An Intrusion Detection Approach Based On Understandable Neural Network Trees

Qinzhen Xu[†], *Wenjiang Pei*[†], *Luxi Yang*[†], *and Qiangfu Zhao* [††],

[†]School of Information Science and Engineering, Southeast University, Nanjing, 210096, China
[††]Multimedia Device Laboratory, Aizu University, Aizu-wakamatsu 965-8580, Japan

Summary
In this paper we presented a novel intrusion detection mode based on understandable Neural Network Tree (NNTree). NNTree is a modular neural network with the overall structure being a Decision Tree (DT), and each non-terminal node being an Expert Neural Network (ENN). One crucial advantage of using NNTrees is that they keep the non-symbolic model ENN's capability of learning in changing environments. Another potential advantage of using NNTrees is that they are actually "gray boxes" as they can be interpreted easily if the number of inputs for each ENN is limited. The experimental results are demonstrated on the KDD Cup 1999 intrusion detection dataset. The results show that the trained NNTree achieves a simple ENN at each non-terminal node as well as a satisfying recognition rate of the network packets dataset. Furthermore, the learned model contains the information about those features of critical importance for detection. One can accordingly focus his observation on those important attributes and hence decrease the complexity of the feature space. Besides, the learning results may also enlighten researchers on exploring new features for those difficult detecting attack types.

Key words:
Intrusion detection; Neural Network Tree; Expert Neural Network; Decision Tree; Self-organized feature learning.

1. Introduction

The rapid expansion in the degree of interconnection of computer systems has caused the rising insecurity of information assets. Identifying unauthorized usage of a computer system is thus become increasing important. Intrusion detection systems are the right systems designed to implement the identifying process. In recent years, numerous researchers have investigated in this field. Those interesting methodologies include statistical models, immune system approaches, protocol verification, file and taint checking, neural networks, whitelisting, expression matching, state transition analysis, dedicated languages, genetic algorithm, and burglar alarms[1]. Some of the examples are as follows. Sung-Bae incorporated neural network and fuzzy logic into the Intrusion Detection System (IDS) based on anomaly detection using Hidden Markov Model (HMM)[2]. He also proposed an effective HMM-based IDS that improves the modeling time and performance by only considering the privilege transition flows based on the domain knowledge of attacks[3]. His latest work described an intrusion detection technique based on evolutionary neural networks, which determined the structure and weights of network during evolution process[4]. Richard and Robert proposed a method to improve the performance of IDS by improving the baseline keywords and using discriminant neural network[5]. Azzedine and Kathia, *et al.* used an immune based Intrusion Detection Mode to achieve a significant size reduction of the logs file[6]. David and Daniel took the advantage of multi-resolution wavelet analysis to compress the traffic, and subsequently investigated the effect of compression upon the reconstructed signal's self-similarity, as measured by its estimated Hurst parameter [7].

In this paper, a novel intrusion detection approach based on understandable Neural Network Tree (NNTree) is proposed for intrusion detection. Models for machine learning can be roughly divided into two categories: symbolic and non-symbolic (or sub-symbolic). Symbolic models such as Decision Tree (DT) are generally considered understandable because a reasoning procedure can be provided for each decision. However, they are not good for learning in changing environments. Each time some new data are observed, the system must be revised substantially or a new system must be designed again. On the other hand, non-symbolic models, such as neural networks, are good at learning in changing environments because the system can be revised easily through re-training the free parameters. However, non-symbolic models are usually "black-boxes" and cannot be understood easily. In many situations we need a learning model that can have the advantages of both symbolic and non-symbolic models. For example, in intrusion detection problem, it is required to revise the learner or refine the knowledge using data observed each day or even each minute. At the same time, it also needs to give the understandable reasons for making decisions to confirm the reliability. A simple way to solve this problem is to learn by a non-symbolic model (e.g., a neural network), and to interpret by a symbolic model (say, a Boolean function). In general, however, the problem for interpreting a trained neural network is NP-complete [8]. To solve the above problem, we introduced the NNTree in

Manuscript received October 5, 2006.
Manuscript revised November 12, 2006.

our study[9]. An NNTree is a DT with each non-terminal node being an Expert Neural Network (ENN). Generally speaking, NNTrees are more powerful than traditional DTs because the ENNs can extract more complex and better features for making decisions[10], and the performance can be improved through re-training[11]. Further, an NNTree can be interpreted easily if we restrict the number of inputs of each ENN[12]. Thus, if we use NNTrees, the NP-complete problem for interpretation can be avoided.

The organization of this paper is as follows. In the next section, the evolutionary design process of NNTrees with self-organized feature learning is provided. In Section 3, the intrusion detection experimental results are presented to verify the proposed mode. Section 4 gives the conclusions.

2. Intrusion Detection Approach Based on Understandable NNTree

2.1 Split of internal nodes

An NNTree is a hybrid model with the overall structure being a DT and each non-terminal node of DT being an ENN. To construct a DT, it is often assumed that a training set consisting of feature vectors and their corresponding class labels are available. The DT is then constructed by partitioning the feature space in such a way as to recursively generate the tree. The most important step in the construction process is splitting the nodes. One of the popular algorithms for designing DT is C4.5[13]. In C4.5, the information gain ratio is used as the criterion for splitting nodes. The basic idea is to partition the current training set in such a way that the average information required to classify a given example can be reduced most. Let S stands for the current training set (with $|S|$ training examples), and n_i the number of cases belonging to the i-th class ($i = 1, 2, \cdots, N$), the average information (entropy) needed to identify the class of a given example is

$$\text{info}(S) = -\sum_{i=1}^{N} \frac{n_i}{|S|} \log_2 \left(\frac{n_i}{|S|}\right) \tag{1}$$

Now suppose that S is partitioned into n sub-sets $S_1, S_2, \cdots, S_n$ by some test, the information gain is given by

$$\text{gain}(X) = \text{info}(S) - \text{info}_X(S) \tag{2}$$

Where

$$\text{info}_X(S) = \sum_{i=1}^{n} \frac{|S_i|}{|S|} \text{info}(S_i) \tag{3}$$

By analogy with the definition of entropy, the potential information generated by partition is

$$\text{split\_info}(X) = -\sum_{i=1}^{n} \frac{|S_i|}{|S|} \log_2 \left(\frac{|S_i|}{|S|}\right) \tag{4}$$

Then information gain ratio is defined as follows

$$\text{gain\_ratio}(X) = \text{gain}(X) / \text{split\_info}(X) \tag{5}$$

To design a decision tree recursively, each time the current training set is partitioned into several sub-sets by testing the value of one of the features. If the feature has n values, there will be n sub-sets. If the feature is continuous, there are two sub-sets. The point is to select the feature to maximize the information gain ratio.

In an NNTree, all non-terminal nodes are ENNs. Each ENN in this study is a Multi-Layer Perceptron (MLP) with one hidden layer and 2 output neurons. For any given training example, it is assigned to the i-th ($i = 0,1$) sub-set if the i-th output neuron has the largest value. The point here is to find an ENN to maximize the information gain ratio. To find an ENN for each non-terminal node, we cannot use supervised learning algorithms such as BP (Back-Propagation) because there are no teacher signals. Evolutionary Algorithms (EAs) are more suitable for this purpose. In this study, we just adopted the Standard Genetic Algorithms (SGAs). There are three basic operators: truncate selection, one-point crossover, and bit-by-bit mutation. The genotype of an MLP is the concatenation of all weight vectors (including the threshold values) represented by binary numbers. Information gain ratio is defined as one item of the fitness.

2.2 Understandable NNTree

The crucial advantage of using NNTrees is that they keep the non-symbolic model ENN's capability of learning in changing environments as described in our previous work[9]. As a potential advantage for intrusion detection, in this paper, we also consider the understanding aspect of the NNTrees. The future purpose is to propose methods for designing intrusion detection approach based on NNTrees that are both interpretable and comprehensible. By interpretable here we mean that the NNTree can be interpreted easily. By comprehensible we mean that the knowledge obtained through interpretation is understandable. When we consider interpretation of an NNTree using a Boolean function, the understandability of an NNTree can be measured using the complexity of the Boolean function obtained from it. This measure, however, cannot be used during learning. Instead, we can measure the understandability using the complexity of the ENNs, which in turn can be measured using the number of hidden neurons. The rationale is that an ENN with more hidden neurons will produce a more complex decision surface, and a more complex Boolean function must be used to interpret it. With the assumption that the inputs are all binary numbers, the interpretability of an NNTree can be measured using the computational cost, which is given by $\text{Cost} = O(N \times 2^m)$, where N is the number of non-terminal nodes, and m is the upper bound of the number of inputs for each ENN. The NNTree will be interpretable if m is a small integer.

If we consider interpretability and comprehensibility together, designing an NNTree becomes a Multiple Objective Optimization (MOO) problem. The following objectives must be optimized together: (1) Maximize the partition ability of the ENNs. (2) Maximize the interpretability of the ENNs. (3) Maximize the comprehendsibility of the ENNs. To solve this problem, we proposed a simple MOO based Genetic Algorithm (MOO-GA)[14]. Starting from the root node, the MOO-GA is used recursively for designing the ENNs of the non-terminal nodes. Note that in our study, the partitioning ability is measured by the information gain ratio.

One approach for solving MOO problems is to use weighted sum method, to reduce the objectives to a single one. In this method, we often consider one of the objectives as the main goal to optimize, and all others are used as penalties. In our study, we introduced a simple MOO based GA for evolutionary design of NNTrees[14]. The algorithm is a modified version of Goldberg's method, which is a kind of Pareto-ranking approach[15]. The information gain ratio and the number of hidden neurons are taken as the two fitness items, i.e. the node has both a high information gain ratio and a small number of hidden neurons ranks ahead. The modification is to sort the individual ENNs again according to their information gain ratio when they have the same rank. The modified algorithm can provide one (the best individual) unique solution, rather than a set of non-dominating solutions. This is a simple method for automatic selection of the best solutions. It is also reasonable because we usually assign a higher priority to individuals with larger information gain ratio, or better partitioning ability. Note that it is very important to keep the partitioning ability of the ENNs while trying to reduce the number of hidden neurons and the number of inputs. If the partitioning ability is reduced, the tree can become very large, even if each ENN is small. Such kinds of NNTrees are no more comprehensible. If we use ENNs with proper number of hidden neurons and proper number of inputs, the NNTree can be learnable, reliable, interpretable and comprehensible.

2.3 Self-organized feature learning

So far we have considered the case in which all inputs are binary. However, in many applications (e.g. intrusion detection), the inputs are continuous. If the inputs of an NNTree are real numbers represented by binary numbers of B bits, the cost for interpretation will be $\text{Cost} = O(N \times 2^{mB})$. For instance, if $m = 4$ and $B = 16$, the computation cost will be proportional to $N \times 2^{64}$. Thus, even if the number of inputs for each ENN is small, the NNTree may not be interpretable. To reduce the computational complexity, especially the spatial complexity, we propose in this paper to discretize the continuous inputs using self-organized learning in each dimension. The basic idea is to find a small number of critical points for each continuous input by self-organized feature learning, and discretize the input using the critical points. For example, if there are 8 critical points for each continuous input, the total number of binary inputs for each ENN will be less than $3m$, and the interpretability of the NNTree will be much better than uniform discretization. Since the values for each input are usually not uniformly distributed, non-uniform discretization can reduce the number of discrete values greatly.

At the beginning, we have also tried to draw the class labels along each dimension and discretize the feature according to the number of changing points. However, this method is not good because too many discrete points are usually produced. In this paper, we discretize the features of the datasets in two steps. First, we draw the histogram for each dimension and decide roughly the number of critical points according to the distribution of the higher peaks in the histogram. Then we find the positions of the critical points using one-dimensional self-organized learning. In this paper, the well-known Winner-Take-All (WTA) learning rule is adopted[16].

Suppose that C critical points, $w_1^k, w_2^k, \cdots, w_C^k$, are used to discretize the k-th feature, the critical points can be found as follows:

Step1 Initialize $w_i^k (i = 1, 2, \cdots, C)$ at random. Usually, both the feature values and the critical points are normalized so that they take values from [0,1].

Step2 Given a training example x, find the winner $w_i^k (i = 1, 2, \cdots, C)$ such that

$$\left| x^k - w_i^k \right| = \min_{1 \le j \le m} \left| x^k - w_j^k \right| \tag{6}$$

where x^k is the k-th element of x.

Step3 Modify the winner as follows:

$$w_i^k \leftarrow w_i^k + \alpha (x^k - w_i^k) \tag{7}$$

where $\alpha \in (0,1)$ is the learning rate.

Step4 Terminate the iteration if some condition is satisfied; otherwise, return to Step 2.

In this study, the initial value of the learning rate is 0.5, and it is reduced linearly in each learning cycle. The terminating condition is simply to see if the number of learning cycles (or epochs) reaches to a given number T, which is set to 100 through our experiments.

3. Experimental Results and Discussion

Experiments were conducted to verify the performance of intrusion detection approach based on understandable NNTrees. We trained the NNTrees using 1000 instances randomly selected from the shuffled dataset, and tested the NNTrees using 4000 instances randomly selected from the remained dataset. All the experimental data is available from the *corrected* intrusion data set of KDD cup 1999. It includes 60593 normal instances and 4 types of attack data listed in Table 1. The continuous features include *duration, src_bytes, dst_bytes,wrong_fragment, urgent, hot, num_failed_logins, num_compromised, root_shell, su_attempted, num_root, num_file_creations, num_shells, num_access_files, num_ outbound_cmds, count, srv_count, serror_rate, srv_serror _rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_ srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv _count, dst_host_same_srv_ rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_ rerror_rate, dst_host_srv_rerror_rate*; the symbolic features include *protocol_type, service, flag, land, logged_in, is_host_login, is_guest_login.*

Table 1: The attack types

Attack types	Instances	Class
Dos attack	229853	1
U2R attack	70	2
R2L attack	16347	3
Probe attack	4166	4

Table 2 listed the results of NNTrees constructed on different number of input features and other compared methods averaged on 8 runs. The number of critical points for each continuous feature is $C = 4$. We considered four items in the experiments: recognition rate on training set(R_{train}), recognition rate on test set(R_{test}), size of the NNTrees(T_{size}) and average number of hidden neurons of each ENN(N_{H}) or support vectors for SVM.. The results listed in the table showed that as the number of inputs grew from 2 to 8, the average number of hidden neurons for the ENNs decreased from 4.0 to 3.6, and the average size of those NNTrees behaved a mild variation. While 8 inputs were select for each ENN, the number of hidden neurons deccreased from 4.0 to 3.6 and the average size of NNTrees decreased from 13.1 to 8.5. Both the training recognition rate and the test one listed in Tab. 3 were satisfying. It suggests that when proper number of inputs is selected, one can construct an NNTree with small tree size, simple ENN as well as high recognition rate. It indicates that a relative ordinary decision surface can be achieved and thus comparatively simple Boolean function could perform the interpretation for each node.

In this work, we also compared the performance of intrusion detection mode based on understandable NNTree with the one based on a three layer BP neural network (BP-NN) and support vector machine (SVM). The BP-NN was set to a given number of hidden neurons compared to the average sum of the hidden neurons used in the whole NNTree. It suggested NNTree based intrusion detection approach reached better detection accuracy than both SVM and BP-NN. Further, when turned to the interpretation step, a more complex Boolean function seemed to be required to interpret the complex decision surface produced by an SVM with large number of support vectors or a neural network with a large number of hidden neurons. While taking an NNTree into account, owed to its fine topology, one could firstly interpret each node with a simple Boolean function and then combined all the nodes to get an understandable result, and that the NP-complete problem was also avoided.

Table 2: The compared detection rate of intrusion detection approach based on NNTrees

Method	n_{input}	R_{train} (%)	R_{test} (%)	T_{size}	N_{H}
NNTree	2	98.58	97.69	13.1	4.0
	4	98.84	97.77	10.0	3.9
	6	98.86	98.01	10.6	3.6
	8	98.85	98.15	8.5	3.6
SVM	41	98.08	96.32	--	78.2
BP-NN	41	80.53	78.36	--	40.0

One example of NNTrees constructed on the training set is given in Fig.1, where C_1 and C_2 is the class distribution of training data and testing data, respectively. $L = 0$ and 1 denote those normal instances and attacked ones, respectively. For instance, the training sample distribution at node E_1 was 209 normal instances and 791 attacked ones, and the testing sample distribution was 807 normal instances and 3193 attacked ones. The features (F_{select}) selected by the evolution process and the number of hidden neurons N_{H} of each ENN is listed in Table 3. In the example, the number of inputs for each ENN and critical points for all continuous features was both empirically set to 4. It has been proved efficient enough by the statistical result described in the latter experiments. The example showed that at most condition the number of hidden neurons for each non-terminal node was less than 5. It suggests that all the ENNs embedded in the DT are quite simple but really efficient, which means a relative simple decision surface can be achieved at each node and thus a simple Boolean function may be competent for interpretation.

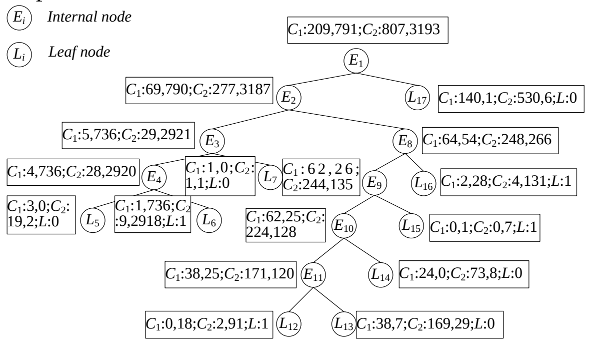
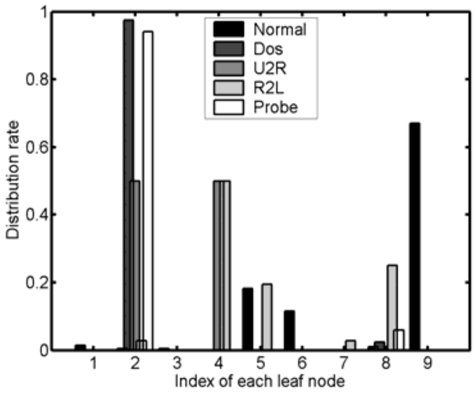

E_i Internal node
L_i Leaf node
C_1:209,791;C_2:807,3193
E_1
C_1:69,790;C_2:277,3187
E_2
L_{17}
C_1:140,1;C_2:530,6;L:0
C_1:5,736;C_2:29,2921
E_3
E_8
C_1:64,54;C_2:248,266
C_1:4,736;C_2:28,2920
E_4
C_1:1,0;C_2:1,1;L:0
L_7
C_1:62,26;C_2:244,135
E_9
L_{16}
C_1:2,28;C_2:4,131;L:1
C_1:3,0;C_2:19,2;L:0
L_5
C_1:1,736;C_2:9,2918;L:1
L_6
C_1:62,25;C_2:224,128
E_{10}
L_{15}
C_1:0,1;C_2:0,7;L:1
C_1:38,25;C_2:171,120
E_{11}
L_{14}
C_1:24,0;C_2:73,8;L:0
C_1:0,18;C_2:2,91;L:1
L_{12}
L_{13}
C_1:38,7;C_2:169,29;L:0

Fig. 1 An example of NNTrees constructed on the database

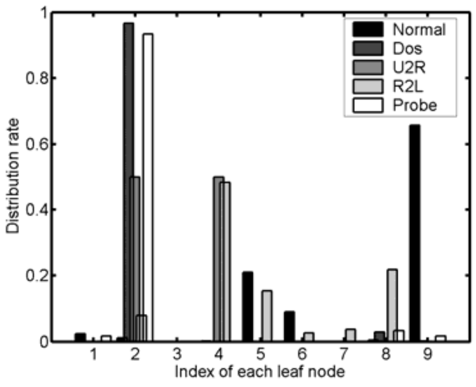
Table 3: Features selected and number of hidden neurons of each ENN

Node	F_{select}	N_H
E_1	*service, hot, num_shells, serror_rate*	4
E_2	*flag, srv_count, srv_serror_rate, dst_host_same_src_port_ rate*	5
E_3	*hot, su_attempted, count, diff_srv_rate*	4
E_4	*is_guest_login, same_srv_rate, dst_host_same_src_port_ rate, dst_host_srv_serror_rate*	2
E_8	*num_failed_logins, is_host_login, is_guest_login, dst_host_srv_serror_rate*	3
E_9	*protocol_type, logged_in, num_outbound_cmds, dst_host_srv_count*	3
E_{10}	*root_shell, su_attempted, srv_diff_host _rate, dst_host_srv_count*	5
E_{11}	*src_bytes, num_shells, count, dst_host_srv_rerror_rate*	5

The training distribution rate of each attack at each leaf node is shown in Fig.2(a), where the x-label $\{1,2,\cdots,8,9\}$ corresponds to those leaf nodes $L_{i\{i=5,6,7,12,13,14,15,16,17\}}$ in Fig.1. As the training result showed that the distribution rates of the Dos and Probe attack instances were much higher at L_6 than any other remained leaf nodes. It suggests that features used at E_4 , i.e., *is_guest_login, same_srv_rate, dst_host_same_src_port_ rate, dst_host_srv_serror_rate* have much higher authority in recognizing those 2 types of attacks. Analogically, features selected at E_4 and E_{11} are efficient for recognition of the U2R attack instances; features selected at E_8 and E_{11} are efficient for detection of R2L attack instances. The Dos and Probe attack instances are the two most legible attack types since more than 96% of these instances were recognized at the first leaf node L_6. While the R2L attack instances were difficult to recognize and time-consuming, since it needed more features including the ones selected at E_4, E_8, and E_{11}. The distribution rate of each attack type at each leaf node for test was shown in Fig.2 (b). The result was quite close to that of training set, which indicated that the generalization ability of proposed mode was satisfying.



(a) The distribution rate of instances for traning



(b) Theditribution rate of instances for test

Fig. 2 The histogram of distribution of the attack and normal instances at each leaf node of the trained NNTree for training and test,respectively.

4. Conclusions

In this paper, we proposed a novel intrusion detection mode based on understandable NNTrees. Experimental results showed that the NNTrees built from the self-organized feature learned training data were satisfying for the high recognition rate and a better generalization ability.

Furthermore, the learned model contains the information about those features of critical importance for detection. One can accordingly focus his observation on those important attributes and hence decrease the complexity of the feature space. Besides, the learning results may also enlighten researchers on exploring new features for those difficult detecting attack types.

We also observed that the number of critical points in each dimension is usually small for the datasets we used. This means that a few (e.g. 3 or 4) binary bits are enough to represent each continuous input. The number of inputs for each ENN is also limited and 8 inputs are proved to be efficient enough to construct an NNTree with both small tree size and hidden neuron number in the experiment, and thus, the NNTrees so obtained are much more interpretable.

We have tried to design an intrusion detection mode based on both interpretable and comprehensible NNTrees. The design process itself, however, may be time consuming because GA is used. Currently, we are trying to propose more efficient and effective algorithms for designing the NNTrees. Another point is that in this paper we have not interpreted the NNTrees explicitly. We will complete this part in the next step, and compare the extracted rules (Boolean functions) with those obtained using other methods. As for the quantization of the dataset, we may use some algorithms (say the R^4-rule) for finding the critical points automatically. We will study this topic in the future.

Acknowledgments

This work was partly supported by the Project of NSFC(Grant 60672095), National High Technology Project (Grant 2003AA143040) of China, a Foundation of Excellent Doctoral Dissertation of Southeast University (YBJJ0412) and the Project of NSFJS (Grant BK2001042).

References

[1] E. Biermann, E.Cloete, L.M. Venter, A comparison of Intrusion detection systems, *Computers and* Security, 20(2001)8, 676–683.

[2] Cho Sung-Bae, Incorporating soft computing techniques into a probabilistic intrusion detection system, *IEEE Trans. on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 32(2002)2, 154–160.

[3] Cho Sung-Bae, Park Hyuk-Jang, Efficient anomaly detection by modeling privilege flows using hidden Markov model, *Computers and Security*, 22(2003)1, 45–55.

[4] Sang-Jun Han, Sung-Bae Cho, Evolutionary Neural Networks for Anomaly Detection Based on the Behavior of a Program, *IEEE Transactions on Systems, Man and Cybernetics—Part B*, 36(2006)3, 559–570

[5] Richard P. Lippmann, Robert K. Cunningham, Improving intrusion detection performance using keyword selection and neural networks, *Computer Networks*, 34(2000)4, 597–603.

[6] Boukerche Azzedine, Jucá Kathia Regina Lemos, Sobral João Bosco, Notare Mirela Sechi Moretti Annoni, An artificial immune based intrusion detection model for computer and telecommunication systems, *Parallel Computing*, 30(2004)5/6, 629–646.

[7] David A. Nash, Daniel J. Ragsdale, Simulation of self-similarity in network utilization patterns as a precursor to automated testing of intrusion detection systems, *IEEE Trans. on Systems, Man, and Cybernetics—Part A: System and Humans*, 31(2001)4, 327–331.

[8] Hiroshi Tsukimoto, Extracting Rules from trained neural networks, *IEEE Trans. on Neural Networks*, 11(2000)2, 377–389.

[9] Q.Z. Xu, Q.F. Zhao, W.J. Pei, L.X. Yang, Z.Y. He, Interpretable Neural Network Tree for Continuous-Feature Data Sets, *Neural Information and Processing—letters and reviews*, 3(2004)3, 77–84.

[10] H. Guo, S. B. Gelfand, Classification trees with neural network feature extraction, *IEEE Trans. on Neural Networks*, 3(1992)6, 923–933.

[11] T. Takeda, Q.F. Zhao, Growing Neural Network Trees Efficiently and Effectively, Proc. International Conference on Hybrid Intelligent Systems (HIS'03), Melbourne, 2003, 107–115.

[12] S. Mizuno, Q.F. Zhao, Neural network trees with nodes of limited inputs are good for learning and understanding, Proc. 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL2002), Singapore, 2002, 573–576.

[13] J. R. Quinlan, C4.5: Programs for Machine Learning, San Mateo, Morgan Kaufmann Publishers, 1993, 17–25.

[14] C. Lu, Q. F. Zhao, W. J. Pei, Z. Y. He, A multiple objective optimization based GA for designing interpretable and comprehensible neural network trees, Proc. IEEE International Conference on Neural Networks & Signal Processing, China, 2003, 518–521.

[15] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Boston, Addison-Wesley, 1989, 1–217.

[16] T. Kohonen, The self-organizing map, *Proceedings of the IEEE*, 78(1990)9, 1464–1480.

Qinzhen Xu QinZhen Xu received the M.S. degree from Biomedical Engineering Department of Southeast University in 2000, and currently a Ph.D. student of Radio Engineering Department of Southeast University, China. Her research interests include hybrid learning model, infromation security and image processing.