Artificial Neural Networks Architecture For Intrusion Detection Systems and Classification of Attacks

Mohammed Sammany[1], Marwa Sharawi[2], Mohammed El-Beltagy[3], Imane Saroit[4]

***Abstract*— The ubiquity of the Internet poses serious concerns on the security of computer infrastructures and the integrity of sensitive data. Intrusion Detection Systems (IDS) aim at protecting networks and computers from malicious network-based or host-based attacks. The underlying assumption of intrusion detection is an attack will noticeably affect system performance or behavior. Neural networks method is a promising technique, which has been used in many classification problems. The present study is aimed to solve a multi-class problem of intrusion detection using MLP in which not only the attack records are distinguished from normal ones, but also the attack type is identified. The results showed that the designed system is capable of classifying records with 93.43% accuracy with two hidden layers of neuron.**

***Index Terms*— Intrusion Detection Systems, Neural networks, Multi-layered Perceptron (MLP)**

I. Introduction

THE rapid development and expansion of World Wide Web and local network systems have changed the computing world in the last decade. The highly connected computing world has also equipped the intruders and hackers with new facilities for their destructive purposes. The costs of temporary or permanent damages caused by unauthorized access of the intruders to networks and computer systems have urged different organizations to, increasingly; implement various systems to monitor data flow in their networks. These systems are generally referred to as Intrusion Detection Systems (IDSs) [1]. There are two main approaches to the design of IDSs: *misuse* and *anomaly* detection techniques [2]. In a misuse detection based IDS, intrusions are detected by looking for activities that correspond to known signatures of intrusions or vulnerabilities. On the other hand, the anomaly detection based IDSs detect intrusions by searching for abnormal network traffic. The abnormal traffic pattern can be defined either as the violation of accepted thresholds for frequency of events in a connection or as a user's violation of the legitimate profile developed for normal behavior.

One of the most commonly used approaches in expert system based intrusion detection systems is rule-based analysis using Denning's profile model [3]. Rule-based analysis relies on sets of predefined rules that are provided by an administrator or created by the system. Unfortunately, expert systems require frequent updates to remain current. This design approach usually results in an inflexible detection system that is unable to detect an attack if the sequence of events is even slightly different from the predefined profile. The problem may lie in the fact that the intruder is an intelligent and flexible agent while the rule based IDSs obey fixed rules. This problem can be tackled by the application of soft computing techniques in IDSs. Soft computing is a general term for describing a set of optimization and processing techniques that are tolerant of imprecision and uncertainty. The principal constituents of soft computing techniques are Fuzzy Logic (FL), Artificial Neural Networks (ANNs), Probabilistic Reasoning (PR), and Genetic Algorithms (GAs) [4].

Classifiers are tools that partition sets of data into different classes based on specified features in that data. The idea behind the application of soft computing techniques and particularly ANNs in implementing IDSs is to include an intelligent agent in the system that is capable of disclosing the latent patterns in abnormal and normal connection audit records, and to generalize the patterns to new (and slightly different) connection records of the same class. In fact, there has been a huge literature achieved to address the intrusion detection problem using neural networks. For example, Ryan et al. [5] described an off-line anomaly detection system (NNID), which utilized a back-propagation MLP neural network. The MLP was trained to identify users' profile and at the end of each log session, the MLP evaluated the users' commands for possible intrusions (offline). Cannady [6] used a three layer neural network for offline classification of connection records in normal and misuse classes. The system designed in this study was intended to work as a standalone system (not as a preliminary classifier whose result may be used in a rule-based system). The feature vector used in [6] was composed of nine features all describing the current connection and the commands used. A dataset of 10,000 connection records including 1,000 simulated attacks was used. The training set included 30% of the data. The final

M. Sammany, Department of Mathematics, Computational Mathematics Branch, Faculty of Science, Cairo University, Egypt (e-mail: sammanyddr1@hotmail.com).

M. Sharawi, Department of Information Technology, Faculty of Computers and Information, Cairo University, Egypt (e-mail: msaid@Bue.edu.eg).

M. El-Beltagy. , Department of Decision Support System, Faculty of Computers and Information, Cairo University, Egypt (e-mail: Mohammed @ computer.org).

I. Saroit, Department of Information Technology, Faculty of Computers and Information, Cairo University, Egypt (e-mail: iasi63@hotmail.com).

result is a two class classifier that succeeded in classification of normal and attack records in 89-91% of the cases. In yet another study [7], the authors used three and four layer neural networks and reported results of about 99.25% correct classification for their two class (normal and attack) problem. Cunningham and Lippmann [8] used ANNs in misuse detection. They used an MLP to detect Unix-host attacks by searching for attack specific keywords in the network traffic. Different groups used self-organizing maps (SOM) for intrusion detection [9]. In most of the previous studies [5], [6], [7], the implemented systems were neural networks with two possible outputs: normal or anomaly. In these studies, some types of attacks and a set of normal records were included in the dataset; however, the output of the neural network was 1 or 0 for normal or attack conditions (the attack type was not determined by the neural network). This paper is aimed to solve an off-line multi class problem based on the idea of Moradi and Zulkernine [10] in which not only the attack records are distinguished from normal ones, but also the attack type is identified. However, the total average accuracy obtained by the proposed neural network model was 93.43%, using two hidden layers MLP. The early stopping criteria been used as a method to stop the training process, and thus to guarantee the generalization capability. The promising results of the present study show the potential applicability of ANNs for developing high efficiency practical IDSs.

Paper Organization: - Section I has introduced the basic ideas of intrusion detection and the motivations for this study. Section II has introduced the basic ideas about the intrusion detection, kinds of intrusion detection Systems, and types of attacks. Section III presents the Data set and the features used for classifying network connection records. Section IV presents the neural network method as an intrusion detector. In section V, we present the implementation procedure and the discussion of our experimental results. Finally, in Section VI the paper ends with a conclusion and the possibilities for future work.

II. Intrusion Detection System

A. An Overview

Intrusion detection is software, hardware or combination of both used to detect intruder activity by using a set of techniques and methods at the network and host level. There are two types of intrusion detection systems, that is, the *Network Intrusion Detection Systems* (NIDS) and the Host Intrusion Detection Systems (HIDS) [11]. NIDS are intrusion detection systems that capture data packets traveling on the network media and match them to a database of signatures, which are the pattern that we look for inside data packets. Depending upon whether a packet matched with an intruder signature, an alert generated. However, the HIDS installed as agents on a host. These intrusion detection systems can look into system and application log files to detect any intruder activity. Some HIDS are proactive, i.e. they can sniff the network traffic coming to a particular host on which the HIDS installed and alert in real time. These alerts are any sort of user notification such as, pop-up windows, logging to a console, sending e-mail and so on. Generally, Intrusion detection systems fall into three basic categories [12]:

- **Signature-based intrusion detection systems (Misuse Detection)**

  Like computer viruses, intruders have signatures that can be detected using software. Based upon a set of signatures and rules, these detection systems are able to find and log suspicious activities and generate alerts using a pattern matching technique.

- **Anomaly-based intrusion detection systems**

  It is also known as profile-based detection since the systems purpose is to examine the statistical and characteristic behavior. It usually depends on packet anomalies present in protocol header parts. In some cases, these methods produce better results compared to signature-based IDS.

There are four phases of the analysis process applied in the both types of detection system [12].These phases are *preprocessing*, *analysis*, *response*, and *refinement*(see Fig. 1).

Preprocessing is a process, which contains different classifications processes, with purpose to format the data collected from IDS. Formatted data can be easily broken down further into classifications. Those classifications depend on analysis methods we use, i.e. rule-based or profile-based detection. The ***Analysis*** phase is done through a comparison between knowledge base and classified data. The data can logged as intrusion in this phase. ***Response*** can be performed manually after analysis or automatically under present attack, it consists of few activities such are: alert sending via SMS, mail, SNMP trap etc. Finally, in ***refinement*** phase we try to fine-tune our IDS policy, which gives us opportunity to minimize number of false-positives.

- **Target Monitoring**

  In the third type of intrusion detection systems, a standing check after changes is done. Several objects as file-

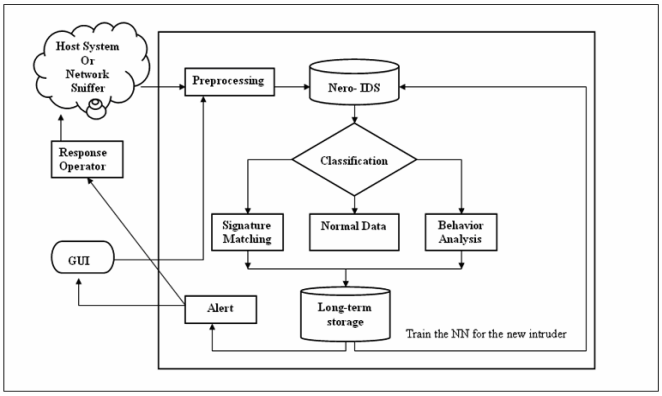


Fig.1. The phases of the analysis process in a proposed IDS system.

modifications and program-logon are of special interest to this kind of control. The control is done through a crypto-algorithm and a crypto-check sum is calculated from the objects of interest. The comparison between new and old check sums is made, and system is compromised, if there is mismatch [14].

B. *Types of Networking Attacks*

There are four major categories of networking attacks. Every attack on a network can be placed into one of these groupings [13].

- **Denial of Service (DoS):** A DoS attacks is a type of attack in which the hacker makes a memory resources too busy to serve legitimate networking requests and hence denying users access to a machine e.g. apache, smurf, Neptune, ping of death, back, mail bomb, UDP storm,etc.
- **Remote to User attacks (R2L):** A remote to user attack is an attack in which a user sends packets to a machine over the internet, and the user does not have access to in order to expose the machines vulnerabilities and exploit privileges which a local user would have on the computer, e.g. xlock, guest, xnsnoop, phf, sendmail dictionary etc.
- **User to Root Attacks (U2R):** These attacks are exploitations in which the hacker starts off on the system with a normal user account and attempts to abuse vulnerabilities in the system in order to gain super user privileges, e.g. perl, xterm.
- **Probing:** Probing is an attack in which the hacker scans a machine or a networking device in order to determine weaknesses or vulnerabilities that may later be exploited so as to compromise the system. This technique is commonly used in data mining, e.g. satan, saint, portsweep, mscan, nmap etc.

Two different attack types were included for this study: *SYN Flood (Neptune)* and *Satan.* These two attack types were selected from two different attack categories (denial of service and probing) to check for the ability of the intrusion detection system to identify attacks from different categories.

SYN Flood (Neptune) is a denial of service attack to which every TCP/IP implementation is vulnerable (to some degree). For distinguishing a Neptune attack, network traffic is monitored for a number of simultaneous SYN packets destined for a particular machine [14].

Satan is a probing intrusion, which automatically scans a network of computers to gather information or find known vulnerabilities.

The purpose of classifiers in IDSs is to identify attacks from all four groups as accurately as possible.

C. *Locations of Intrusion Detection Systems in Networks*

Usually an intrusion detection system captures data from the network and applies its rules to that data or detects anomalies in it. Depending upon the network topology, the type of intrusion activity (i.e. internal, external or both), and our security policy (what we want to protect from hackers), IDSs can be positioned at one or more places in the network [11]. For example, if we want to detect only external intrusion activities, and we have only one router connecting to the Internet, the best place for an intrusion detection system may be just inside the router or a firewall. On the other hand, if we have multiple paths to the internet, and we want to detect internal threats as well, we should place one IDS box in every network segment. Fig. 2 shows typical locations where you can place an intrusion detection system.

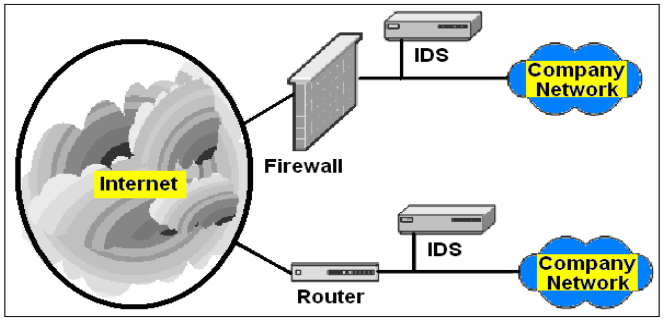


Fig. 2. Typical locations where an intrusion detection systems should be placed [11].

III. Data Sets and Features

The 1999 version of MIT Lincoln Laboratory – DARPA (Defense Advanced Research Projects Agency) intrusion detection data was used in this research [18]. According to [10] "The sample version of the dataset included more than 450,000 connection records. A subset of the data that contained the desired attack types and a reasonable number of normal events were selected manually..." The final dataset used in this study included 13000 records (the same used in [10]), and the data vector was a *35 dimensional* vector.
The features of the data set have been divided into three groups and provided descriptions for each group (A complete description of the full set of features is shown; see Fig.(A.1) in Appendix (A), as inputs to the neural network).

- **Group 1** includes features describing the *commands* used in the connection (instead of the commands themselves). These features describe the aspects of the commands that have a key role in defining the attack scenarios. Examples of this group are number of file creations, number of operations on access control files, number of root accesses, etc.
- **Group 2** includes features describing the *connection specifications*. This group includes a set of features that present the technical aspects of the connection. Examples of this group include, protocol type, flags, duration, service types, and number of data bytes from source to destination, etc.
- **Group 3** includes features describing the *connections to the same host in last 2 seconds.* Examples of this group are, number of connections having the same destination host and using the same service, number of connections

to the current host that have a rejection error, number of different services on the current host, etc.

Different possible values for selected features were extracted and a numerical value was attributed to each of them. Some of the features had binary values where some others had a continuous numerical range (such as duration of connection) [16].

IV. Artificial Neural Networks and Intrusion Detection

Neural networks are a uniquely powerful tool in multiple class classification [17], especially when used in applications where formal analysis would be very difficult or even impossible, such as pattern recognition, nonlinear system identification, and control. Provided the neural network has been given sufficient time to train, the property of generalization ensures that the network will be able to classify patterns that have never been seen before. However, the accuracy of classification problems depends on a variety of parameters, ranging from the architecture of the actual neural network to the training algorithm of choice.

MLP is a multiplayer feed-forward network consists of an input layer, one or more hidden layers, and an output layer. The output layer supplies the response of the network to the activation patterns applied to the input layer. The present study is aimed to solve a three-class class problem, which can be extended to cases with more attack types. The objective of MLP is to assign the input patterns to one of the categories that are represented in terms of neural networks' outputs, so that they represent the probability of class membership. The symbolic representation has been used to express each of the three conditions in such a way that, a "1" in an column indicates the occurrence of the column's corresponding string and a "0" indicates a non-occurrence. Thus, we have three cases of classification probabilities, that is, $[1\ 0\ 0]$ for Normal conditions, $[0\ 1\ 0]$ for Neptune attack and $[0\ 0\ 1]$ for the Satan attack. For the construction of MLP to manage the classification task of this problem, we put three neurons in the output layer (neuron for each class) with the *TanhSigmoid* activation function for every neuron. One problem that can occur during neural network training is over-fitting. In an over-fitted ANN, the error (number of incorrectly classified patterns) on the training set is driven to a very small value, however, when a new data is presented, the error is large. In these cases, the ANN has memorized the training examples; however, it has not learnt to generalize the solution to new situations. One possible solution for the over-fitting problem is an improving generalization is called *early stopping criteria* [18]. In this technique, the available data is divided into three subsets. The first subset is the training set, which is used for training and updating the ANN parameters. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error will normally decrease during the initial phase of training similar to the training set error. However, when the ANN begins to over-fit the data, the error on the validation set will typically begin to rise. When the validation error increases for a specified number of iterations, the training is stopped, and the weights that produced the minimum error on the validation set are retrieved [19].

V. Experimental Results and Classifier Evaluation

Using *Back-propagation algorithm* [20], a two hidden layers MLP with 50 and 30 neurons, in the first and the second layer respectively, has been implemented as an intrusion detector. The learning rate has been chosen in such a way that it has high value between the input and hidden layers, while they have a small values between the second and the output layer. This is due to the fact that the high values of learning rates in the backward layers speed up the convergence rate without affecting the stability of the network [21]. Fig. (A.1) (See Appendix A) illustrates the architecture and parameters used to in simulation process. The proposed model has been trained on a training set consists of 10400 patterns, twenty percent of the training set has been truncated for validation, and the network has been tested on 2600 pattern not seen before (apportioned equivalently upon the three classes). Table 1 shows the number of patterns that are correctly classified and rate of correct classification for each class in addition to the average classification accuracy of the three classes, which measures the whole performance of the proposed model. Fig. 3 reveals the mean square error of the training process versus the progress of training epochs for the training

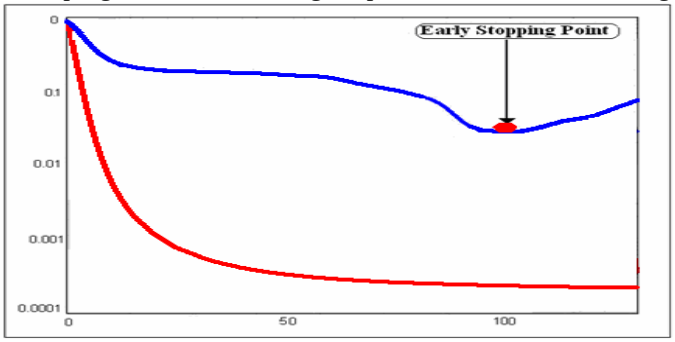


Fig. 3. Mean square error of the training process versus the progress of training epochs for the training and validation sets and validation sets, and the stopping criterion of the training.

Discussion

Early stopping criterion for validation set was applied to stop the training process. At the minimum point, the number of epochs was 100 and the value of Cross Validation Error (CVE) was 0.05. Then, the training has been continued again and the weights have been saved for each epoch until the training process has been stopped after 1000 epochs. While monitoring the CVE along with the MSE for the training process, it has been noticed that the CVE start to increase starting from this point, therefore, this point can be considered a good generalization point. Using the saved weights, the

TABLE I

CORRECT CLASSIFICATION RATE FOR EACH OF THE THREE CLASSES AND THE TOTAL AVERAGE CLASSIFICATION ACCURACY OF THE PROPOSED MODEL

Class Name	Number of Test Patterns for each Class	The Number of Correctly Classified Patterns	The Correct Classification Percentage
Normal	866	834	96.3 %
Neptune	866	801	92.4 %
Satan	866	794	91.6 %
Classification Rate Average			93.43 %

network has been tested on the early stopping. When unseen data (test set) was fed to the neural network the correct classification rate was more than 93.43%.

VI. CONCLUSION AND FUTURE WORK

In this paper we present an intrusion detection system based neural networks. The implemented neural network model used to solve a three-class problem, that is, normal, attack patterns, and the type of the attack. However, its further development to several classes is straightforward. When unseen data is presented to the model, the results obtained revealed a great deal of accuracy (93.43%) with two hidden layers of MLP.

Practical IDSs should have several attack types; therefore, it is possible, as a future development to the present study, to include more attack scenarios in the dataset. In addition, in order to avoid unreasonable complexity in the neural network, an initial reduction of the records to normal and general categories of attacks can be done as a the first step, before introducing the data to the network.

Genetic algorithms are a searching technique that is inspired by the mechanism of selection exists in the nature, where the stronger individuals are likely to survive in a population. As a future study, genetic algorithms can be used as a tool to optimize the architecture and parameters of a neural network given a specific fitness criterion. A comparative study with Tikhonov regularization method can take place as well.

ACKNOWLEDGMENT

The authors would like to thank Mr:

Mehdi Moradi, from Queen's University Kingston, Ontario, Canada, for presenting his work in this domain and for giving the data sets used in our experimentations, and for his guidance while doing this research.

REFERENCES

[1] Kemmerer, R. A., and Vigna, G., "Intrusion Detection: A Brief Introduction and History", *Security & Privacy, IEEE Computer Magazine*, pp. 27-30, 2002.

[2] W. Lee, S. J. Stolfo, K. Mok, "A Data Mining Framework for Building Intrusion Detection Models", *Proceedings of 1999 IEEE Symposium of Security and Privacy*, pp. 120-132.

[3] D. E. Denning, "An intrusion detection model," IEEE Transactions on Software Engi neering, vol. 13, no. 2, pp. 222– 232, 1987.

[4] Piero P. Bonissone, "Soft computing: the convergence of emerging reasoning technologies," Soft Computing Journal, vol.1, no. 1, pp. 6-18, Springer-Verlag 1997.

[5] J. Ryan, M. Lin, and R. Miikkulainen, "Intrusion Detection with Neural Networks," AI Approaches to Fraud Detection and Risk Management: Papers from the 1997 AAAI Workshop, Providence, RI, pp. 72-79, 1997.

[6] J.Cannady, "Artificial neural networks for misuse detection," Proceedings of the 1998 National Information Systems Security Conference (NISSC'98), Arlington, VA, 1998.

[7] Srinivas Mukkamala, "Intrusion detection using neural networks and support vector machine," Proceedings of the 2002 IEEE International Honolulu, HI, 2002.

[8] R.Cunningham and R. Lippmann, "Improving intrusion detection performance using keyword selection and neural networks," *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, Purdue, IN, 1999.

[9] P. Lichodzijewski, A.N. Zincir Heywood, and M. I. Heywood,"Host-based intrusion detection using self-organizing maps,"*Proceedings of the 2002 IEEE World Congress on Computational Intelligence*, Honolulu, HI, pp. 1714-1719, 2002.

[10] M. Moradi, and M. Zulkernine, "A Neural Network Based System for Intrusion Detection and Classification of Attacks," *IEEE International Conference on Advances in Intelligent Systems - Theory and Applications*, Luxembourg-Kirchberg, Luxembourg, November 15-18, 2004.

[11] R. Ur Rehman, "Intrusion Detection Systems with snort," Prentice Hall PTR Upper Saddle River, New Jersey 07458, www.phptr.com.,2003.

[12] C. Endorf, E. Schultz & Jim Mellander. "Intrusion Detection & Prevention. The McGraw-Hill/Osborne Companies". 2004.

[13] Das, K., J., "Attack Development for Intrusion Detection Evaluation,", *Master thesis, Massachusetts Institute of Technology (MIT), USA*, 2000.

[14] Kristopher Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," *Masters Thesis, MIT*, 1999.

[15] MIT Lincoln Laboratory, http://www.ll.mit.edu.

[16] A. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *Symposium on Applications and the Internet*, pp. 209–216, 2003.

[17] Theodorios and Konstantinos Koutroumbas, "*Pattern Recognition*," Cambridge: Academic Press, 1999.

[18] S.Haykin, " Neural Network Comprehensive Foundation," Second Edition , Prentice Hall , 1999.

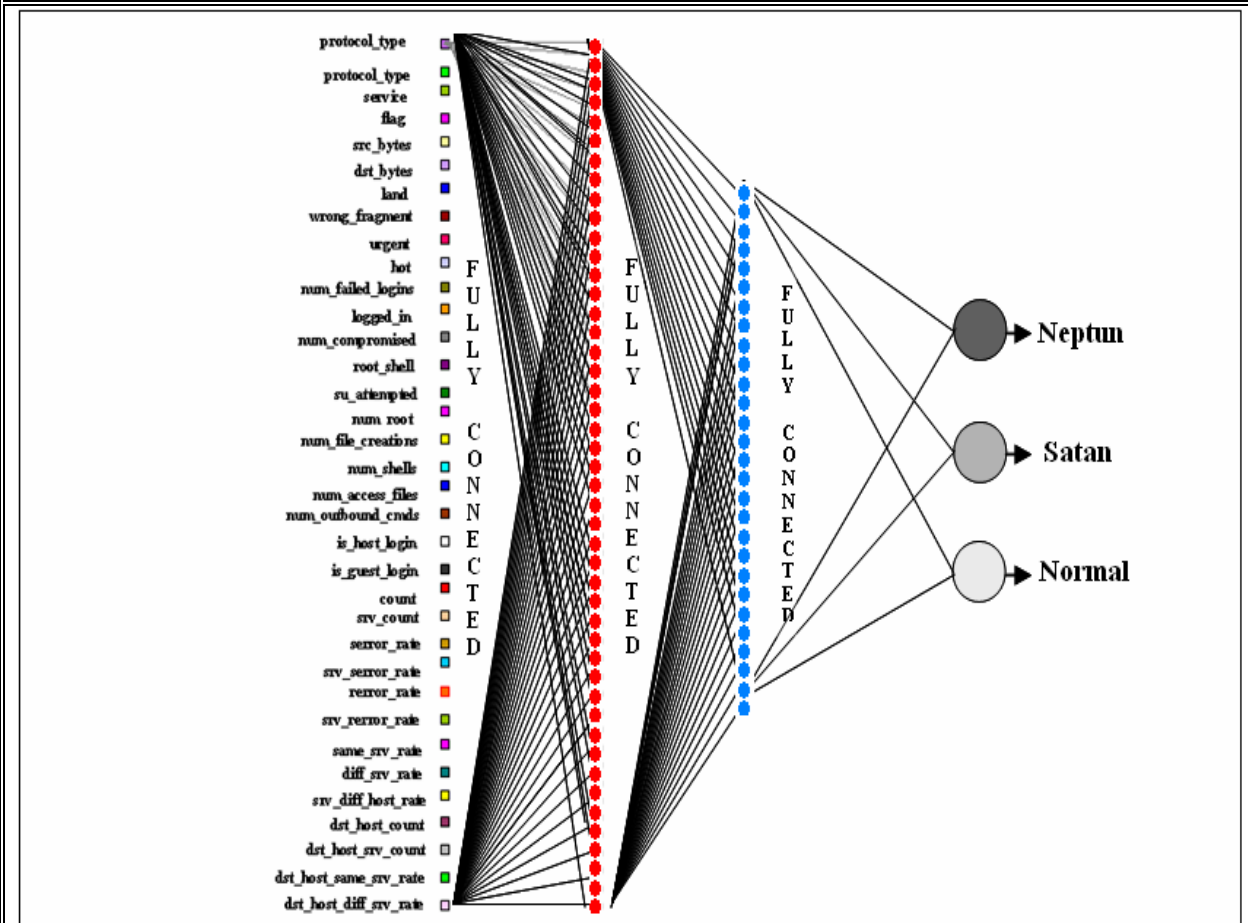
[19] MATLABOnlineSupport:www.mathworks.com/access/helpdesk/help/techdoc/matlab.sht ml.

[20] Rumelhart, D.E., Widrow, B., and Lehr, M.A. - "The Basic Ideas in Neural Networks", Communications *of the Association for Computing Machinery*, Vol. 37, No. 3,pp. 87 92.1994.

[21] Jacques Hadamard " Sur les problemes aux derivees partielles et leur signification physique" (1902)

APPENDIX (A)

Learning Rate			Momentum Constant			Eps	CVE	MSE
η_1	η_2	η_2	α_1	α_2	α_2	100	1×10^{-4}	1×10^{-5}
0.9	0.1	0.001	0.7	0.7	0.7			



1) length (number of seconds) of the connection (Numerical, Continuous) *2)* type of the protocol (tcp=0, udp=1, icmp=2) *3)* network service on the destination *4)* normal or error status of the connection (S0=0, SF=1, S1=2, REJ=3, S2=4) *5)* number of data bytes from source to destination (Numerical, Continuous) *6)* number of data bytes from destination to source (Numerical, Continuous) *7)* 1 if connection is from/to the same host/port; 0 otherwise *8)* number of "wrong" fragments (Numerical, Continuous) *9)* number of urgent packets (Numerical, Continuous) *10)* number of "hot" indicators (Numerical, Continuous) *11)* number of failed login attempts (Numerical, Continuous) *12)* 1 if successfully logged in; 0 otherwise *13)* number of "compromised" conditions (Numerical, Continuous) *14)* 1 if root shell is obtained; 0 otherwise *15)* 1 if "su root" command attempted; 0 otherwise *16)* number of "root" accesses (Numerical, Continuous) *17)* number of file creation operations (Numerical, Continuous) *18)* number of shell prompts (Numerical, Continuous). *19)* number of operations on access control files (Numerical, Continuous) *20)* number of outbound commands in an ftp session (Numerical, Continuous) *21)* 1 if the login belongs to the "hot" list; 0 otherwise *22)* 1 if the login is a "guest" login; 0 otherwise *23)* number of connections to the same host as the current connection in the past two seconds (Numerical, Continuous) *24)* number of connections to the same service as the current connection in the past two seconds (Numerical, Continuous) *25)* % of connections that have "SYN" (S0) errors (Numerical, Continuous) *26)* % of connections that have "SYN" errors (Numerical, Continuous) *27)* % of connections that have "REJ" errors (Numerical, Continuous) *28)* % of connections that have "REJ" errors (Numerical, Continuous) *29)* % of connections to the same service (Numerical, Continuous) *30)* % of connections to different services (Numerical, Continuous) *31)* % of connections to different hosts (Numerical, Continuous) *32)* number of connections having the same destination host (Numerical, Continuous) *33)* number of connections having the same destination host and using the same service (Numerical, Continuous) *34* % of connections having the same destination host and using the same service (Numerical, Continuous) *35)* % of different services on the current host (Numerical, Continuous) *Target* label of the sample connection (normal=0, Neptune=1, Satan=2)

Fig. (A.1). The architecture and the most important parameters of the neural network used in the simulation process. In addition, the features used of the Intrusion detection problem illustrated as inputs to the network.