IMPLEMENTATION OF ECHOSTATE NETWORK IN NIDS

[1]Meera Gandhi,
Research scholar,
Department of Computer science and engg.
Sathyabama University,
Sholinganallur ,
Chennai-600100 meera.gandhi@gmail.com

[2]S.K.Srivatsa,
Professor,
ICE, St.Joseph's College of Engg.,
Chennai – 600119 profsks@hotmail.com

Abstract

Identifying instances of network attacks by comparing current activity against the expected actions of an intruder has become an important. Most current approaches to misuse detection involve the use of rule-based expert systems to identify indications of known attacks. Artificial neural networks provide the potential to identify and classify network activity based on limited, incomplete, and nonlinear data sources. Transmission of data over the internet keeps on increasing, which needs to protect connected systems also increasing. Intrusion Detection Systems (IDSs) are the latest technology used for this purpose. Although the field of IDSs is still developing, the systems that do exist are still not complete, in the sense that they are not able to detect all types of intrusions. Some attacks which are detected by various tools available today cannot be detected by other products, depending on the types and methods that they are built on. In this work, an artificial neural network using echo state network algorithm has been used to implement the IDS. This paper proposes an approach to implement recurrent echo state network real time IDS. Twenty four packet information both normal and intrusion have been considered for training. Testing has been done with new sets of packet information. The result of intrusion detection (ID) is very close to 90%. The topology of the echo state network is (41 X 20 X 1). The network converged with 24 iterations. However, very huge amount of packets are to be evaluated to know the complete performance of the developed system.

Keywords: Echo state network, Intrusion detection, misuse detection, neural networks, computer security

1. INTRODUCTION

The complexity, as well as the importance, of distributed computer systems and information resources is rapidly growing. Due to this, computers and computer networks are often exposed to computer crime. Many modern systems lack properly implemented security services; they contain

a variety of vulnerabilities and, therefore, can be compromised easily. As network attacks have increased in number over the past few years, the efficiency of security systems such as firewalls have declined.

It is very important that the security mechanisms of a system are designed to prevent unauthorized access to system resources and data. Building a complete secure system is impossible and the least that can be done is to detect the intrusion attempts so that action can be taken to repair the damage later. Organizations are increasingly implementing various systems that monitor IT security breaches. Intrusion detection systems (IDSs) have gained a considerable amount of interest within this area. The main task of IDS is to detect an intrusion and, if necessary or possible, to undertake some measures eliminating the intrusions.

Because most computer systems are vulnerable to attack, intrusion detection (ID) is a rapidly developing field. Intrusion Detection Systems (IDSs) detect intrusions using specific methodologies that are specific to each of them. A method describes how an IDS analyzes data to detect possible intrusions, based on the analysis approaches. The analysis approaches are anomaly detection and misuse detection. There are many methods that are used. Examples of them include statistical approaches, protocol anomaly detection, neural networks [4-6][14][18][20][24] , file checking, expert systems , rule-based measures[19], and genetic algorithms (GAs) [26][35].

2. BACKGROUND

Intruders tend to find new ways to compromise systems each day. As more intrusions occur, the weaknesses of existing technologies like firewalls are exposed. Since it is impossible to build a complete secure system, IDSs are used to detect the intrusions that occur. This is why IDSs are gaining acceptance in every organization. To understand what an IDS is, first one should know what intrusion and intruders are. Intrusion is the unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable. To detect intrusions and to prevent them, one has to be aware of how an intruder can cause intrusions.

The primary ways an intruder can get into the system is through primary intrusion, system intrusion and remote intrusion. ID is the process of monitoring the events occurring in a computer system or network, and analyzing them for intrusions. The prevention of intrusions should be done through effective IDSs. An IDS is a software or hardware product that automates this monitoring and analysis process.

The types of IDSs can be described in terms of three fundamental functional components. They are the information source, analysis and response. The information source of the system mainly depends on where the IDSs are being placed, hence it is also known as the monitoring locations of the IDS. The information sources are mainly of three types: network-based IDSs, host-based IDSs and application-based IDSs [Bace, 2002]. Since the focus of this work is on network-based IDS, the other two types will not be considered here. Network-based IDSs detect attacks by capturing and analyzing network packets.

They search for attack signatures within the packets. Signatures might be based on actual packet contents, and are checked by comparing bits to known patterns of attack. If the bits are matched to known patterns of attack, then an intrusion is triggered. Once the information sources have monitored network traffic, the next step is to analyze the events to detect the intrusion. The two main techniques or approaches used to analyze events to detect attacks are misuse detection and anomaly detection. Response is the set of actions that the system takes once it detects intrusions. Some of the responses [11-13] involve reporting results and findings to a pre-specified location, while others are more actively automated responses.

Commercial IDSs support both active and passive responses, and sometimes a combination of the two. IDSs can be viewed as the second layer of protection against unauthorized access to networked information systems because despite the best access control systems, intruders are

still able to enter computer networks. IDSs expand the security provided by the access control systems by providing system administrators with a warning of the intrusion.

They also provide the system administrators with necessary information about the intrusions. This assists the system administrators in controlling the intrusions that has occurred, in order to avoid them in the future or to minimize the damage that may occur due to an intrusion [15-17]. Although IDSs can be designed to verify the proper operation of access control systems by looking for the attacks that get past the access control systems, IDSs are more useful when they can detect intrusions that use methods that are different from those used by the access control systems. For this purpose, they must use more general and more powerful methods than simple database look-ups of known attack scenarios.

2.1. Signature basics

A network IDS signature is a pattern that we want to look for in traffic. Some of the methods that can be used to identify each one:

- Connection attempt from a reserved IP address. This is easily identified by checking the source address field in an IP header.

- Packet with an illegal TCP flag combination. This can be found by comparing the flags set in a TCP header against known good or bad flag combinations.

- Email containing a particular virus. The IDS can compare the subject of each email to the subject associated with the virus-laden email, or it can look for an attachment with a particular name.

- DNS buffer overflow attempt contained in the payload of a query. By parsing the DNS fields and checking the length of each of them, the IDS can identify an attempt to perform a buffer overflow using a DNS field. A different method would be to look for exploit shell code sequences in the payload.

- Denial of service attack on a POP3 server caused by issuing the same command thousands of times. One signature for this attack would be to keep track of how many times the command is issued and to alert when that number exceeds a certain threshold [21].

- File access attack on an FTP server by issuing file and directory commands to it without first logging in. A state-tracking signature could be developed which would monitor FTP traffic for a successful login and would alert if certain commands were issued before the user had authenticated properly [22-23].

2.1.1 Purpose of Signatures

Different signatures have different goals. The obvious answer is that, want to be alerted when an intrusion attempt occurs. Why we might want to write or modify a signature. Perhaps seeing some odd traffic on network and want to be alerted the next time it occurs. It has been noticed that it has unusual header characteristics, and want to write a signature that will match this known pattern. Perhaps are interested in configuring IDS to identify abnormal or suspicious traffic in general, not just attacks or probes. Some signatures may tell which specific attack is occurring or what vulnerability the attacker is trying to exploit, while other signatures may just indicate that unusual behavior is occurring, without specifying a particular attack. It will often take significantly more time and resources to identify the tool that's causing malicious activity, but it will give more information as to why being attacked and what the intent of the attack is.

2.1.2 Header Values

Simple signature characteristic header values are presented. Some header values are clearly abnormal, so they make great candidates for signatures. A classic example of this is a TCP packet with the SYN and FIN flags set. This is a violation of request for comments (RFC 793) (which defines the TCP standard), and has been used in many tools in an attempt to circumvent firewalls, routers and intrusion detection systems. Many exploits include header values that purposely violate RFCs, because many operating systems and applications have been written on the assumption that the RFCs would not be violated and don't perform proper error handling of such traffic.

Many tools either contain coding mistakes or are incomplete, so that crafted packets produced by them contain header values that violate RFCs. Both poorly written tools and various intrusion techniques provide distinguishing characteristics that can be used for signature purposes [25]. There's a catch. Not all operating system (OS) and applications completely adhere to the RFCs. In fact, many have at least one facet of their behavior that violates an RFC. Over time, protocols may implement new features that are not included in an RFC.

New standards emerge over time, which may "legalize" values that were previously illegal; RFC 3168, for Explicit Congestion Notification (ECN), is a good example of this. So an IDS signature based strictly on an RFC may produce many false positives. Still, the RFCs make a great basis for signature development, because so much malicious activity violates RFCs. Because of RFC updates and other factors, it's important to review and update existing signatures periodically [27] [29].

2.1.3 Sample signature

- Various source IP addresses
- TCP source port 21, destination port 21
- Type of service 0
- IP identification number 39426
- SYN and FIN flags set
- Various sequence numbers set
- Various acknowledgment numbers set
- TCP window size 1028

Packet values that are completely normal don't make good signature characteristics by themselves, although they are often included to limit the amount of traffic that we study. By including the normal IP protocol value of 6 for a protocol, so that only check TCP packets. But other characteristics that are completely normal, such as the type of service set to 0, are much less likely to be helpful in signature development.

A signature based on few suspicious characteristics may be too specific [32-34]. Although it would provide much more precise information about the source of the activity, it would also be far less efficient than a signature that only checks one header value. Signature development is always a tradeoff between efficiency and accuracy. In many cases, simpler signatures are more prone to false positives than more complex signatures, because simpler signatures are much more general [36][39-40]. But more complex signatures may be more prone to false negatives than simpler signatures, because one of the characteristics of a tool or methodology may change over time.

2.1.4. Schematic flow diagram

The sequence of steps required for the IDS by implementing an ESNN has been given in Figure 1

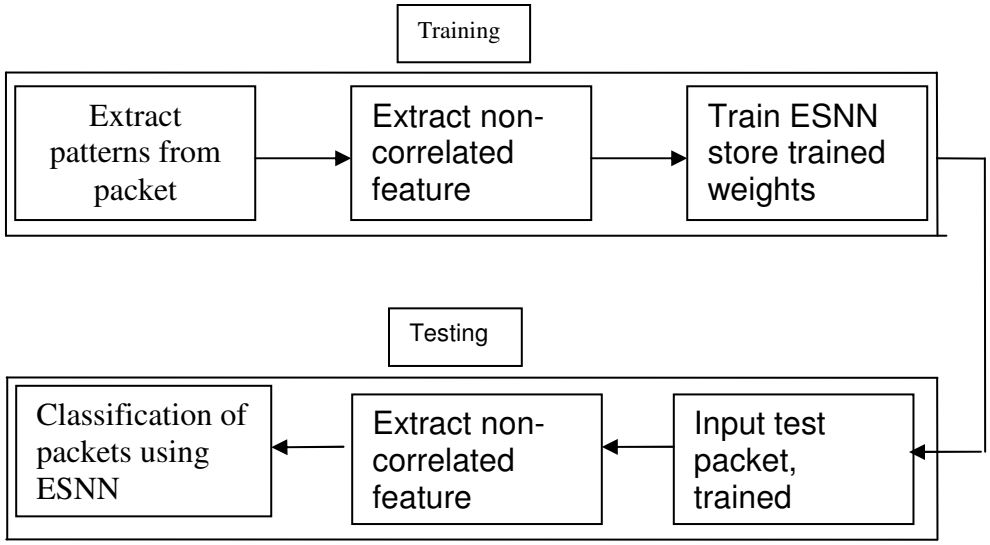

FIGURE.1 Schematic diagram of the IDS

After training the ESNN, a set of final weights are obtained and stored in a file. During the process of testing, a test packet converted into uncorrelated features followed by classification by ESNN using final weight values arrived during training phase.

2.2. Echo State Neural Network (ESNN)

An artificial neural network (ANN) is an abstract simulation of a real nervous system that contains a collection of neuron units, communicating with each other via axon connections. Such a model bears a strong resemblance to axons and dendrites in a nervous system. Due to this self-organizing and adaptive nature, the model offers potentially a new parallel processing paradigm. This model could be more robust and user-friendly than the traditional approaches. ANN can be viewed as computing elements, simulating the structure and function of the biological neural network. These networks are expected to solve the problems, in a manner which is different from conventional mapping. Neural networks are used to mimic the operational details of the human brain in a computer. Neural networks are made of artificial 'neurons', which are actually simplified versions of the natural neurons that occur in the human brain. A neural architecture comprises massively parallel adaptive elements with interconnection networks, which are structured hierarchically.

Artificial neural networks are computing elements which are based on the structure and function of the biological neurons [3]. These networks have nodes or neurons which are described by difference or differential equations. The nodes are interconnected layer-wise or intra-connected among themselves. Each node in the successive layer receives the inner product of synaptic weights with the outputs of the nodes in the previous layer [1], [8-9]. The inner product is called the activation value

Dynamic computational models require the ability to store and access the time history of their inputs and outputs. The most common dynamic neural architecture is the time-delay neural network that couples delay lines with a nonlinear static architecture where all the parameters (weights) are adapted with the back propagation algorithm. Recurrent neural networks (RNNs) implement a different type of embedding that is largely unexplored. One of the main practical problems with RNNs is the difficulty to adapt the system weights. Back propagation through time and real-time recurrent learning; have been proposed to train RNNs. These algorithms suffer from

computational complexity, resulting in slow training, complex performance surfaces, the possibility of instability, and the decay of gradients through the topology and time. The problem of decaying gradients has been addressed with special processing elements (PEs).

ESNN [28],[30-31] possesses a highly interconnected and recurrent topology of nonlinear PEs that constitutes a "reservoir of rich dynamics" and contains information about the history of input and output patterns. The topology of the network is shown in Figure 2. The outputs of internal PEs (echo states) are fed to a memory less but adaptive readout network (generally linear) that produces the network output. The interesting property of ESNN is that only the memory less readout is trained, whereas the recurrent topology has fixed connection weights. This reduces the complexity of RNN training to simple linear regression while preserving a recurrent topology, but obviously places important constraints in the overall architecture that have not yet been fully studied.

The echo state condition is defined in terms of the spectral radius (the largest among the absolute values of the eigen values of a matrix, denoted by (|| . ||) of the reservoir's weight matrix (|| W || < 1). This condition states that the dynamics of the ESNN is uniquely controlled by the input, and the effect of the initial states vanishes. The current design of ESNN parameters relies on the selection of spectral radius. There are many possible weight matrices with the same spectral radius.

ESNN is composed of two parts (Jaeger, H 2002): a fixed weight (|| W || < 1) recurrent network and a linear readout. The recurrent network is a reservoir of highly interconnected dynamical components, states of which are called echo states. The memory less linear readout is trained to produce the output. The recurrent discrete-time neural network is given in with M input units, N internal PEs, and L output units.

The value of the input unit at time n is
$u(n) = [u_1(n), u_2(n), \ldots, u_M(n)]^T$, (1)

The internal units are
$x(n) = [x_1(n), x_2(n), \ldots, x_N(n)]^T$, and (2)

Output units are
$y(n) = [y_1(n), y_2(n), \ldots, y_L(n)]^T$. (3)

The connection weights are given

- in an N x M weight matrix $W^{back} = W_{ij}^{back}$ for connections between the input and the internal PEs,
- in an N × N matrix $W^{in} = W_{ij}^{in}$ for connections between the internal PEs
- in an L × N matrix $W^{out} = W_{ij}^{out}$ for connections from PEs to the output units and
- in an N × L matrix $W^{back} = W_{ij}^{back}$ for the connections that project back from the output to the internal PEs.

Here
M is the no. of neurons in the input layer
N is the no. of neurons in the hidden layer and
L is the no. of neurons in the output layer
The activation of the internal PEs (echo state) is updated by using the relation

$$x(n + 1) = f(W^{in} u(n + 1) + Wx(n) + W^{back} y(n)), \quad (4)$$

where
$f = (f_1, f_2, \ldots, f_N)$ are the internal PEs' activation functions.

All f_i's are hyperbolic tangent function $\frac{e^x - e^{-x}}{e^x + e^{-x}}$. The output from the readout network is computed as follows

$y(n + 1) = f^{out}(W^{out}x(n + 1)),$. (5)

where

$f^{out} = (f_1^{out}, f_2^{out}, ..., f_L^{out})$ are the output unit's of nonlinear functions.

The ESNN topology specified in this work is {41 x no. of reservoirs x 1}, where two nodes are in the input layer, one in the output layer and any number of reservoirs in the hidden layer. The connections between input-hidden layers, hidden-output layer are initialized with random numbers. The training of the ESNN is done with choosing initial random weights in a range of 0.25 to 0.55. The random weights are chosen within a small range for easier quicker settlement of final weights and also to prevent the network from further oscillation.

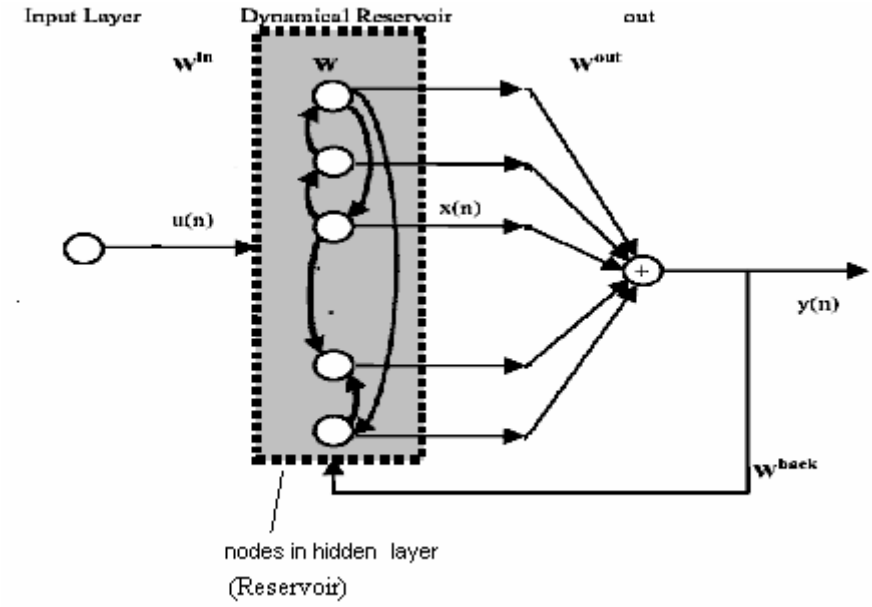

FIGURE.2: An echo state neural network (ESNN)

2.2.1 Implementation of IDS using ESNN

To obtain the final trained weights by training the ESNN

Step 1: Find the uncorrelated features of the packet
Step 2: Fix the target values (class label)
Step 3: Set the no. of inputs, no. of reservoirs, and no. of outputs
Step 4: Initialize connection matrices-using random weights for
no.of reservoirs versus no. of inputs,
no.of outputs versus no. of reservoirs,
no. of reservoirs versus no. of reservoirs
Step 5: Determine values of matrices less than a threshold for updating the weights
Step 6: Normalize the reservoir matrix by finding its eigen value.
Step 7: The initial state matrix is updated with tanh() function. The inputs for the tanh() function are
{input pattern X weights between input and hidden layer **+**
desired output X weights between output and hidden layer +

normalized reservoir matrix}
Step 8: Store the weight matrices.

2.2.2 Implementation of ESNN for IDS using the trained weights of ESNN

Step 1: The trained weights are given as inputs.
Step 2: Apply the uncorrelated features of the packet
Step 3: Process the inputs with the trained weights
Step 4: Employ transfer function to get the output of the ESNN
Step 5: Set threshold and classify intrusion or not.
Overall algorithms for training and testing
1. Obtain the uncorrelated feature of the packet
2. Train the ESNN with the inputs and target outputs. Trained weights are obtained once all the Patterns are presented to the ESNN.
3. Test the ESNN with a new packet, and classify for intrusion.

2.3 Network Intrusion Detection System (NIDS) using ANN

2.3.1 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.

2.3.2 Feature Extraction

Feature selection and extraction [37], [38] is one of the pivotal problems in implementing the intrusion detection system. Network stream itself is not suitable directly as the input for the ESNN, so it is necessary to extract some features from the network stream. The uncorrelated features extracted from the network stream form a feature vector which 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 respectively. The feature vector will serve for the input of the ESNN Classifier., then the ESNN Classifier will judge whether the feature vector represents an intrusion or not.

2.3.3. Experimental Setup

The simulation results were obtained from the standard KDD data set. 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

Table 1. Sample KDD dataset

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

9	0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255, 254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
10	0,tcp,http,SF,223,185,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00,1.00,0.00,0.00,71,255, 1.00,0.00,0.01,0.01,0.00,0.00,0.00,0.00,normal.

Instead of KDD data set, free sniffer software's like network sniffer, packet sniffer and more software's can be used to extract the values of a packet, which can be further labeled as normal or an attack to be used for training. The contents of the packet should be suitably modified into meaningful numerical values. A sample dataset used for training is shown in Table 2.

Table 2 . Sample dataset used for training

S.No	**Patterns used for training** **Input to ESNN after uncorrelating the features of patterns**	**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	.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	.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	.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	.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	.2

2.3.4. Results and Discussion

The topology of ESNN used is 41 X 20 X 1; no. of nodes in the input layer is 41, no. of nodes in the hidden layer is 20 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 ESNN. However, it would take enormous amount of time for the ESNN to learn the patterns. Hence, only 24 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 performance of the ESNN .The x axis represents the packets. It is a combination of accepted packets and intruding packets. The legend '' indicates the desired target used for normal and intrusion. The '○' represents the output of the network. Table 3 gives number of patterns used for training and testing the performance of ESNN in classifying the intrusion packet. Table 4 gives number of patterns classified and misclassified.

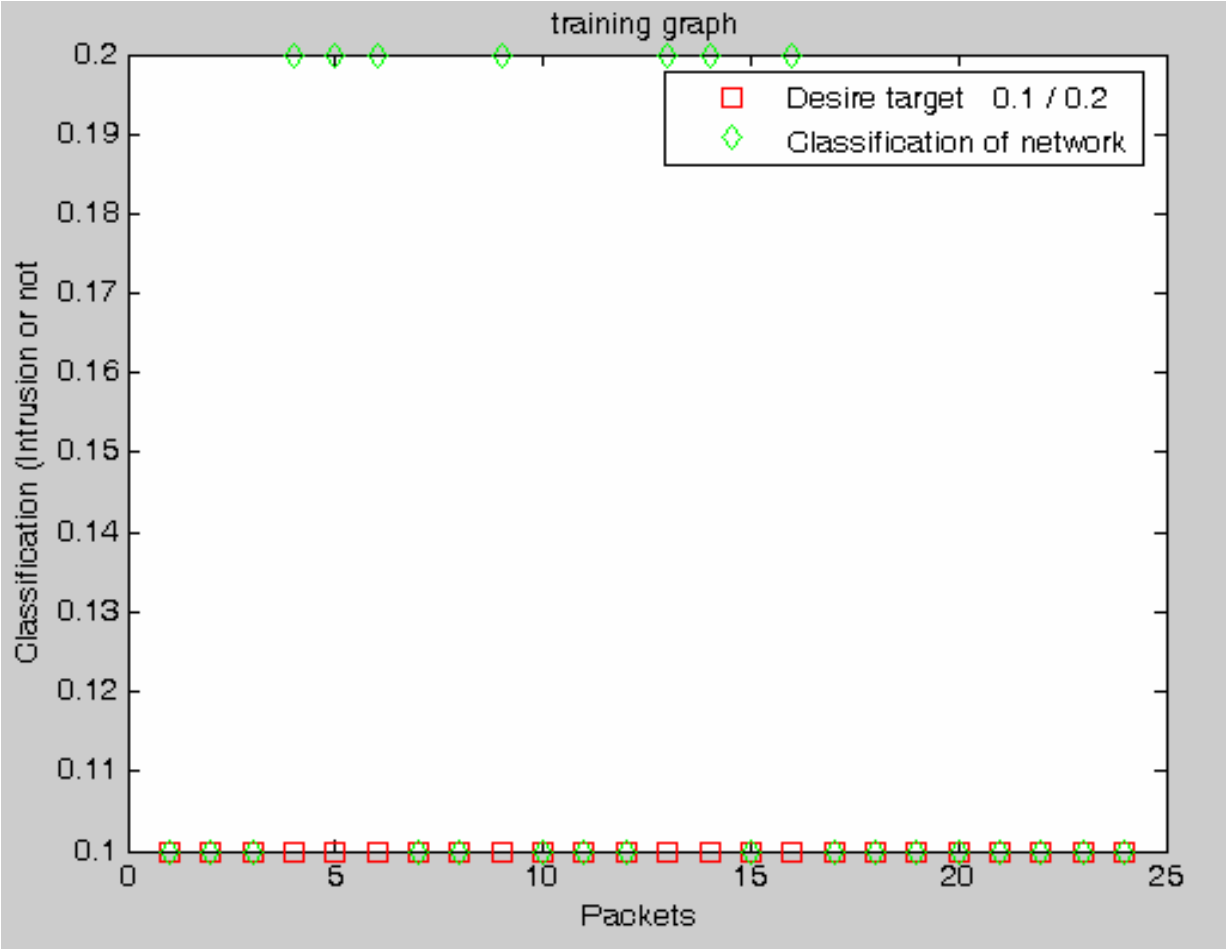

FIGURE 3. Packet classification

Table 3 : Distribution of patterns chosen for training

Packet Type	Total numbers used for training
Normal	17
Intrusion	7

Table 4 : Classification performance

Packet type	Total number tested	No. classified	No. misclassified
Normal	17	15	2
Intrusion	7	2	5

3. CONCLUSION AND FUTURE WORK

In this work, KDD dataset has been considered to experiment the performance of ESNN in classifying the LAN intrusion packets. A topology of 41 X 20 X 1 had been chosen. The future work will involve in implementing an echo state neural network for classification of intrusion packet. Future work will focus on comparison of echo state work with liquid state machine and back-propagation algorithm

4. REFERENCES

[1] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), Parallel distributed processing: Explorations in the microstructure of cognition (Vol.1, pp. 318-362).

[2] Denning, Dorothy. (February, 1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering, Vol. SE-13, No. 2.*

[3] Lippmann.R.P,; AN Introduction to computing with neural nets;IEEE Transactions on ASSP Mag. 35,4(2) 4-22, 1987.

[4] Fox, Kevin L., Henning, Rhonda R., and Reed, Jonathan H. (1990). A Neural Network Approach Towards Intrusion Detection. *In Proceedings of the 13th National Computer Security Conference.*

[5] HIROSE,Y., YAMSHITA,K., AND HIJIYA,S., 1991, Back-propagation algorithm which varies the number of hidden units, Neural Networks, Vol.4, No.1, pp-61-66.

[6] Debar, H. & Dorizzi, B. (1992). An Application of a Recurrent Network to an Intrusion Detection System. In *Proceedings of the International Joint Conference on Neural Networks.* pp. (II)478-483.

[7] Debar, H., Becke, M., & Siboni, D. (1992). A Neural Network Component for an Intrusion Detection System. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy.*

[8] Hammerstrom, Dan. (June, 1993). Neural Networks At Work. *IEEE Spectrum*. pp. 26-53.

[9] BERSHAD, N.J., SHYNK, J.J., AND FEINTUCH, P.L., 1993, Statistical analysis of the single-layer-back-propagation algorithm: Part-I-Mean weight behaviour, IEEE Trans. on Acoustics, Speech and Signal Processing, Vol.41, No.2, pp.573-582.

[10] BERSHAD, N.J., SHYNK, J.J., AND FEINTUCH, P.L., 1993, Statistical analysis of the single-layer back-propagation algorithm: Part-II-NMSE and classification performance, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol.41, No.2, pp.583-591.

[11] Helman, P. and Liepins, G., (1993). Statistical foundations of audit trail analysis for the detection of computer misuse, *IEEE Trans. on Software Engineering*, 19(9):886-901.

[12] Ilgun, K. (1993). USTAT: A Real-time Intrusion Detection System for UNIX. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*. pp. 16-28.

[13] Denault, M., Gritzalis, D., Karagiannis, D., and Spirakis, P. (1994). Intrusion Detection:

Approach and Performance Issues of the SECURENET System. In *Computers and Security Vol. 13, No. 6, pp. 495-507*

[14] Frank, Jeremy. (1994). Artificial Intelligence and Intrusion Detection: Current and Future Directions. In *Proceedings of the 17th National Computer Security Conference.*

[15] Mukherjee, B., Heberlein, L.T., Levitt, K.N. (May/June, 1994). Network Intrusion Detection. *IEEE Network*. pp. 28-42.

[16] Chung, M., Puketza, N., Olsson, R.A., & Mukherjee, B. (1995) Simulating Concurrent Intrusions for Testing Intrusion Detection Systems:Parallelizing. In *Proceedings of the 18th NISSC*. pp. 173-183.

[17] Cramer, M., *et al*. (1995). New Methods of Intrusion Detection using Control-Loop Measurement. In *Proceedings of the Technology in Information Security Conference (TISC) '95.* pp. 1-10.

[18] Kohonen, T. (1995) *Self-Organizing Maps*. Berlin: Springer.

[19] Staniford-Chen, S. (1995, May 7). Using Thumbprints to Trace Intruders. UC Davis.

[20] Tan, K. (1995). The Application of Neural Networks to UNIX Computer Security. In *Proceedings of the IEEE International Conference on Neural Networks, Vol.1* pp. 476 – 481.

[21] Ghost, A.K., *et al.* (September 27, 1997). "Detecting Anomalous and Unknown Intrusions Against Programs in Real-Time". DARPA SBIR Phase I Final Report. Reliable Software Technologies.

[22] Porras, P. & Neumann, P. (1997). EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proceedings of the 20th NISSC.*

[23] Puketza, N., Chung, M., Olsson, R.A. & Mukherjee, B. (September/October, 1997). A Software Platform for Testing Intrusion Detection Systems. *IEEE Software, Vol. 14, No. 5*

[24] Ryan, J., Lin, M., and Miikkulainen, R. (1997). Intrusion Detection with Neural Networks. *AI Approaches to Fraud Detection and Risk Management: Papers from the 1997 AAAI Workshop (Providence, Rhode Island)*, pp. 72-79. Menlo Park, CA: AAAI.

[25] Tan, K.M.C & Collie, B.S. (1997). Detection and Classification of TCP/IP Network Services. In *Proceedings of the Computer Security Applications Conference*. pp. 99-107.

[26] Sebring, M., Shellhouse, E., Hanna, M. & Whitehurst, R. (1988) Expert Systems in Intrusion Detection: A Case Study. In *Proceedings of the 11th National Computer Security Conference.*

[27] GRAHAM, R. 2000. FAQ: Network Intrusion Detection Systems, http://www.robertgraham.com

[28] Jaeger, H.; The echo state approach to analyzing and training recurrent neural networks; (Tech.Rep. No. 148). Bremen: German National Research Center for Information Technology, 2001.

[29] BACE, R. 2002. Intrusion Detection, Macmillan Technical Publishing

[30] Jaeger, H.; Tutorial on training recurrent neural networks, covering BPPT, RTRL,EKF and the "echo state network" approach (Tech. Rep. No. 159).; Bremen: German National Research Center for Information Technology, 2002.

[31] Jaeger, H;. Short term memory in echo state networks; (Tech. Rep. No. 152) Bremen: German National Research Center for Information Technology. 2002.

[32] KAZIENKO, P., AND DOROSZ, P. 2003, Intrusion Detection Systems (IDS) Part I – network intrusions; attack symptoms; IDS tasks; and IDS

[33] BACE, R AND MELL, P., 2004, NIST Special Publication on Intrusion Detection Systems, http://www.nist.gov

[34] GORDEEV, M. 2004. Intrusion Detection Techniques and Approaches, http://www.ict.tuwein.ac.a

[35] JEAN-PHILIPPE 2004, Application of Neural Networks to Intrusion Detection, http://www.sans.org

[36] Albert Mo Kim Cheng,On-Time and Scalable Intrusion Detection in Embedded Systems, Real-Time Systems Laboratory,Department of Computer Science,University of Houston, TX 77204, USA

[37] H. Güneş Kayacık, A. 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

[38] R.S.Michalski, K.A.Kaufman, J.Pietrzykowski, B.Sniezynski, J.Wojtusiak, Intelligent Information Systems 2006, New Trends in Intelligent Information Processing and Web Mining Ustron, Poland, June 19-22, 2006

[39] Steven Cheung, Bruno Dutertre, Martin Fong, Ulf Lindqvist, Keith Skinner and Alfonso Valdes, Using Model-based Intrusion Detection for SCADA Networks,Computer Science Laboratory,SRI International,December 7, 2006

[40] Ajith Abraham, Ravi Jain, Johnson Thomas and Sang Yong Han, D-SCIDS: Distributed SoftComputing intrusion detection system, Journal of Network and Computer Applications 30 (2007) ,PP 81–98