International Journal of Computer Networks and Communications Security

VOL. 5, NO. 1, JANUARY 2017, 7–14
Available online at: www.ijcncs.org
E-ISSN 2308-9830 (Online)/ ISSN 2410-0595 (Print)

Towards A New Architecture of Detecting Networks Intrusion Based on Neural Network

Berlin H. LEKAGNING DJIONANG[1] and Dr. Gilbert TINDO[2]

[1, 2]University of Yaoundé I, Faculty of science, Yaoundé, Cameroon

[1]dberlinherve@gmail.com,[2]gtindo@uycde.uninet.cm

ABSTRACT

Networks intrusion detection systems appear nowadays as one of the most efficient solution for detecting illegal or suspicious activities in a network. Using neural networks to meet this objective has been studied by many authors. Most of the solutions provided in the literature face the problem of relevance and reliability. One of the major reasons of such a situation is the misconception of a correct profile. In this paper, a modular architecture is proposed, in which each module is dedicated to detecting a particular type of attack. Firstly, each module is trained with all attributes, then secondly some of the attributes used for training are pruned from the training set. This modularity allows us to know which of the system's module threaten the NIDS performance. The experimentation led us to observe that, depending on the type of the attack, some of the attributes have a marginal effect, although in the opposite, some other have a dominant effect. We have done a comparative study of the solution proposed and the others in the literature. It appears that our solution is more efficient on certain type of attacks. The NSL-KDD dataset has been used to train, test and evaluate our architecture.

Keywords:*NIDS, Neural networks, MLP, NSL-KDD Dataset.*

1 INTRODUCTION

The advent of networks has allowed a wide range of services. These services are subject to many attacks and thus security mechanisms have become a necessity. Network intrusion detection systems (NIDS) are one of the most used mechanisms nowadays to detect intrusions. The purpose of network intrusion detection systems is to protect networks against attacks that can be identified by firewalls.

Intrusion detection systems are generally classified in three categories [1]:the IDS which use the behavioral approach, the IDS which use the scenario approach and the IDS which use the specification approach. The behavioral analysis actually operates in two steps: a learning step and a detection step. During the learning step, the system learns how to recognize what is an abnormal behavior, and during the detection step, the system identifies abnormal behaviors [2]. The scenario analysis doesn't have a learning step. It comes with a predefined database of attack scenarios that they have to recognize (signature). The approach based on specifications is a method that builds the normal profile without using any learning algorithm [3].

Many good profiles learning methods (statistics, artificial learning and expert systems) have been used in the literature. Neural networks have been used by many authors to build a network intrusion detection system [4, 5, 6, 7, 8, 9]. One of the major NIDS' problems is that the efficiency is governed by a single whole system which is in the rule of either detecting types of attacks or detecting classes of attacks. We propose a modular architecture for detecting networks intrusions by types of attacks using neural networks. This will help us to know which modules pull down performances. Our architecture is based on the behavioral approach.

One of the major problems of this approach is the design of a normal profile [10]. The automatic building of the fields that make a user profile is a real challenge. The quality of the fields is an important factor on the efficiency of the NIDS [11]. We experiment in this work, the importance of selecting attributes in order to build the right profiles.

The remainder of our paper is organized as following : we will present and justify the choice of neural network model for referent profiles learning in section 2 ; in section 3 we have given a brief review of the works related to the neural networks ; at the section 4, we propose our architecture and its

functioning ; the section 5 is devoted to the description of inputs and the preprocessing mode of them ; then we go through the experiment and the analysis of results of our works in section 6 et 7 ; then we end this work with a conclusion in the section 8.

2 NEURAL NETWORKS

Neural networks are networks that contain nodes which are strongly connected, with elementary processors working in parallel and linked by weighted edges. These connection weights govern the functioning of the network. Each elementary processor computes a unique output based on the received inputs. Neural networks have many advantages in implementing a network intrusion detection system. They are really efficient and fast in the task of classification [4]. They are able to learn easily and identify new threats which are submitted to them. Neural network are able to deal with incomplete and imprecise data, and from multiple sources. The natural speed of neural networks helps to reduce damages when a threat is detected [5]. Neural networks usage helps to extract nonlinear relationships that exist between different fields of a packet, and to real-timely detect complex attacks [6]. Neural networks, after having learnt correctly, have a good ability of generalization. They are able to compute the adequate outputs with precision even for data that haven't been learnt. The flexibility given by neural networks is also one of the intrusion detection's assets [12].

3 PREVIOUS WORKS

James Canady [5] is the first researcher to propose an intrusion detection system based on network protocols analysis. He uses the multi-layers perceptron (MLP) scheme for learning. The table below (Table1) shows some results of those researchers who used neural network models to design an intrusion detection system. The learning process used to modify the network parameters by those researchers was [13]:

- ✓ Present data, formatted as a vector, to the neural network;
- ✓ Check if the generated outputs match with the desired outputs;
- ✓ Change the network parameters to try to match the desired outputs.

Tableau 1: Some works results

Criteria / Author	Recognition rate	False positive	False negative
Alan 2002 [14]	24%	-	-
Mehdi 2004 [13]	87%	-	-
Golovko 2005 [6]	94,3%	-	-
Vaitsekhovich 2008[7]	93,21%	12,90%	-
Khattab 2009 [15]	97%	2,4%	0,8%
Iftikar 2009 [16]	98%	1,5%	-
Muna 2010 [9]	78%	-	-
Aslihan 2012 [8]	93,42%	2,95%	-
Yousef 2012 [12]	95,4%	2,6%	-

4 PROPOSED ARCHITECTURE AND ITS FUNCTIONNING

The architecture proposed is described in this section, before showing its functioning. Then we stand out and demonstrate one of its properties.

4.1 Architecture

We propose a new modular architecture in which each module helps to detect only one type of attack. Each module discriminate each type of attack to normal packets. This architecture's modularity helps us to solve the performance problem of NIDS. We propose a bi-neural model to achieve that. We define bi-neural networks as networks that discriminate two classes. Our architecture (**Figure 1**) propose a binary discrimination model (B_MLP) that doesn't select any attribute and a binary discrimination model (SB_MLP) that statically selects attributes. The second model aim to see the impact of attributes selection on the classifier's decision. Form types of attacks, we should usem+1 models (B_MLP) in our architecture. Our architecture has four levels, in which the first one preprocess the data. The second one helps to discriminate normal packets from abnormal ones. If the packet that has been analyzed is abnormal, then the packet is given to the others models (level three) in order to determine which type of attack it is. The element A (level four)in these architectures choose a reference which will decide which type of attack it is. The algorithm used to learn from these networks is described in **Figure 2**. Each model is made of three layers: one entry layer, one hidden layer and one output layer. For each model, we look for the number of neurons from the hidden level that helps to get the best dynamical detection rate.

We have named this architecture MAMBiM: Multiple Attacks, Multiple Binary MLP.

4.2Functioning

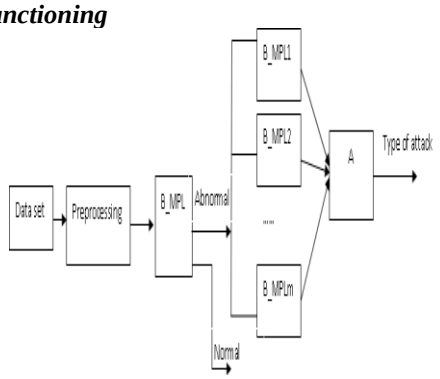

Fig. 1. Four level intrusion detection architecture

1. *Normalize the inputs.*
2. *For each network, look for the number of neurons from the hidden layer that gives the best recognition rate and save it in a file.*
3. *Learn from each network using the reverse propagation algorithm. Save the weights obtained in a file.*
4. *Evaluate our network with evaluation data. This is done using the weights saved while learning and the number of neurons from the hidden layer obtained.*

Fig. 2. Algorithm executed by each network

For each data vector, we make the preprocessing step as mentioned in section 5. The errors reverse propagation algorithm is used for each of our bi-neuronal models learning. Each bi-neuronal network has to be optimal when classifying each type of attack in such a way that the recognition rate gets closer 100.

4.3 Training and Test Step

We have to train B_MLP with the training dataset and test it with test data. We have to do the same task for B_MLPi with the training set corresponding to the normal packets and their respective types.

4.4 Evaluation types

The first level of our network discriminate abnormal profiles from normal ones. If the data vector that has been read is an abnormal packets, then we give it to the second binary neural network. Each model produces an output and a referee will decide according to these outputs which type of attack it is. The referee will only handle a vector of ***m*** outputs. The smaller output will determine the type of attack. And if all the others are greater than the threshold value or the classification criteria, then it is an unknown type of attack.

4.5 Properties

Let m be the number of types of attack to be detected. If n is the input data size and p the number of neurons from the hidden layer of each bi-neural model (B_MLP), the size of the architecture in

terms of the number of neurons is in the order of O $(p * n * m)$.

Proof: The size of our network is $(p * n + 1)(m + 1)$.

5 DESCRIPTION OF DATA AND PREPROCESSING

Since 1999, KDD Cup 99 is used as an example of dataset in behavioral intrusion detection systems [17]. Each packet from the KDD Cup 99 dataset is made of 41 fields and labeled as normal or abnormal packets with the types of attacks. Besides these fields, 37 are of type numeric and 4 are of type non numeric. KDD Cup 99 regroups 37 types of attacks. These attacks are subdivided into four major classes: DOS, U2R, R2L and Probes [18, 19].

- **DOS (Denial of service attacks):** they aim to threaten services availability by saturating computer resources, server of targeted networks. These attacks achieved in networks have as direct consequences the freezing of network traffic.
- **Probes:** attacks that aim to gather information from the target, which can help attacker to start on attack. There exist many types of probe attacks: some abuse rightful users and others use the engineering scheme to gather information.
- **R2L (Remote To Local):** attack that aim to bypass or usurp authentication credentials of a target in order to execute some commands. Most of these attacks derive from social engineering [17].
- **U2R (User To Root):** The attacks come from inside. The attacker usurps the administrator's password and thus the others users. Most of these attacks derive from the saturation of a buffer caused by programming errors [17].

KDD99 dataset contain many redundant packets within both training and test data [17]. Redundant data are able to give more importance to a type of attack than what it merits. [17] Proposes NSL-KDD which is an excellent dataset for comparing network IDS. Our experiment has been done using NSL-KDD. The type of attacks and their number in the training datasets and test datasets are proposed in **table 2** in appendix.

5.1 Preprocessing

Preprocessing is related to non-numeric fields. The non-numeric fields are: type of protocol (TCP, UDP, ICMP), type of service (aol, auth, bgp … Z39_50), flag (OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2, S3, SF, SH) and the class of packet (Normal or Abnormal). Regarding the type of protocol, we affect the following numeric values: TCP=1, UDP=2 and ICMP=3. We affect 1 to normal packets and 0 to abnormal packets. Regarding the fields type of service and flag, we can assign them numerical values mapped to the increasing or decreasing order of their total number. [20] Has shown the limits of such an approach, he propose to assign random values to these fields. We have, in our works, given a random value in the range 1 to 10 to the flag fields, and a random value in range 1 to 65 to the type of service field.

5.2 Normalization

It consists on transforming data to make them vary between 0 and 1, in order to make them more homogeneous and then simplify the learning process from the network. In this paper, we will use the Min-Max normalization. Let min_x and man_x the minimum and the maximum of the values of the attribute X, which has value V, the normalized value $\boldsymbol{V}' = \frac{v - min_x}{man_x - min_x}$. For each attribute from the data vector, compute its normalized value replace it by the normalized value computed.

6 EXPERIMENTS AND RESULTS

The experiment has been done on both our two models. The results by types of attacks and category of attacks are respectively presented in **Figure 5**, and in **table 3**. To evaluate our models, we will use four indicators which are: The recognition rate (TR), the false positive rate (TFP), the detection rate (TR) and the false negative rate (TFN). These rates as evaluated as following*:*

- $\boldsymbol{TR} = \frac{NN + AA}{NN + AA + AN + NA} * \mathbf{1}00,$
- $\boldsymbol{TFP} = \frac{NA}{NA + AA} * \mathbf{100},$
- $\boldsymbol{TFN} = \frac{AN}{AN + NN} * \mathbf{100},$
- $\boldsymbol{TD} = \frac{AA}{AA + NA} * \mathbf{100}$with :

NN: Normal packet detected as a Normal one;

NA: Normal packet detected as an abnormal one;

AN: Abnormal packet detected as a Normal one;

AA: Abnormal packet detected as abnormal one.

For the experimentation, 80% of data are used for training, in which 20% are reserved for evaluation and 20% are reserved for testing. The set of data submitted to each network is reduced compared to initial data.

.

The experiment results are presented in Table 3 and 4, and the comparative plot in **Figures 3, 4, 5** and **6**. **TA** represents the type of attacks in the plot drawn We have two models of our architecture, the first (Arch_m1) that doesn't take care of attributes selection and the second one (Arch_m2) that takes care of it.

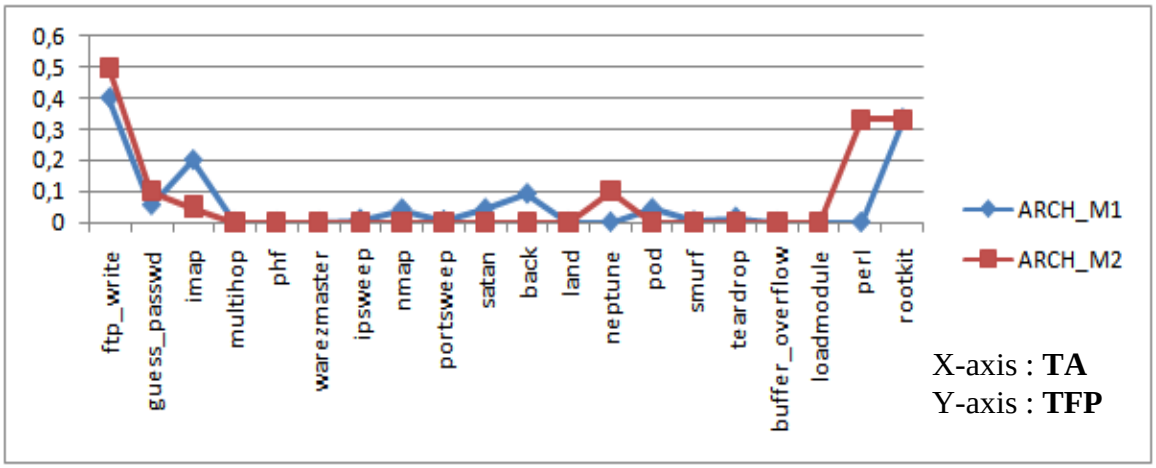

Fig. 3. Comparative study of false positive rate in both models

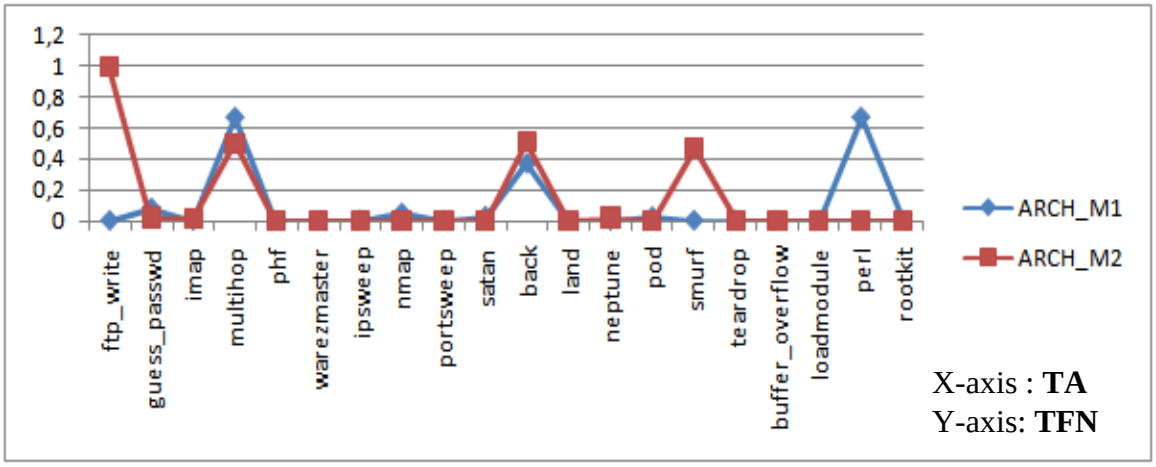

Fig. 4. Comparative study of false negative in both models

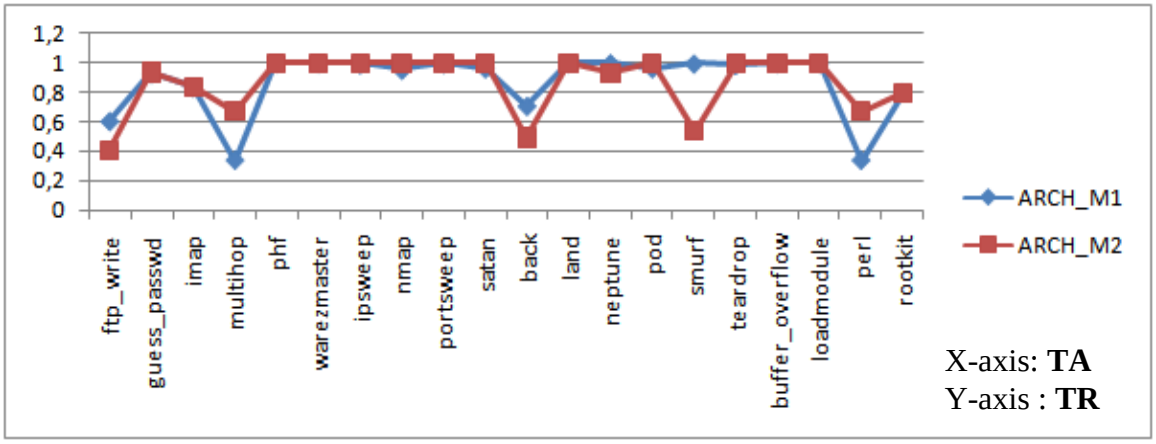

Fig. 5. Comparative study of recognition rate in both models

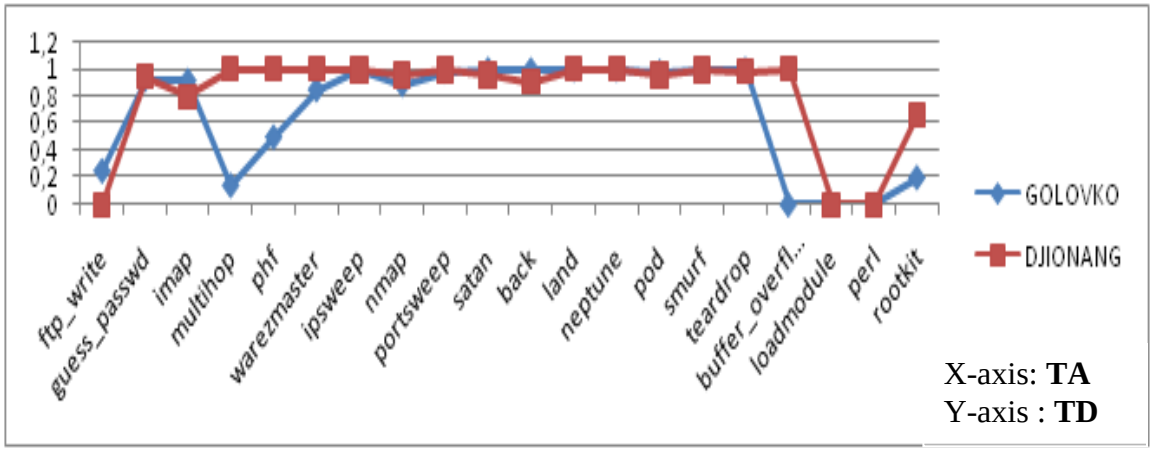

Fig. 6. Comparative study with the GOLOVKO works

7 RESULTS ANALYSIS

We have, for our second model architecture, fixed 12 attributes which are: duration, protocol, service, flags, source address, destination address, land, wrond_fragment, urgent, hot and the class (Normal, Abnormal). Looking the results shown in table 3, we notice that for certain type of attacks, the recognition rate decreases. That is the proof that this set of attributes is not pertinent to detect this type of attacks. An important requirement appears in the way of choosing relevant attributes that can describe each attack.

Figures 3 and **4** present a comparative study of both models in regard to false positive rate and false negative rate. The pics (ftp_write, back, smurf) that we observe in **Figure 4** teach us many things on the impact of selecting bad attributes in the false positive or false negative rate.

The Table 4 compares our results with the results shown in [6].The detection rate is used because it's the one used by Golovko to present his results. We have chosen it to compare our results simply because it detects types of attacks, like our architecture. **Figure 6** shows how the plot representing our results is clearly above the plot representing Golovko's works. Our results, compared to Golovko's results, are broadly better and more observable for the attacks of type R2L.

Table 3: Comparative study of categories of attacks

Category	Frequency	Recognition rate	
		Arch_m1	Arch_m2
R2L	103	86%	71,8%
PROBES	12630	96%	96,12%
DOS	44953	99,9%	91,51%
U2R	52	92,3%	94,2%

Table 3 clearly shows us how selecting non representative attributes can impact global recognition rate. The DOS attacks case is the most perceptible. We go from a 99,9% classification rate in the architecture **model 1** to 91,51% classification rate in the architecture **model 2**. This reveals a fact: the choice of inputs data has a considerable impact on the quality of an intrusion detection system.

Figure 5 shows us that, globally, the plot representing architecture **model 2** results is almost above the plot representing the results of architecture **model 1**. This is the proof that selection, when it is well done, helps to better the IDS's quality.

8 CONCLUSIONS

We have in this paper, proposed a modular architecture for network intrusion systems based on neural networks. The modularity of our architecture allows us to remove bricks that threaten the intrusion detection systems performances. Our works permit us to show the interest of selecting attributes. The architecture with a layer dedicated to attributes selection helps to get a good classification for certain types of attacks. Our work give a better performance compared to Golovko's works. In terms of perspectives, we plan to improve our architecture by selecting dynamically the relevant attributes.

Table4: Comparative Study

Type of attack	Detection rate(TD)	
	GOLOVKO2005	DJIONANG2015
ftp_write	25%	-
guess_passwd	92,45%	94,44%
imap	91,67%	80%
multihop	14,29%	100%
phf	50%	100%
warezmaster	85%	100%
ipsweep	99,12%	99,06%
nmap	88,74%	95,74%
portsweep	98,81%	99,44%
satan	99,81%	95,35%
back	99,5%	90,83%
land	100%	100%
neptune	99,98%	99,93%
pod	98,11%	95,53%
smurf	100%	99,63%
teardrop	99,8%	98,34%
buffer_overflow	0%	100%
loadmodule	0%	-
perl	0%	-
rootkit	20%	66,67%

9 ACKNOWLEDGMENT

The authors would like to thank ISESTMA for supporting the development of this work.

10 REFERENCES

[1] Asmaa Shaker, Sharer Gore « Importance of Intrusion Detection System » International Journal of Scientific & Engineering Research, Janvier 2011.

[2] Ludovic Mé and Véronique Alanou. Détection d'intrusion dans un système informatique : Méthodes et outils. TSI, 15(4):429–450, 1996.

[3] Guillaume Hiet, « Détection d'instructions paramétrée par la politique de sécurité grâce au contrôle collaboratif des flux d'informations au sein du système d'exploitation et des applications : mise en œuvre sous linux pour les programmes java » Université de Rennes, Décembre 2008

[4] G. DREYFUS "les réseaux de neurones" Mécanique Industriel et Matériaux, n51, septembre 1998

[5] Cannady,J. , "Artificial Neural Network s for Misuse Detection," Proceedings, National Information Systems Security Conference (NISSC '98), October, Arlington , VA, pp . 443 -456. 1 998

[6] Vladimir Golovko, PavelKochurko "Intruision recognition using neural networks" International Scientific Journal of computing, 2005, vol. 4, Issue3, 37-42

[7] Leanid VAITSEKHOVICH, Vladimir GOLOVKO « Employment of neural network baser classifier for intrusion detection » actamechanica et automatica, vol2, no4, 2008

[8] AslihanOzkaya && BekirKarlik "Protocol Type Based Intrusion Detection Using RBF Neural Network" International Journal of Artificial Intelligence and Expert Systems (IJAE), volume (3): Issue (4):2012

[9] Muna Mhammad&& Monica Mehrotra "Design Network Intrusion Detection System using Hybrid Fuzzy-Neural Network" International Journal of Computer Science and Security, volume(4): Issue (3): 2010

[10] Ning P. &&Jajodia S. Intrusion Detection Techniques. In H. Bidgoli (Ed.), +-theInternet Encyclopedia. John Wiley & Sons. (2003).

[11] GuofeiGu&al "Towards an Information-Theoretic Framework for Analyzing Intrusion Detection Systems" Computer Security ESORICS 2006 (2006): 527-546

[12] Yousef Abuadlla&& all "Flow-Based Anomaly Intrusion Detection System Using Two Neural Network Stage", Computer Science and Information systems 11(2): 601-622: 2012

[13] Mehdi MORADI and Mohammad ZULKERNINE, "A Neural Network based System for intrusion detection and Classification of Attacks" In 2004 IEEE International on Advances in Intelligent Systems.

[14] ALAN BIVENS && all "Network based intrusion detection using neural network" Intelligent Engineering Systems through Artificial Neural network, 2002, pp. 579-584

[15] M. Khattab Ali &all "The Effect of Fuzzification on Neural Networks Intrusion Detection System", IEEE computer society: 2009

[16] Iftikhar Ahmad && all "Application of Artificial Neural Network in Detection of Probing Attacks" 2009 IEEE Symposium on Industrial Electronics and Applications(ISIEA 2009), October 4-6, 2009, Kuala Lumpur, Malaysia

[17] Mahbod Tavallaee&& all "A Detailed Analysis of the KDD CUP 99 Data Set" Proceding of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Application (CISDA 2009)

[18] Srinivas Mukkamala&& all "Intrusion detection using an ensemble of intelligent paradigms", Journal Network and Computer Applications 28 (2005), 167-182

[19] Matthew Vincent Mahoney "A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic ", these of Florida Institute of Technology, May 2003

[20] Aslihan Ozkaya&&Bekir Karlik "Protocole Type Based Intrusion Detection Using RBF Neural Network" International Journal of Artificial Intelligence and Expert Systems (IJAE), volume (3): Issue (4):2012.

APPENDIX

Attacks from NLS-KDD99, grouped by category. These data are organized in term of training and test data.

Table 2: Type of Attacks

Category	Type of attack	Training	Test	Category	Type of attack	Training	Test
Normal	Normal	67 343	9711	DOS	neptune	41214	4657
R2L	ftp_write	8	3		pod	201	41
	guess_passwd	53	1231		processtable	0	685
	httptunnel	0	133		smurf	2646	665
	imap	11	1		teardrop	892	12
	multihop	7	18		udpstorm	0	2
	named	0	17	U2R	buffer_overflow	30	20
	phf	4	2		loadmodule	9	2
	sendmail	0	14		perl	3	2
	snmpgetattack	0	178		ps	0	15
	snmpguess	0	331		rootkit	10	13
	warezmaster	20	944		sqlattack	0	2
	worm	0	2		xterm	0	13
	xlock	0	9				
	xsnoop	0	4				
Probes	ipsweep	3599	141				
	mscan	0	996				
	nmap	1493	13				
	portsweep	2931	157				
	saint	0	319				
	satan	3633	735				
DOS	apache2	0	734				
	back	956	359				
	land	18	7				
	mailbomb	0	293				