International Journal of Soft Computing 3 (4): 277-281, 2008
ISSN: 1816-9503

Application of Back-Propagation Algorithm in Intrusion Detection in Computer Networks

[1]Meera Gandhi and [2]S.K. Srivatsa
[1]CSE, Sathyabama University, Chennai, India
[2]St.Joseph's College of Engineering, Chennai, India

Abstract: Computer systems are prone to attacks by incoming malicious packets which are having information against the Request for Comments (RFC) standards. When the fields of the packets have information that is not part of the standards, then the packets are named as intruders. How to detect all possible intrusion packet in addition to other form of intrusion based on behavior of system, is a challenging task even for the leading Operating System (OS) manufacturer. In spite of existing conventional technologies, artificial neural networks have been explored for intrusion detection with little amount of research. In this research, supervised Artificial Neural Network (ANN) trained by the Back-Propagation Algorithm (BPA) has been implemented with varying values of learning factor, η. The implemented system will become foolproof if the ANN is trained with all possible intrusion packet types. This study explains a better way of training the ANN for achieving more than 98% of intrusion detection when minimum number of intrusion packets is given during training ANN. Our experimental results show the performance of intrusion packets detection using back propagation algorithm. Thousand packet information of both normal and intrusion have been considered for implementation. The result of Intrusion Detection (ID) is very close to 99%. The topology of the ANN is (41×10×1). The network converged with 550 iterations. However, very huge amount of packets are to be evaluated to know the complete performance of the developed system.

Key words: Intrusion detection, neural network, topology, back-propagation algorithm

INTRODUCTION

Due to the complexity of distributed systems and information resources increasing, computer networks are exposed to computer crime. Many modern systems lack properly implemented security services. As network attacks have increased in number over the past few years, the efficiency of security systems have declined.

The rapid and extensive growth of internet technology increases the importance of protecting computer networks from attacks. In the last years, the number of network attacks has been raised very promptly that has led to significant problems due to DOS (denial of service).

Intrusion Detection Systems (IDS) are used as a computer network security tool and permit to alert an administrator in case of attack. Nowadays, there exist different approaches for intrusion detection. They are signature analysis, rule-based method, embedded sensors, neural networks, artificial immune systems (Bershad *et al*., 1993, a, b; Graham, 2000; Bace and Mell, 2004; Gordeev, 2004). The most of these IDS can detect the known attacks and have poor ability to detect new attacks. Neural network techniques have been applied and investigated for intrusion detection (Lippmann, 1987; Hiros *et al*., 1991; Kazienko and Doross, 2003; Jean-Philippe, 2004). Such approaches are based on different strategies; one is anomaly detection that uses analysis of the audit records, produced by the operating system. The other one is based on network protocol analysis.

Among the neural networks, feed forward networks, namely Multilayer Perceptrons (MLP) is very much used for different applications. This network type has been proven to be universal function approximator. The important feature of MLP is the ability to generalize. Therefore, MLP can be powerful tool for design of intrusion detection systems. Neural networks can suffice in identifying intrusions.

ARTIFICIAL NEURAL NETWORK (ANN)

An Artificial Neural Network (ANN) is a mathematical way of simulating the capability of human brain. The category of supervised method requires inputs and target outputs. The Back-Propagation Algorithm (BPA) is a supervised method that uses steepest-descent method to reach global minima. The multilayer perceptron is shown in Fig. 1. The flowchart for the BPA is given in Fig. 2. The

Corresponding Author: Meera Gandhi, CSE, Research Scholar, Sathyabama University, Chennai, India

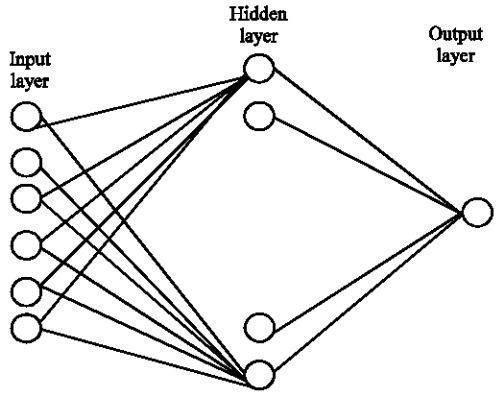

Fig. 1: Back propagation network

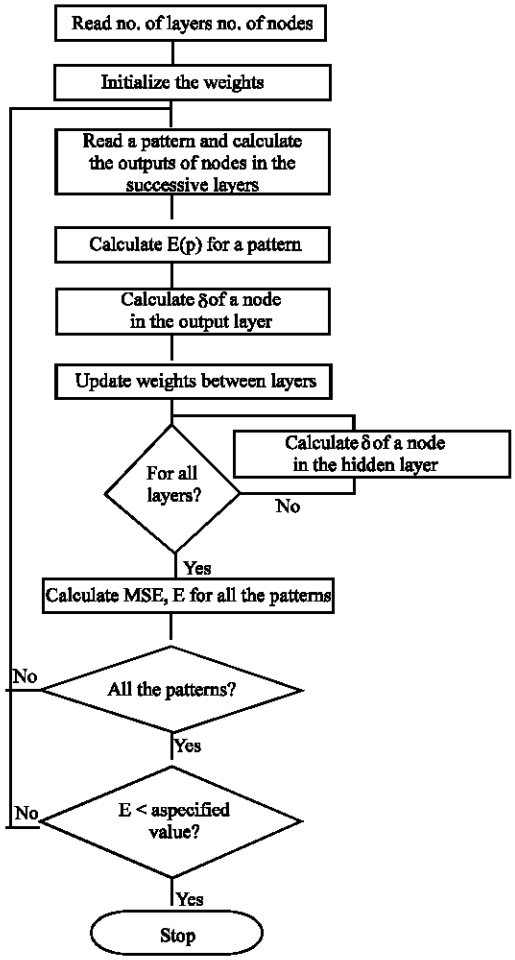

Fig. 2: Flow chart for BPA

number of layers and number of nodes in each layer is decided. The connections between nodes are initialized with random weights. A pattern from the training set is presented in the input layer of the network and the error is calculated in the output layer. The error is propagated backwards towards the input layer and the weights are updated. This procedure is repeated for all the training patterns. At the end of iteration, test patterns are presented to ANN and the classification performance of ANN is evaluated. Further training of ANN is continued till the desired classification performance is reached.

Steps involved

Forward propagation: The output of each node in the successive layers is calculated.

$$o \text{ (output of a node)} = 1/(1+\exp(-\Sigma w_{ij} x_i)) \quad (1)$$

Where:

w_{ij} : The weight matrix connecting nodes of the previous layer i with nodes of next layer j.

x_i : The variables of a pattern.

o : The output of a node in the successive layer.

The error E (p) of a pattern number p is calculated

$$E(p) = (1/2)\ \Sigma(d(p) - o(p))^2 \quad (2)$$

Where,

d - ---target value of a pattern

Reverse propagation: The error δ for the nodes in the output layer is calculated

$$\delta(\text{output layer}) = o(1-o)(d-o) \quad (3)$$

The new weights between output layer and hidden layer are updated

$$w(n+1) = w(n) + \eta\ \delta(\text{output layer})\ o(\text{hidden layer}) \quad (4)$$

Where: η is the learning factor and $(0<\eta<1)$

The error δ for the nodes in the hidden layer is calculated

$$\delta(\text{Hidden layer}) = o\ (1-o)\ \Sigma\delta(\text{output layer})\ w\ (\text{updated weights between hidden and output layer}) \quad (5)$$

The weights between hidden and input layer are updated.

$$w(n+1) = w(n) + \eta\ \delta(\text{hidden layer})\ o(\text{input layer}) \quad (6)$$

The above steps complete one weight updation. The remaining training patterns are presented and Eq. 1-6 are followed which form one iteration. The training of the network is stopped once the desired Mean Squared Error (MSE) is reached as given below

$$E(MSE) = \Sigma E(p) \quad (7)$$

The final updated weights are saved for detecting the intrusion.

NETWORK INTRUSION DETECTION SYSTEM (NIDS) USING BPN

Packet capture: Packet capture is the beginning process in NIDS. It can be implemented by setting the working mode of the network card as the promiscuous mode. The network card under common mode can only receive the packet whose destination address is the network card itself. Only those packets are not sufficient to serve for the data source of the NIDS. So, it is necessary to set the network card's working mode as the promiscuous mode. Under this mode, the network card can receive not only the packets sent to itself but also the packets routed to some other hosts. Thus the NIDS can monitor the network stream of all hosts of some local area network and detect whether intrusion happens or not.

Feature extraction: Feature selection and extraction is one of the pivotal problems in implementing the intrusion detection system. Network stream itself is not suitable directly as the input of the Back-Propagation (BP) classifier, so it is necessary to extract representative features from the network stream. The features extracted from the network stream (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html) form a feature vector that serves for the description of the packet. Whether the feature vector can describe the network stream correctly and efficiently or not has a large effect on the efficiency and correctness of the NIDS. Selecting several features such as the protocol code, the packet head length, the checksum, the port number and some TCP Flags, etc have been done. Based on these features, a vector is obtained as follows to describe an intrusion. The following representation is some of the intrusion types which contain some sequence of intrusion appearance

Attack (type) = (P-id, H-Len, C-sum ,S-port, D-port, ICMP-type, ICMP-Code, Flag, P-Len, P-data)

Above is the general description form of an abstract attack.
Perhaps some concrete examples can explain the vector well.

Attack (CGI) = (Tcp,32, O, 2345, 80, null, null, A, 421, get CGI-bin)
Attack (FTP) = (TCP, 24, 16, 21, 21, null, null, PA, 256, ROOM)
Attack (Redirect) = (ICMP, 20, null, null, 8, 3, null, 192, la)
Attack (UDP) = (UDP, 16, 10, 138, 126, null, null, nuli,448,3c)

If the features of a packet are found as any of the above, it represents a CGI attack; a FTP attack, a REDIRECT attack and a UDP attack (Sundaram, 2001; Verwoerd and Hunt, 2001), respectively. The feature vector will serve for the input of the BP Classifier., then the BP Classifier will judge whether the feature vector represents an intrusion or not.

EXPERIMENTAL SETUP

The simulation results were obtained from the standard Knowledge Discovery Dataset (KDD) data set (Gunes *et al*., 1999; Stevern *et al*., 2006; Ajith *et al*., 2007). It is a well defined as normal and with different types of attack for TCP, UDP, ICMP, etc. A set of sample data set is shown in Table 1. Each row is a pattern. The fields in each pattern describe the properties of respective packet. The various attacks considered during training are

back dos
buffer_overflow u2r
ftp_write r2l
guess_passwd r2l
imap r2l
ipsweep probe
land dos
loadmodule u2r
multihop r2l
neptune dos
nmap probe
perl u2r
phf r2l
pod dos
portsweep probe
rootkit u2r
satan probe
smurf dos
spy r2l
teardrop dos
warezclient r2l
warezmaster r2l

Instead of KDD data set, free Sniffer soft wares like network sniffer, packet sniffer and more soft wares can be

Table 1: Sample KDD dataset

S.No	Packet details
1	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00, 0.00, 1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
2	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00, 0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
3	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00, 0.00, 1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
4	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00, 0.00,0.00, 1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
5	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00, 1.00, 0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
6	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00, 1.00,0.00,0.00,255,255,1.00,0.00,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
7	0,udp,domain_u,SF,29,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,0.00,0.00,0.00, 0.00,0.50,1.00,0.00,10,3,0.30,0.30,0.30,0.00,0.00,0.00,0.00,0.00,normal.
8	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00, 0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal
9	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00, 1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
10	0,tcp,http, SF,223,185,0,0,0,0,1,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00, 1.00,0.00,0.00,71,255,1.00,0.00,0.01,0.01,0.00,0.00,0.00,0.00, normal.

Table 2 Sample dataset used for training

S.No	Patterns used for training Input to ANN	Target outputs
1	0 .2 .01 .1 105 146 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0.00 0.00 0.00	
	0.00 1. 00 0.00 0.00 255 254 1.00 0.01 0.00 0.00 0.00 0.00 0.00 0.00	0.1
2	0 .2 .01 .1 105 146 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0.00 0.00 0.00	
	0.00 1.00 0.00 0.00 255 254 1.00 0.01 0.00 0.00 0.00 0.00 0.00 0.00	0.1
3	0 .2 .01 .1 105 146 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0.00 0.00 0.00	
	0.00 1.00 0.00 0.00 255 254 1.00 0.01 0.00 0.00 0.00 0.00 0.00 0.00	0.1
4	0 .2 .01 .1 105 146 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0.00 0.00 0.00	
	0.00 1.00 0.00 0.00 255 254 1.00 0.01 0.00 0.00 0.00 0.00 0.00 0.00	0.2
5	0 .2 .01 .1 105 146 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0.00 0.00 0.00	
	0.00 1.00 0.00 0.00 255 254 1.00 0.01 0.01 0.00 0.00 0.00 0.00 0.00	0.2

used to extract the values of a packet, which can be further labeled as normal or an attack to be used for training (Michalski *et al*., 2006). The contents of the packet should be suitably modified into meaningful numerical values. A sample dataset used for training is shown in Table 2.

RESULTS AND DISCUSSION

The topology of ANN used is 41×10×1; no. of nodes in the input layer is 41, no. of nodes in the hidden layer is 10 and no. of nodes in the output layer is 1. The labeling is set as 0.1 (Normal) or 0.2 (attack). It is mandatory to use huge amount of patterns to be presented for training ANN. However, it would take enormous amount of time for the ANN to learn the patterns. Hence, only 1000 patterns have been considered for training purpose. The dataset has been separated as training and testing (intrusion detection). Training indicates the formation of final weights, which indicate a thorough learning of intrusion and normal packets along with corresponding labeling.

Figure 3 shows the convergence curve for the topology of 10 nodes in the hidden layer. The network

Table 3: Distribution of patterns chosen for training

Packet type	Total numbers used for training
Normal	363
Intrusion	637

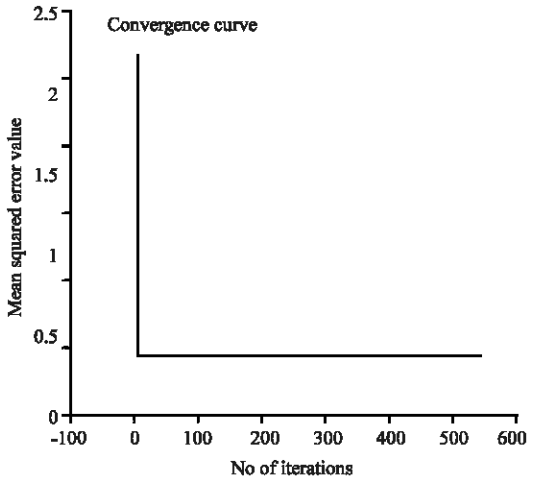

Fig. 3: Mean squared error curve

was trained to reach MSE of 0.001. It converged in 550 iterations. If the number of patterns is further increased

Table 4: Classification performance

Packet type	Total number tested	No. Classified	No. Misclassified
Normal	363	360	3
Intrusion	637	600	37

Table 5: False acceptance/rejection rate

Packet type	False Acceptance Rate (FAR)	False Rejection Rate (FRR)
Normal	5.8% (37/637)	0.8% (3/360)
Intrusion	10.1%(37/363)	0.4%(3/637)

from 1000, then the convergence iterations will increase. In addition, the convergence depends on the orthogonolity of data presented for training. If the subsequent patterns are not orthogonal, the convergence would take a long time or it may not converge. Table 3 gives distribution of patterns chosen for training and testing. The classification performance is given in Table 4. In Table 5, False Acceptance Rate (FAR) and False Rejection Rate (FRR) are presented.

CONCLUSION

In this research, KDD dataset has been considered to experiment the performance of BPA in classifying the LAN intrusion packets. A topology of 41×10×1 had been chosen. Instead of 10 nodes in the hidden layer, different number of nodes could have been chosen. That would give different performance of classification rate.

REFERENCES

Ajith Abraham, Ravi Jain, Johnson Thomas and Sang Yong Han, 2007. D-SCIDS: Distributed soft computing intrusion detection system. J. Network and Comput. Applications, 30: 81-98.

Bace, R. and P. Mell, 2004. NIST Special Publication on Intrusion Detection Systems, http://www.nist.gov.

Bace, R., 2002. Intrusion Detection, Macmillan Technical Publishing.

Bershad, N.J. *et al*., 1993. Statistical analysis of the single layer back-propagation algorithm: Part-II-NMSE and classification performance. IEEE. Trans. Acoustics, Speech and Signal Proc. 41 (2): 583-591.

Bershad, N.J. *et al*., 1993. Statistical analysis of the single-layer-back-propagation algorithm: Part-I-Mean weight behavior. IEEE. Trans. Acoustics, Speech and Signal Proc., 41 (2): 573-582.

Gordeev, M., 2004. Intrusion Detection Techniques and Approaches, http://www.ict.tuwein.ac.a.

Graham, R., 2000. FAQ: Network Intrusion Detection Systems, http://www.robertgraham.com.

Güneþ, H. and A. Kayacýk, 1999. Nur Zincir-Heywood, Malcolm I. Heywood, Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets Dalhousie University, Faculty of Computer Science, 6050 University Avenue, Halifax, Nova Scotia. B3H 1W5.

Hirose, Y. *et al*., 1991. Back-propagation algorithm which varies the number of hidden units. Neural Networks, 4 (1): 61-66.

Jean-philippe, 2004. Application of Neural Networks to Intrusion Detection, http://www.sans.org.

Kazienko, P. and P. Dorosz, 2003. Intrusion Detection Systems (IDS) Part I (network intrusions; attack symptoms; IDS tasks; and IDS.

Lippmann, R.P., 1987. AN Introduction to computing with neural Nets. IEEE. Trans. ASSP Mag. 35, 4(2): 4-22.

Michalski, R.S., K.A. Kaufman, J. Pietrzykowski, B. Sniezynski and J. Wojtusiak, 2006. Intelligent Information Systems, New Trends in Intelligent Information Processing and Web Mining, Ustron, Poland.

Steven Cheung, Bruno Dutertre, Martin Fong, Ulf Lindqvist, Keith Skinner and Alfonso Valdes, 2006. Using Model-based Intrusion Detection for SCADA Networks, Computer Science Laboratory, SRI International.

Sundaram, A., 2001. An Introduction to Intrusion Detection, http://www.acm.org.

Verwoerd, T. and R. Hunt, 2001. Intrusion Detection Techniques and Approaches, http://www.elsevier.com.