 10 approximately equal partitions such that the mean response value is approximately equal in all the partitions, i.e., each partition contains roughly the same proportions of all types of class labels present in the original dataset. After partitioning, 9/10 of

dataset is used for training and 1/10 of dataset is used for testing. This procedure has been repeated 10 times and the overall error rate is calculated by taking average of error rates on each partition, i.e., the final output is the average result of these ten folds. After ten-fold cross-validation, it has been found that TP rate and FP rate of classes like imap, phf, perl, spy, and multihop are nil as the number of records in these class types are less than 10 and therefore these records are not present in most of the partition. The results obtained shows that the proposed neurotree classification algorithm is able to detect intrusive activities in network. This satisfies objective 3.

5.4 Detection with mismatch between training and test dataset

The performance of the proposed work is analyzed using different types of attacks in training and testing. The training set and the test set are denoted as TR and TE respectively. First, we created TR by including anomaly records of 22 attack types and some normal traffic patterns. On the other hand, TE is constituted of anomaly traffic patterns of 22 attack types and normal patterns that are not present in the training dataset. Additionally, 17 new attack types (like Mailbomb, Saint, Sqlattack etc.) that are not present in TR have been also included. Essentially, TR and TE are made disjoint. This satisfies objective 5.

5.5 Redundancy check

In this analysis, redundant records present are replaced by a single copy. After redundancy check, it has been found that the anomaly class had more redundancy than the normal class. Some invalid records which are found in the original KDD are also removed. This process assists the neurotree learner from getting biased towards frequent records.

6. Test results and discussions

6.1 Feature extraction

The purpose of this work is to examine the various existing attribute selection algorithms in terms of detection accuracy and to compare those algorithms with the proposed algorithm. Out of the total 41 network traffic features, used in detecting intrusion, some features are potential in revealing evidence of anomalies. Therefore, such predominant features are extracted from the 41 features.

6.2 Attribute evaluators

Attribute evaluator is used for ranking all the features according to some metric. Various attribute evaluators available in Waikato Environment for Knowledge Analysis (WEKA) [31] are used in this work which includes CfsSubsetEval, ChiSquaredAttributeEval, ConsistencySubsetEval, InfoGainAttributeEval and GainRatioAttributeEval.

S.No.	Algorithm	No. of features selected	Detection rate
1	BestFirst+ Consistency SubsetEval	11	97.01
2	GeneticSearch + CfsSubsetEval	20	98.16
3	GeneticSearch + ConsistencySubsetEval	20	97.86
4	GreedyStep wise + CfsSubsetEval	9	97.97
5	Ranker+ ChiSquared AttributeEval	36	98.13
6	RankSearch + CfsSubsetEval	22	98.40
7	RankSearch + ConsistencySubsetEval	34	98.15
8	Proposed Feature Selection	17	98.93
9	NIL	41	98.96

Tab. II *List of features selected by various feature selection algorithms.*

6.3 Search methods

These methods search the set of all possible features in order to find the best set of features. Five search methods including BestFirst, GeneticSearch, GreedyStepwise, Ranker and RankSearch available in WEKA [31] are used in this proposed work for comparison purpose. The details of the combinations and the features selected by each combination and their detection accuracy are in Tab. II.

The proposed algorithm uses weighted sum genetic feature selection algorithm to select distinguishing features. The GA parameters are: Chromosome Length = 41 (one gene per network traffic feature), Population Size = 100, Crossover Probability $P_{\mathrm{c}} = 0.7$ and Mutation Probability $P_{\mathrm{m}} = 0.001$. It has been found that the best set of features is selected within 2000 generations by the genetic algorithm. Finally, 17 salient features as in Tab. II have been selected by the proposed genetic algorithm and the classification is based on these predominant features.

Tab. II proves the findings of Breiman et al. [15] that detection accuracy is not much affected by the choice of attribute selection measure. Although the detection accuracy obtained by various algorithm is near to the proposed algorithm, the number of features selected by the proposed feature selection algorithm is less when compared to other algorithms. Thus the detection time of intrusion detection by using the features selected by the proposed feature selection algorithm is less when compared to other algorithms. However, the number of features selected by BestFirst+ ConsistencySubsetEval and GreedyStepwise + CfsSubsetEval is less when compared to other algorithms but their detection accuracy is reduced by approximately 2.

6.4 Building and training classifier

A multi-layer feed forward NN has been used in this proposed work due to its simple structure and easy realization. Back-propagation algorithm has been used to train the network in order to adjust the weights effectively to reduce the errors. The number of NNs used for bagging is 10. Each NN in the NNE consists of input

layer, hidden layer and output layer. The number of input nodes in the input layer is equal to the number of selected features (in the proposed work, 17 input nodes are used) and the output layer based on number classes used. For example, 5 nodes in the output layer indicate the classes namely normal, DOS attack, probing attack, R2L attack and U2R attack. The value of the output node ranges from 0 to 1. The output node with the highest value is chosen and its corresponding class is named as its output.

Hidden layer consists of number of nodes (called hidden nodes) that connect the input and output nodes. The number of neurons in the hidden layer must be chosen carefully. This is because if it is too small, the network will not be trained adequately to acquire the learning behaviour of the series. If it is too large, the network will become exceedingly specialized and thereby loses its generalizing capability. Therefore, we adapted the design rules suggested by Baum [3] for carrying out experiments to achieve better performance. According to the rule, the preferred number of hidden layers is 2 and the combination of first hidden layer with 28 nodes and the second hidden layer with 16 neurons combination worked out well. Various NN architectures have been examined by varying the number nodes in the hidden layers to test their learning performance in terms of fitness value. The network with 17-28-16-5 neurons is better than those of the other architectures.

Activation functions for hidden layers are required in order to introduce non-linearity into the network and it is the non-linearity that makes multilayer feed forward networks very powerful. For back propagation learning, the activation function must be differentiable and therefore sigmoidal function has been utilized in this work. The maximum number of epochs for training NN is set as 1000. In order to avoid over-fitting, training of NN is stopped when the error value does not change in the consecutive five epochs. Moreover, an extended C4.5 DT algorithm has been implemented to classify the records. The generated training records are given and the corresponding gain ratio for each of the attribute is calculated. The discrete and continuous attributes are identified from the input records. Then the tree is constructed based on the gain ratio. Following the each individual path in the tree, the rules are generated.

6.5 Evaluation of neurotree on proposed framework

The evaluation of the proposed neurotree algorithm has been carried out using the detection rate (DR) as the metrics.

6.5.1 Performance comparisons of neurotree with NN and C4.5

To demonstrate the increase in detection performance, the detection rate of neurotree classifier with other classifiers like C4.5 and NN has been compared. The performance comparison of the proposed system with C4.5 and NN has been presented in Tab. III. It has been observed that the neurotree provides a better detection performance in classifying the network traffic patterns. Although, the DRs for U2R and R2L are less for NN and C4.5, the neurotree approach proposed and implemented in this work outperforms these two techniques and provides detection rate more than 86% for both U2R and R2L type of attacks. Fig. 2 shows the detection rate comparison of NN based IDS, C4.5 based IDS and neurotree based IDS.

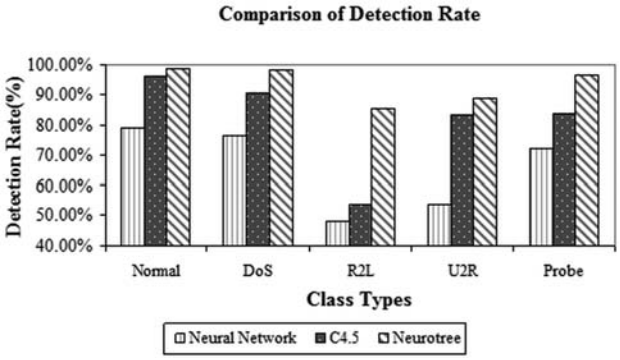

Fig. 2 *Detection rate comparisons of NN, C4.5 and neurotree.*

From the graph, it has been observed that the proposed neurotree model based IDS outperforms the other techniques especially in R2L and U2R attack types.

Tab. IV shows the error rate of normal and various anomaly classes. It also depicts the number of test samples incorrectly classified by each of the classifier namely NN, C4.5 and neurotree. Also, the overall error rates of each of three approaches are depicted. From Tab. IV, it has been observed that error rate for neurotree is 20% less when compared to NN. Fig. 3 shows the error rate comparison of various classes. It has been observed that among the different attack types, R2L has higher error rate of 14.5% when compared with other classes for neurotree.

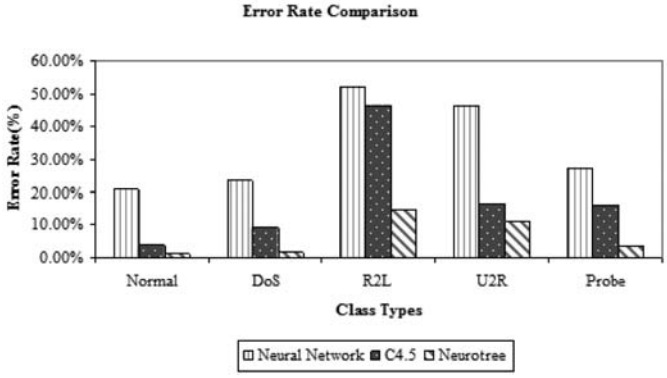

Fig. 3 *Error rate comparisons of various IDS.*

6.5.2 Analysis of error rate of NNE

Reduction of error is more important in IDS as mentioned in the previous section. It could be noted from Tab. V that the decrease in the mean absolute error, on an average, is 0.32 from 0.56% to 0.24%. Similarly, the root mean squared error has a drop of 0.98 from 4.53% to 3.55% on an average, relative absolute error has reduced by 5.7393 from 10.1917% to 4.4524% and root relative squared error has

S.No.	Class Types	Total Test Samples	Neural Network		C4.5		Neurotree	
			Correctly Classified	Detection Rate	Correctly Classified	Detection Rate	Correctly Classified	Detection Rate
1	Normal	47911	38011	0.7934	46008	0.9603	47313	0.9875
2	DoS	7458	5701	0.7644	6756	0.906	7331	0.983
3	R2L	2754	1320	0.4793	1480	0.5374	2354	0.8548
4	U2R	200	107	0.535	167	0.835	178	0.89
5	Probe	2421	1756	0.7253	2033	0.8397	2337	0.9653
	Overall Accuracy	60744	46895	0.7720	56444	0.9292	59280	0.9797

Tab. III *Detection rate comparisons of NN, C4.5 and neurotree approaches.*

S.No.	Class Types	Total Test Samples	Neural Network		C4.5		Neurotree	
			Correctly Classified	Detection Rate	Correctly Classified	Detection Rate	Correctly Classified	Detection Rate
1	Normal	47911	9900	0.2066	1903	0.0398	598	0.0125
2	DoS	7458	1757	0.2357	702	0.0941	127	0.0170
3	R2L	2754	1434	0.5207	1274	0.4626	400	0.1452
4	U2R	200	93	0.465	33	0.165	22	0.11
5	Probe	2421	665	0.2747	388	0.1603	84	0.0347
	Overall Error	60744	13849	0.228	4300	0.0708	1231	0.0203

Tab. IV *Error rate comparisons of NN, C4.5 and neurotree approaches.*

reduced to 21.4525. It has been noted that there is a change in total error as the error function of NNE is changed based on the ratio of false positive error and false negative error. Thus, the proposed activation function which has been designed to reduce the error rates and hence provides a better classification. This proves that the claim is valid. i.e., reduction of false alarm errors and increase of the detection rates of the classifier.

Evaluation Metrics	NNE	Extended C4.5	Neurotree
Mean absolute error	0.0056	0.0026	0.0024
Root mean squared error	0.0453	0.0515	0.0355
Relative absolute error [%]	10.1917	4.8379	4.4524
Root relative squared error [%]	27.3176	31.1162	21.4525
Detection Rate [%]	97.3621	97.08	98.9341

Tab. V *Evaluation metrics comparisons of NNE, extended C4.5 and neurotree.*

6.5.3 Investigation of detection capability under 2 scenarios

A. Scenario 1

In Scenario 1, dataset is formed using five classes (namely DoS, Normal, R2L, U2R and Probe) and with 17 features selected using weighted sum GA. In this testing phase, the detection rate of R2L class is very low when compared to other classes. The overall detection rate is 97.97%. The TP rate and FP rate for each class is shown in Tab. VI.

B. Scenario 2

In Scenario 2, dataset is formed using 23 classes like smurf, back, normal etc. with 17 features selected using weighted sum GA. The overall accuracy is 98.9%. From this, it can be concluded that neurotree detection paradigm performs better even when specific attack types are provided. The TP and FP value obtained for each of 23 classes is shown in Tab. VII.

S.No	Class	TP Rate	FP Rate
1	Normal	0.9875	0.0468
2	DoS	0.983	0.0048
3	R2L	0.8548	0.0024
4	U2R	0.89	0
5	Probe	0.9653	0.004

Tab. VI *TP and FP rate obtained using neurotree for five classes.*

From Tab. VII, it has been observed that the TP rate for Neptune, Teardrop, Pod and guess_passwd is 100%. Also, it has been found that most of the classes have a TP rate above 98%. In addition, after ten-fold cross-validation it has been

S.No.	Class	TP Rate	FP Rate
1	Normal	0.997	0.005
2	Warezclient	0.884	0
3	Neptune	1	0
4	Ipsweep	0.986	0.001
5	Teardrop	1	0
6	Satan	0.977	0
7	Portsweep	0.985	0
8	Smurf	0.998	0
9	Nmap	0.983	0
10	Warezmaster	0.714	0
11	Back	0.99	0
12	Land	0	0
13	Pod	1	0
14	buffer_overflow	0.833	0
15	Loadmodule	0	0
16	Rootkit	0	0
17	guess_passwd	1	0
18	ftp_write	0	0
19	Imap	0	0
20	Spy	0	0
21	Perl	0	0
22	Multihop	0	0
23	Phf	0	0

Tab. VII *TP and FP rate obtained using neurotree for 23 classes.*

found that TP rate and FP rate of classes like imap, phf, perl, spy, and multihop are nil as the number of records in these class types are less than 10 and therefore these records are not present in most of the partition.

7. Conclusions

This work proposes an IDS by integrating NN and DT for multi-class categorization. The system is aimed at making improvements over the existing work in two perspectives. A neurotree model is employed as the classification engine which imparted a detection rate of 99% and an error rate of 1% for 23 classes and 98% for five classes which is superior to NN and C4.5. Also, an error function based on FN rate and FP rate is utilized in the learning algorithm which has been targeted at minimizing the error rates. Experiments conducted summarizes the characteristics of this proposed method with various performance metrics like TP rate, FP rate, accuracy, DR and with various error metrics like MAE, RMSE etc. It has been observed that the proposed system performs well even when the dataset has different number of classes and completely unseen data. This justifies that the proposed method is a promising strategy to be applied on intrusion detection.

References

[1] AMOR N.B., BENFERHAT S., ELOUEDI Z. Naive Bayes vs decision trees in intrusion detection systems. In: L.M. LIEBROCK, ed. *Proceedings of ACM Symposium on Applied Computing*, Cyprus. New York: ACM, 2004, pp. 420–424.

[2] BAUER E., KOHAVI R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants machine learning. *Machine Learning*. 1999, 36(1), 105–139, doi: 10.1023/A:1007515423169.

[3] BAUM E.B., HAUSSLER D. What size net gives valid generalization? *Neural Computation*. 1998, 1(1), 151–160, doi: 10.1162/neco.1989.1.1.151.

[4] BENFERHAT S., TABIA K. On the combination of naive Bayes and decision trees for intrusion detection. In: M. MOHAMMADIAN, ed. *Proceedings of IEEE International Conference on Computational Intelligence for Modelling, Control and Automation*, Vienna, Austria. IEEE, 2005, pp. 211–216.

[5] BREIMAN L., FRIEDMAN J., STONE C.J., OLSHEN R.A. *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[6] CANNADY J. Artificial neural networks for misuse detection. In: R. POSCH, ed. *Proceedings of National Information Systems Security Conference*, Arlington, Virginia, USA. National Institute of Standards and Technology, 1998, pp. 443–456.

[7] COHEN S., ROKACH L., MAIMON O. Decision-tree instance-space decomposition with grouped gain-ratio. *Information Sciences*. 2007, 177(1), 3592–3612, doi: 10.1016/j.ins.2007.01.016.

[8] COVOES T.F., HRUSCHKA E.R. Towards improving cluster-based feature selection with a simplified silhouette filter. *Information Sciences*. 2011, 181(1), 3766–3782, doi: 10.1016/j.ins.2011.04.050.

[9] FAYYAD E.M., UTHURUSAMY R. Evolving data mining into solutions for insights. *ACM Communication*. 2002, 45(1), 28–31, doi: 10.1145/545151.545174.

[10] GADDAM S.R., PHOHA V.V., BALAGANIR K.S. K-Means+ID3: A novel method for supervised anomaly detection by cascading K-Means clustering and ID3 decision tree learning methods. *IEEE Transactions on Knowledge and Data Engineering*. 2007, 19(1), 345–354, doi: 10.1109/TKDE.2007.44.

[11] GARCIA S., FERNANDEZ A., LUENGO J., HERRERA F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*. 2010, 180(1), 2044–2064, doi: 10.1016/j.ins.2009.12.010.

[12] HAN S.-J., CHO S.-B. Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Transactions on System, Man, Cybernetics, Part B*. 2006, 36(1), 59-70, doi: 10.1109/TSMCB.2005.860136.

[13] HU Y.-C. Analytic network process for pattern classification problems using genetic algorithms. *Information Sciences*. 2010, 180(1), 2528–2539, doi: 10.1016/j.ins.2010.03.008.

[14] JOO D., HONG T., HAN I. The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors. *Expert Systems with Applications*. 2003, 25(1), 69–75, doi: 10.1016/S0957-4174(03)00007-1.

[15] KDD CUP 1999 DATA. Competition Dataset [online]. The UCI KDD Archive, Information and Computer Science, University of California, Irvine, 2007 [viewed 2014-06-01]. Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[16] KOUTSOUTOS S., CHRISTOU I.T., EFREMIDIS S. A classifier ensemble approach to intrusion detection for network-initiated attacks. In: I. MAGLOGIANNIS, K. KARPOUZIS, M. WALLACE, J. SOLDATOS, eds. *Proceedings of the International Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, Vienna, Austria. Amsterdam: IOS Press, 2007, pp. 307–319.

[17] LINDA O., VOLLMER T., MANIC M. Neural network based intrusion detection system for critical infrastructures. In: A. MINAI, ed. *Proceedings of IEEE International Joint Conference on Neural Networks*, Atlanta, Georgia, USA. IEEE, 2009, pp. 102–109.

[18] LIU H., YU L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*. 2005, 17(1), 491–502, doi: 10.1109/TKDE.2005.66.

[19] MENAHEM E., SHABTAI A., ROKACH L., ELOVICI Y. Improving malware detection by applying multi-inducer ensemble. *Computational Statistics and Data Analysis*. 2009, 53(1), 1483–1494, doi: 10.1016/j.csda.2008.10.015.

[20] MITRA P., MURTHY C.A., PAL S.K. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002, 24(1), 301–312, doi: 10.1109/34.990133.

[21] QUINLAN J.R. Induction of decision trees. *Machine Learning*. 1986, 1(1), 81–106, doi: 10.1007/BF00116251.

[22] RAJESWARI L.P., KANNAN A. An active rule approach for network intrusion detection with enhanced C4.5 algorithm. *International Journal of Communications, Network and System Sciences*. 2008, 4(1), 285–385, doi: 10.4236/ijcns.2008.14039.

[23] ROKACH L. Ensemble-based classifiers. *Artificial Intelligence Review*. 2010, 33(1), 1–39, doi: 10.1007/s10462-009-9124-7.

[24] ROKACH L. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics and Data Analysis*. 2009, 53(1), 4046–4072, doi: 10.1016/j.csda.2009.07.017.

[25] SARASAMMA S., ZHU Q., HUFF J. Hierarchical Kohonen net for anomaly detection in network security. *IEEE Transactions on System, Man, Cybernetics, Part B*. 2005, 35(1), 302-–312, doi: 10.1109/TSMCB.2005.843274.

[26] SHON T., MOON J. A hybrid machine learning approach to network anomaly detection. *Information Sciences*. 2007, 177(1), 3799–3821, doi: 10.1016/j.ins.2007.03.025.

[27] SHUN J., MALKI H.A. Network intrusion detection system using neural networks. In: M. GUO, ed. *Proceedings of Fourth IEEE International Conference on Natural Computation*, Jinan, China. IEEE, 2008, pp. 242–246.

[28] STEIN G., CHEN B., WU A.S., HUA K.A. Decision tree classifier for network intrusion detection with GA-based feature selection. In: M. GUIMARAES, ed. *Proceedings of the 43rd ACM Annual Southeast Regional Conference*, Georgia, USA. ACM, 2005, pp. 136–141.

[29] TAVALLAEE M., BAGHERI E., LU W., GHORBANI A.A. A detailed analysis of the KDD Cup 1999 data set. In: S. WESOLKOWSKI, ed. *Proceedings of IEEE Symposium on Computational Intelligence in Security and Defense Applications*, Ottawa, Canada. IEEE, 2009, pp. 53–58.

[30] THOMAS C., BALAKRISHNAN N. Improvement in intrusion detection with advances in sensor fusion. *IEEE Transactions on Information Forensics and Security*. 2009, 4(1), 542–551, doi: 10.1109/TIFS.2009.2026954.

[31] WEKA 3. Weka 3: Data Mining Software in Java v.3.5.7. [software]. 2008 [accessed 2014-06-01]. Available from: http://www.cs.waikato.ac.nz/ml/weka/

[32] WEIMING H., WEI H., MAYBANK S. AdaBoost based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*. 2008, 38(1), 577–583, doi: 10.1109/TSMCB.2007.914695.

[33] XIANG C., YONG P.C., MENG L.S. Design of multiple-level hybrid classifier for intrusion detection system. *Journal of Pattern Recognition Letters*. 2008, 29(7), 918-–924, doi: 10.1016/j.patrec.2008.01.008.

[34] YASAMI Y., MOZAFFARI S.P. A novel unsupervised classification approach for network anomaly detection by K-Means clustering and ID3 decision tree learning methods. *The Journal of Supercomputing*. 2009, 53(1), 231–245, doi: 10.1007/s11227-009-0338-x.

[35] ZHOU Z.-H., JIANG Y. NeC4.5: Neural ensemble based C4.5. *IEEE Transactions on Knowledge and Data Engineering*. 2004, 16(1), 770–773, doi: 10.1109/TKDE.2004.11.