IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 3, Ver. IV (May-Jun. 2014), PP 119-125
www.iosrjournals.org

Simulation of Denial of Service (DoS) Attack using Matlab and Xilinx

[1,] I Mukhopadhayay, [2,] S Polle, [3,] P Naskar
[1,2,3,] Institute of Engineering & Management Y-12 Saltlkae Electronics Complex, Kolkata-64, India

Abstract: *As the Internet is growing – so is the vulnerability of the network. Companies now days are spending huge amount of money to protect their sensitive data from different attacks that they face. DoS or Denial Of Service attacks are one of such kind of attacks. In this paper, we at first recognize different kinds of DoS attacks and then propose a new methodology to simulate those.*
Keywords: *Denial of service (DoS), neural network tool (nntool), distributed denial of service (DDoS).*

I. Introduction

DoS attacks today are part of every Internet user's life. They are happening all the time, and all the Internet users, as a community, have some part in creating them, suffering from them or even loosing time and money because of them. DoS attacks do not have anything to do with breaking into computers, taking control over remote hosts on the Internet or stealing privileged information like credit card numbers.

In computing, a denial-of-service attack (DoS attack) or distributed denial-of-service attack (DDoS attack) is an attempt to make a machine or network resource unavailable to its intended users. The intent of a denial-of-service (DoS) attack is to overwhelm the targeted victim with a tremendous amount of bogus traffic so that the victim becomes so preoccupied processing the bogus traffic that legitimate traffic cannot be processed. The target can be the firewall, the network resources to which the firewall controls access, or the specific hardware platform or operating system of an individual host.

If a DoS attack originates from multiple source addresses, it is known as a distributed denial-of-service (DDoS) attack. Typically, the source address of a DoS attack is spoofed. The source addresses in a DDoS attack might be spoofing, or the actual addresses of compromised hosts used as "zombie agents" to launch the attack. This way, an attacker can be anyone with a certain knowledge and access privilege with the master host. All he has to do is to enter a few commands, and the whole zombie army would wake up and mount a massive attack against the victim of his choice.

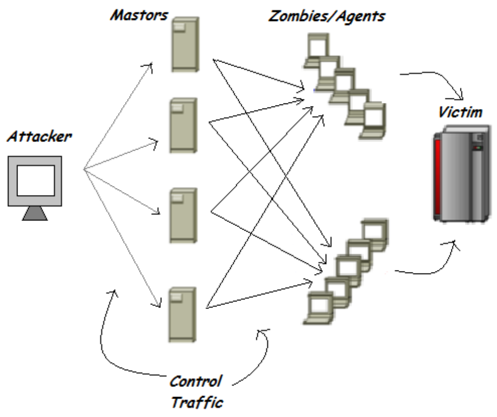


Figure 1 Description of a DDoS attack

Actually, the majorities of denial of service attacks are of distributed type, and have as basis flooding of large quantities of packets. Moreover, used packets might be changed to increase the harmfulness of the attack.

The DoS attacks can be roughly divided into OS-related attacks and networking-related attacks. For OS-related attacks are vulnerable, but most vendors of operating systems have fixed the problem in their latest versions. The vendors also provide patches for their vulnerable OS. Bonk / boink / newtear / teardrop2 is an attack resulting in blue screen freeze and crash. Ping of Death is an attack taking advantage of a known bug in TCP/IP implementation. The attacker uses the ping system utility to make up an IP packet that exceeds the maximum 65,536 bytes of data allowed by the IP specification. Systems may crash or reboot when they received such an oversized packet. Teardrop is an attack exploiting a weakness in the reassembly of IP packet fragments. The attacker creates a sequence of IP fragments with overlapping offset fields. Some systems will crash or reboot when they are trying to reassemble the malformed fragments. SYN flooding is an attack exploiting the three-way handshaking of TCP. The attacker sends the targeted system a flood of SYN packets with spoofed source address, until the targeted system uses up all slots in its backlog queue. Land is very similar to SYN flooding. The adversary floods SYN packets into the network with a spoofed source IP address of the targeted system. Smurf is a new kind of DoS attack. A smurf attacker cripples your router with ICMP echo request packets. Snork is an attack against Windows NT RPC service. It allows an adversary with minimal resources to cause a remote NT system to consume 100% CPU Usage for an indefinite period of time.

After the introduction in section 1, we describe related works in section 2. We propose the architecture in section 3. Section 4 describes Simulation environment and the simulation results we have obtained. The paper is concluded in section 5 with some highlights on future works.

II. Background Work

Some researchers have employed the use of a software simulation tool the DDoSSim which has been developed for comprehensive study of internet DDoS attacks, they reiterated that the DDoSSim enables one to deeply investigate different forms of attacks and protection schemes; the tool has the ability to provide useful recommendations on selecting best protection methods. They make use of the agent-based approach; furthermore, they conducted experiments for protection against DDoS attacks in order to demonstrate some potentials of the DDoSSim. Moreover, they considered the different phases of defence operations, which include the learning, decision making and protection. They further investigated into the adaptation of these protection methods to the actions of the attacker(s).[2]

They suggested a common approach and simulation environment for finding adequate defence methods against DDoS attacks, Attack and defence methods they used include: the attacker which could be a Daemon or a Master, on the other hand, the defence agents are categorized into: initial information processing (sensor), secondary information processing (sampler), attack detection (detector), filtering (filter), and finally the investigation (investigator).

Other researchers have re-emphasize the fact that in order to fight DDoS attacks there is the need for fully understanding the theoretical basis upon which we can protect systems against such attacks, they propose an agent based framework for simulating and modelling DDoS attacks. Furthermore, they presented of a formal specification of a representative spectrum of DDoS attacks; finally they implement an agent based software tool that has the potential of simulating DDoS attacks and responses of victim systems. [3]

The other experiment conducted is to check the ability of the Attack simulator to simulate different forms of attacks. Moreover, they reiterated that the purpose of the simulation based exploration of the Attacker tools is for firstly to check the network security policy at conceptual design stage and then secondly to check security policy of real life attacks.

The authors conducted experiments for various parameters of attack type specification and different victim configurations, also put into consideration is attackers intention as well as influence on input parameters such as; degree of protection given by the network and personal firewalls, victim of the attack, and the degree of the attackers knowledge of the network. Simulation results they obtained include parameters such as; number of terminal level attack options, percentage of attackers intention that are successful, percentage of effective network responses on attack actions, percentage of attack actions that were blocked by firewall, and percentage of ineffective results of attack actions.

Other authors attempt to propose a systematic method for DDoS attack detection. They base their detection on unusual behaviour identification. Furthermore, they utilize energy distribution based on wavelets analysis to detect DDoS attacks, in addition, they mention that in attack free situations, the energy distribution will have limited variations while in attack situations, the traffic in the network will cause a significant energy distribution deviations in a short period of time. They performed experiments on typical internet traffic and results they obtained shows significant changes in energy distributions in DDoS attack situations. Moreover they suggest that this spike in energy distribution should be captured in the early stages of attack to prevent eventual congestion. They employed the use of Ns simulator, and results they obtain shows large differences in energy distribution in the traces with attack, as compared to traces without attacks, with a threshold of 0.01 their scheme is able to identify varieties of attack types.[4]

III. Design of Proposed Architecture

Designing neural network model follows a number of systematic procedures. In general there are five basic steps shown in figure 2. The functionalities of this step are briefly described below.

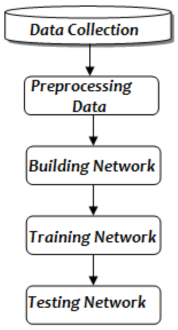


Figure 2 Basic flow of designing artificial neural network model

1. ***Data Collection*****:** First raw data is collected from trusted source to process for further steps. Data is collected according to the demand of the proposed network and dataset format should be compatible with the supported data format of the proposed network .Otherwise data set cannot be processed by the network.

2. ***Processing Data*****:** This block takes the original data from the collected data source, extracts the required features, and converts the data set into Matlab compatible format. This basically performs the data cleaning procedure.

3. ***Building Network:*** After the collection of desired data we have to create the proposed network model with the collected data set and some predefined useful parameter setting. The proposed network model parameters should be set by the user as per the required demand of the output feature.

4. ***Training Network:*** Training the network is a very important feature of DoS attack simulation because here we make the network to behave the way as we proposed to i.e. we predefine the proposed network behaviour as our requirement i.e. to train it with the user defined way for its desirable behaviour.

5. ***Testing Network:*** After the training the network is tested with some sample input data set to obtain the desired out put data set and accordingly if fluctuation arises then we can determine that DoS attack is happened in the network resulting in successful DoS attack identification.

IV. Simulation Environment and Experimental Results

To access the effectiveness of the proposed Denial of Service attacks simulation approach, the following simulation was performed. Pentium® Dual-Core CPU E5200 @ 2.50GHz, having 2.98 GB of RAM was used. The operating system used was Microsoft Windows XP Professional with Service Pack 3. Simulation was performed using Matlab 2010a version 7.9.0.529.
Figure 3 shows the experimental neural network setup for the proposed DoS attacks simulation model.

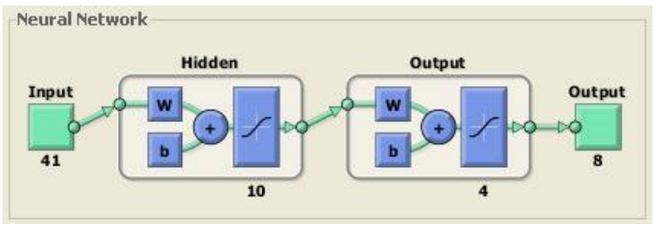


Figure 3 Experimental Neural Network Setup

The input layer of the experimental setup has 41 input neurons to describe 41 attributes in the KDD data set. For the hidden layer we used 12 numbers of neurons along with bias *b* and weight *w* that gave the optimal result as compared to using 20 and 10 neurons respectively for classifying the attacks. Since the network traffic is grouped into eight different classes of attacks, we have taken eight numbers of neurons in the output layer.

After the data was cleaned, pre-processed and encoded it was fed to the neural network model of DDoS. The total time taken for completing the simulation was 3 hrs 2 min. approx. It took 243 iterations to train the particular network.

KDD 99 data set based on the DARPA intrusion detection programme, which is publicly accessible via

MIT Lincoln Lab is first of all collected at this block. This block takes the original data from the MIT Lincoln Lab, extracts the required features, and converts the data set into Matlab compatible format. This basically performs the data cleaning procedure.

The attributes given in the data set are converted into double data type to make it compatible with the ANN Tool box of Matlab. The authors have converted the feature variables "protocol type" with values like tcp=0, udp=1, icmp=2; "error flag" with corresponding values S0=0, SF=1, S1=2, REJ=3, S2=4. The attacks in the data set have been categorized as normal=0, smurf=1, Neptune=2, bacl=3, teardrop=4, pod=5, land=6, snmpgetattack=7, saint=8, isweep=9, portsweep=10 nmap=11, warezclient=12, geoso-pasud=13, wareznader=14, ftpurele=15, and so on.

The Neural network is trained with the given data set taken as source and target. Training parameters are taken as follows

1. Epochs : 1000
2. Goal : 0.000000000000001
3. Max fail : 50

The developed neural network model is tested with the predefined input and output data set to determine fluctuation and consequently the kind of attack.

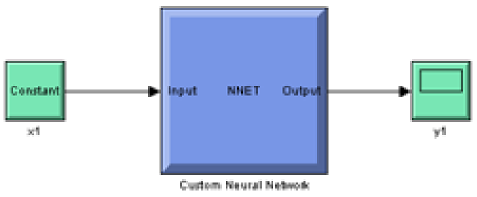


Figure 4 Structure of proposed neural network model

The proposed model is tested with the predefined source and target dataset to obtain desired or fluctuation in result and also to determine the kind of predefined DoS attack.

Figure 5 shows the performance of the system taking into account training, validation and level 1 testing data. It shows that the best validation performance was 1.4138e-007 at epoch 33.

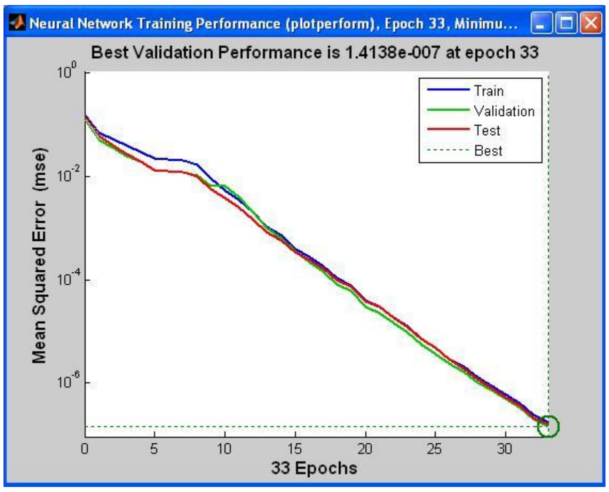


Figure 5 System performance of the during training, validation and testing

Figure 6 shows the neural network training state plot. It also shows validation check at epoch 33 and highlights that there is one validation failure at epoch 10.

The graphical representation of the visual impression of the distribution of Errors (Targets- Outputs) is shown as the Error Histogram plot for the given data in figure 7. It consists of tabular instances shown as adjacent rectangles erected over discrete bins. It shows that maximum error ~ -5e-005

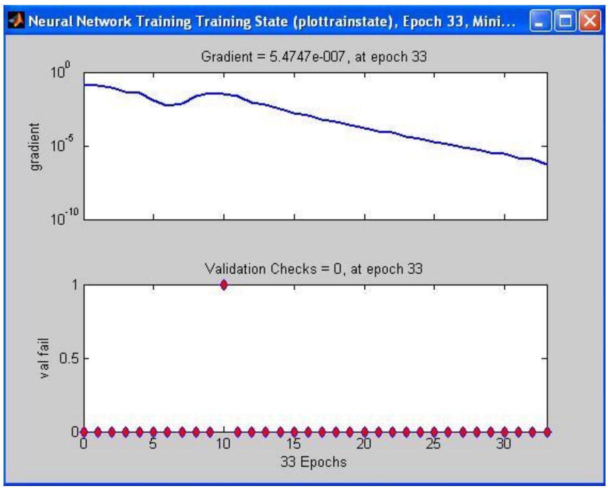


Figure 6 Neural Network Training State plots

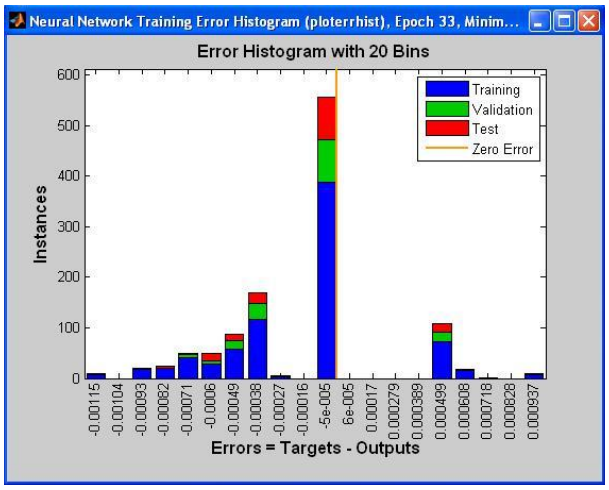


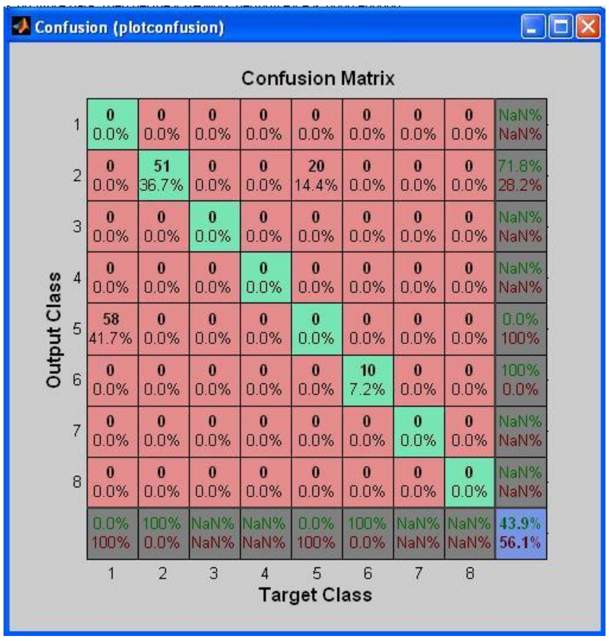
Figure 7 Neural Network Error Histogram plot

Figure 8 shows the Confusion Matrix i.e. Success Rate vs. Error Rate in all the stages like training, validation and level 1 testing. The results are enumerated in TABLE I.

After the network was trained it was tested for level 2 testing with a new set of data. The dataset taken was of 100 plus records. The new confusion matrix showed a success rate of 54.5% and error rate of 45.5%. Then we tested for Level 3 with a new set of data. The new confusion matrix showed a success rate of 100%. We can keep on testing to level 4 and there is always a possibility the success rate may decrease depending upon the data taken. The receiver operating characteristics (ROC) was found to be as expected consisting mostly of true positives and fewer numbers of false positives. Figure 8, figure 9 and figure 10 show the details of confusion matrix and receiver operating characteristics respectively. Table I includes the success and error rates of level 2 and level 3 testing as well. It shows a higher error rate and lower success rate as compared to the level 1 testing phase due to the presence of unclassified attacks outside the level 1 training environment. Increasing the training phase with more training data set may eventually lessen the error rate.

Table I Success and failure rates of all the stages

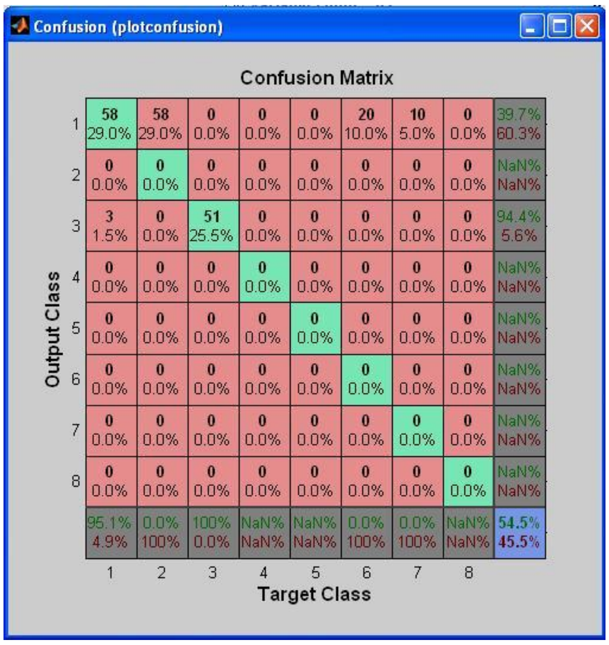
Phase	Success Rate	Failure Rate
Training	85.6%	14.4%
Validation	85.6%	14.4%
Level 1 Testing	43.9%	56.1%
Level 2 Testing	54.5%	45.5%
Level 3 Testing *	100.0%	0.0%


Confusion (plotconfusion)

Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	
1	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
2	0 0.0%	51 36.7%	0 0.0%	0 0.0%	20 14.4%	0 0.0%	0 0.0%	0 0.0%	71.8% 28.2%
3	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
5	58 41.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0.0% 100%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	10 7.2%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
	0.0% 100%	100% 0.0%	NaN% NaN%	NaN% NaN%	0.0% 100%	100% 0.0%	NaN% NaN%	NaN% NaN%	43.9% 56.1%

Figure 8 Confusion Matrixes for Level 1 Testing


Confusion (plotconfusion)

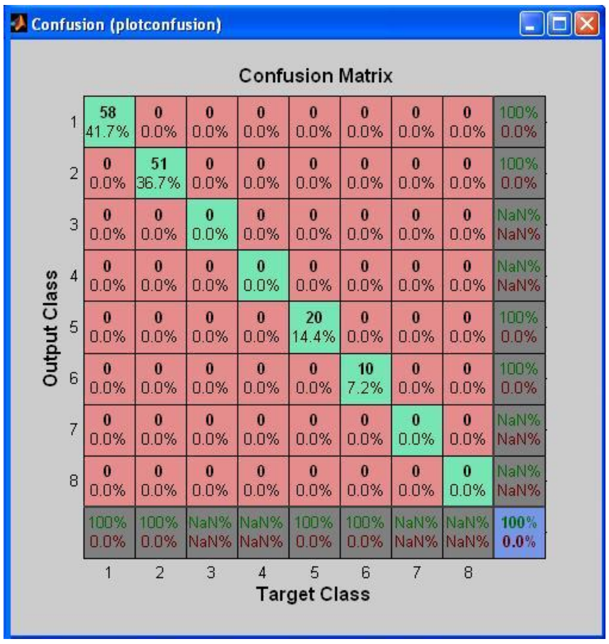
Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	
1	58 29.0%	58 29.0%	0 0.0%	0 0.0%	0 0.0%	20 10.0%	10 5.0%	0 0.0%	39.7% 60.3%
2	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
3	3 1.5%	0 0.0%	51 25.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.4% 5.6%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
	95.1% 4.9%	0.0% 100%	100% 0.0%	NaN% NaN%	NaN% NaN%	0.0% 100%	0.0% 100%	NaN% NaN%	54.5% 45.5%

Figure 9 Confusion Matrixes for Level 2 Testing

V. Conclusion

We have verified the proposed model using artificial neural network in matlab environment. The future scope will be to implement the proposed model in Spartan-3E FPGA and validate the performance with the simulation result. We have already simulated the same using iSim in Xilinx and our future work remains to download to the Spartan Kit



	1	2	3	4	5	6	7	8	
1	58 41.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	51 36.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 14.4%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	10 7.2%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
	100% 0.0%	100% 0.0%	NaN% NaN%	NaN% NaN%	100% 0.0%	100% 0.0%	NaN% NaN%	NaN% NaN%	100% 0.0%

Figure 10 Confusion Matrix for Level 3 Testing

.

References

[1]. Don Holden, "A Rule Based Intrusion Detection System", published in the proceeding of the IFIP TC11. ISBN: 0-444-89699-6

[2]. I. Kotenko and A. Ulanov, "Simulation of internet DDoS attacks and defense," in Information Security, pp. 327–342, Springer, 2006.

[3]. I. Kotenko, A. Alexeev, and E. Man'kov, "Formal framework for modeling and simulation of DDoS attacks based on teamwork of hackers-agents," in Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on, pp. 507–510, IEEE, 2003.

[4]. L. Li and G. Lee, "DDoS attack detection and wavelets," Telecommunication Systems, vol. 28, no. 3-4, pp. 435–451, 2005.

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

[6]. H. Debar, M. Becke, D. Siboni, "A Neural Network Component for an Intrusion Detection System, " IEEE Symposium on Research in Security and Privacy, 1992.

[7]. J. Ryan, M. Lin, R. Mikkulainen, "Intrusion Detection with Neural Networks," Advances in Neural Information Processing Systems, vol. 10, 1998, MIT Press.

[8]. R. Lippmann, R. Cunningham, "Improving Intrusion Detection performance using Keyword selection and Neural Networks," RAID Proceedings, West Lafayette, Indiana, Sept 1999.

[9]. A. Ghosh, A. Schwartzbard, "A study in using Neural Networks for Anomaly and Misuse Detection, " Proceedings of the 8th USENIX Security Symposium, 1999.

[10]. J. Cannady, "Artificial Neural Networks for Misuse Detection," Proceedings of the 1998 National Information Systems Security Conference (NISSC'98), 1998.

[11]. Z. Zhang, J. Li, C.N. Manikopoulos, J. Jorgenson, J. Ucles, "Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification" IEEE Workshop on Information Assurance and Security,2001 , pp. 85–90.

[12]. I Mukhopadhyay et.al. "Hawk Eye Solutions: Expectation Maximization based Network Intrusion Detection System", IJCA Proceedings on International Conference and workshop on Emerging Trends in Technology (ICWET) (1):22-29, 2011. Published by Foundation of Computer Science. ISBN: 978-93-80747-65-3. http://www.ijcaonline.org/proceedings/icwet/num ber1 /2059-aca33 March 2011.

[13]. I Mukhopadhyay et. al., "Back Propagation Neural Network Approach to Intrusion Detection System", published in the proceeding of International Conference on Recent Trend in Information System (ReTIS-11), held between 21st to 23rd December 2011 at Jadavpur University, Kolkata. Page No: 303 – 308 DoI: 10.1109/ReTIS. 2011.6146886

[14]. Pantos, "Packet Reading with libpcap", April 2010 http://www.systhread.net/texts/200805lpcap1.php

[15]. KDD Cup 1999. Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, October 2007

[16]. M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[17]. Knowledge discovery in databases DARPA archive. TaskDescription.http://www.kdd.ics.uci.edu/databases/kddcup99/task.htm