IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 9, April 2016)

Intrusion Detection using ABC optimized Neural Network

Vaishali Anwekar

Khushboo Pawar

Abstract –**The primary aim of intrusion detection is to detect attacks against a computer system. It is an important technology in business sector as well as an active area of research. In Information Security, intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource. It plays a very important role in attack detection, security check and network inspection. This paper presents the performance of Neural Network for various values of number of clusters, based on experiments. The optimization of output is done using Artificial Bee Colony (ABC) algorithm by selecting initial through ABC.**

Keywords – **Artificial Bee Colony Algorithm, Intrusion Detection, Neural Network, and KDD cup'99.**

I. INTRODUCTION

Intrusion detection systems have evolved from monolithic batch-oriented systems to distributed real-time networks of components as shown in Figure 1.

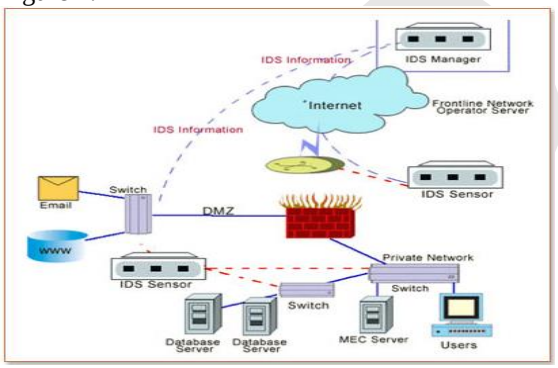

Figure 1: Intrusion Detection Network Structure [1]

In current systems, a number of common building functional blocks can be distinguished:

- Sensor, Probe: These modules form the most primitive data-gathering components of IDS. Implemented in a highly system-specific fashion, they track network traffic, log files or system behaviour - translating raw data into events useable by the IDS monitors.
- Monitor: Monitor components, the main processing segment of IDS, receive events from sensors. These events are then correlated against the IDS behaviour models, potentially producing model updates and alerts. Alerts, events in themselves, indicate occurrences significant to the security of a system, and may be forwarded to higher-level monitors, or to resolver units.
- Resolver: Resolver components receive suspicion reports from monitors (in the form of one or more alerts), and determine the appropriate response - logging, changing the behaviour of lower level components, reconfiguring other security mechanisms (e.g. adding firewall rules), and notifying operators.
- Controller: Facilitating component configuration and coordination, controller components are most significant in distributed ID systems, where manually upgrading, configuring and starting a network-wide series of components would be infeasible. In addition, controller units offer a single point of administration and interrogation for IDS, and may act in a supervisory capacity, restarting failed components.

II. MOTIVATION AND OBJECTIVE

Intrusion detection systems (IDSs) are designed to discover malicious activities that attempt to compromise the confidentiality, integrity and assurance of computer systems. Unlike a firewall that filters "bad" traffic, an IDS analyses packets to detect malicious attack attempts.

Intrusion detection systems have become critical components in network security. Therefore, two factors need to be considered to ensure IDS effectively. First, the IDS should deliver reliable detection results. The detection method should be effective in discovering intrusions since poor detection performance ruins the trustworthiness of the IDS. Second, the IDS should be able to survive in hostile environments or even under attack. However, it is challenging for IDSs to maintain high detection accuracy. An IDS that uses attack signatures to detect intrusions cannot discover novel attacks. As the number of new intrusions increases, these IDSs are becoming incapable of protecting computers and applications. Therefore, a detection approach that is able to discover new attacks is necessary for building reliable IDSs

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 9, April 2016)

A computer system should provide confidentiality, integrity and assurance against denial of service. However, due to increased load and connectivity more and more system is subject to attack by intruders. These attempts try to exploit flaws in the operating system as well as in the application programs. In fact, it is not possible to build a complete secure system. It can have cryptographic methods but they have their own problems as passwords can easily be cracked, users can lose their passwords and entire crypto-system can be broken. Even a truly secure system is vulnerable to abuse by insiders who abuse their privileges. Also, we need a balance between access control and user efficiency as stricter the mechanism, the lower the efficiency. So we need a system which is real time i.e. we would like to detect them as soon as possible and take appropriate action. This is what an intrusion detection system does. It is reactive rather than proactive. This paper tries to build a system which created clusters from its input data by labeling clusters as normal or anomalous data instances and finally used these clusters to classify unseen network data instances as either normal or anomalous. Both training and testing was done using different subset of KDD Cup 99 data which is very popular and widely used intrusion attack dataset.

III. PROPOSED METHOD

Attacks in Data Set

Each connection was labelled as normal or as exactly one specific kind of attack. All labels are assumed to be correct. There were a total of 37 attack types in the data set. The simulated attacks fell in exactly one of the four categories, User to Root; Remote to Local; Denial of Service; and Probe.

Denial of Service (dos): Attacker tries to prevent legitimate users from using a service.
Remote to Local (r2l): Attacker does not have an account on the victim machine, hence tries to gain access.

User to Root (u2r): Attacker has local access to the victim machine and tries to gain super user privileges.

Probe: Attacker tries to gain information about the target host.
Out of these only following attacks are covered in training and testing dataset which come under DOS attack.

Different types of DOS Attacked handled are:

Table 1: DOS Attacks

S. No	Attack	Category
1	Smurf	DOS
2	Neptune	DOS
3	Back	DOS
4	Teardrop	DOS
5	Pod	DOS

- Smurf Attack

The Smurf Attack is a way of generating significant computer network traffic on a victim network. This is a type of denial-of-service attack that floods a system via spoofed broadcast ping messages.
This attack relies on a perpetrator sending a large amount of ICMP (Internet Control Message Protocol) echo request (ping) traffic to IP broadcast addresses, all of which have a spoofed source IP address of the intended victim. If the routing device delivering traffic to those broadcast addresses delivers the IP broadcast to all hosts (for example via a layer 2 broadcast), most hosts on that IP network will take the ICMP echo request and reply to it with an echo reply, multiplying the traffic by the number of hosts responding. On a multi-access broadcast network, hundreds of machines might reply to each packet. According to CERT-CC (Computer emergency response team Coordination Center) the name Smurf comes from name of one of the exploit programs used to execute the attack.

In the late 1990s, many IP networks would participate in Smurf attacks (that is, they would respond to pings to broadcast addresses). Today, thanks largely to the ease with which administrators can make a network immune to this abuse, very few networks remain vulnerable to Smurf attacks.
The fix is two-fold:

1. Configure individual hosts and routers not to respond to ping requests or broadcasts.
2. Configure routers not to forward packets directed to broadcast addresses. Until 1999, standards required routers to forward such packets by default, but, in that year, the standard was changed to require the default to be not to forward.

Another proposed solution is network ingress filtering which rejects the attacking packets on the basis of the forged source address.

- Neptune attack

It is two typical threats to web servers:

- Teardrop attack

It involves sending mangled IP fragments with overlapping, over-sized payloads to the target

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 9, April 2016)

machine. This can crash various operating systems due to a bug in their TCP/IP fragmentation re-assembly code. Windows 3.1x, Windows 95 and Windows NT operating systems, as well as versions of Linux prior to versions 2.0.32 and 2.1.63 are vulnerable to this attack.

- A ping of death (POD)

POD is a type of attack on a computer that involves sending a ping of death (POD): malformed or otherwise malicious ping to a computer. A ping is normally 56 bytes in size (or 84 bytes when IP header is considered); historically, many computer systems could not handle a ping packet larger than the maximum IP packet size, which is 65,535 bytes. Sending a ping of this size could crash the target computer. In early implementations of TCP/IP, this bug was easy to exploit. This exploit has affected a wide variety of systems, including UNIX, Linux, Mac, Windows, printers, and routers.

However, most systems since 1997–1998 have been fixed, so this bug is mostly historical. Generally, sending a 65,536-byte ping packet would violate the Internet Protocol as written in RFC 791, but a packet of such a size can be sent if it is fragmented; when the target computer reassembles the packet, a buffer overflow can occur, which often causes a system crash. In recent years a different kind of ping attack has become widespread ping flooding simply floods the victim with so much ping traffic that normal traffic fails to reach the system.

- Back Attack

It involves sending forged requests of some type to a very large number of computers that will reply to the requests. Using Internet protocol spoofing, the source address is set to that of the targeted victim, which means all the replies will go to (and flood) the target.

In the International Knowledge Discovery and Data Mining Tools Competition, only "10% KDD" dataset is employed for the purpose of training. This dataset contains 22 attack types and out of which DOS attack dataset collected. It contains more examples of attacks than normal connections and the attack types are not represented equally.

Because of their nature, denial of service attacks account for the majority of the dataset. On the other hand the "Corrected KDD" dataset provides a dataset with different statistical distributions than either "10% KDD" or "Whole KDD" and contains 14 additional attacks. The list of class labels and their corresponding categories for "10% KDD" are detailed in table out of which dataset for DOS attack is prepared.

Table 2: Class labels that appears in "10% KDD" dataset

Attack	Class Labels	Category
Smurf	280790	Dos
Neptune	107201	Dos
Back	2203	Dos
Teardrop	979	Dos
Pod	264	Dos
Land	21	Dos
Normal	97277	normal
Satan	1589	probe
Ipsweep	1247	probe
Portsweep	1040	probe
Nmap	231	probe
Warezclient	1020	r21
Guess_passwd	53	r21
Warezmaster	20	r21
Imap	12	r21
ftp_write	8	r21
Multihop.	7	r21
Phf	4	r21
Spy	2	r21
Bitter_overflow	30	u2r
Rootkit	10	u2r
Loadmodule	9	u2r
Perl	3	u2r

Data pre-processing

Data pre-processing comprises following components including document conversion, feature selection and feature weighting. The functionality of each component is described as follows:

1. Dataset prepared with DOS attack which include smurf, Neptune, back, teardrop and POD ping of death attacks / anomaly.
2. Feature selection – reduces the dimensionality of the data space by removing irrelevant or less relevant feature selection criterion.
3. Document conversion- converts different types of documents such as GZ, TCPDUMP to CSV (Comma-separated values) file and ARFF (Attribute-Relation File Format) data file format.

Artificial Bee Colony (ABC) optimized Neural Network approach is used to determine optimum number of clusters in analysed data.

Artificial Bee Colony Algorithm

The ABC algorithm is a swarm based, meta-heuristic method based on the foraging behaviour of honey bee colonies. The model is composed of three important elements: employed and unemployed

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 9, April 2016)

foragers, and food sources. The employed and unemployed foragers are the first two elements, while the third element is the rich food sources close to their hive. The two leading modes of behaviour are also described by the model. These behaviours are necessary for self -organization and collective intelligence: recruitment of forager bees to rich food sources, resulting in positive feedback and simultaneously, the abandonment of poor sources by foragers, which causes negative feedback.

Pseudo Code of the ABC Algorithm

1. Initialize the population of solutions x_{ij}
2. Evaluate the population
3. Cycle=1
4. Repeat
5. Produce new solutions (food source positions) v_{ij} in the neighbourhood of x_{ij} for the employed bees and evaluate them.
6. Put on the greedy selection process between xi and v_i
7. Compute the probability values Pi for the solutions xi by means of their fitness values. In order to calculate the fitness values of solutions

$$\begin{bmatrix} \frac{1}{1+f_i} & if\ f_i \geq 0 \\ 1 + abs(f_i) & if\ f_i < 0 \end{bmatrix} \quad (1)$$

   Normalize p_i values into [0, 1]
8. Produce the new solutions (new positions) v_i for the onlookers from the solutionsx_i, selected depending onp_i, and evaluate them
9. Put on the greedy selection process for the onlookers between x_i and v_i
10. Determine the abandoned solution (source), if exists, and replace it with a new randomly produced solution xi for the scout using the equation
    $x_{ij} = min_j + rand(0,1) * (max_j - min_j)$ (4.2)
11. Memorize the best food source position (solution) achieved so far
12. cycle=cycle+1
13. until cycle= Maximum Cycle Number (MCN)

Neural Network

The Neural Network is a three layer paradigm that entertains the input and processes it to generate output.

Neurons

A class of statistical models is generally recognized as "Neural" if they possess following characteristics:

a. Consisting the set of adaptive weights or numerical parameters tuned by learning algorithm
b. Capable of approximating non-linear functions of their inputs.

The training mode of neurons (Figure 2) sources n number of inputs in parallel manner. The teaching input provides sufficient arguments to a neuron for generating specific output. In this mode the neurons are trained to fire for particular fashion. In case if input pattern does not resemble the taught list of patterns, firing rules takes the decision of firing or holding the inputs.

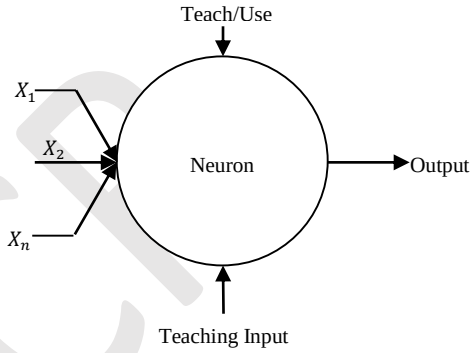

Figure 2: A simple Neuron [13]

Layers

The organization of Neural Network is a pattern of interconnected nodes that contain 'activation function'. The patterns are provided to 'input layer' that communicates with one or more hidden layers (Figure 3) for processing via weighted connections.

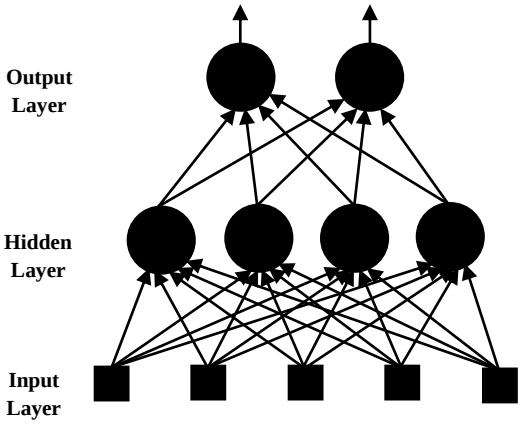

Figure 3: An example of a simple feed-forward network [13]

The connection among the hidden layers and output layer delivers the required answers. The functions of three layers in NN can be defined in following statements:

- ✓ *Input Layer:* This layer accepts the patterns as input. The firing decision for further

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 9, April 2016)

process is positive if the test input match with the training input.

- ✓ *Hidden Layer:* Dependent on the weights of connection with input and the related activities the operations at hidden layer are determined.
- ✓ *Output Layer:* The behaviour is directly proportional to hidden layer(s) activity and weight of connections among both output and hidden layer(s).

Architecture: Feed Forward Neural Network

The signals only travel only single way i.e. from input to output. No feedback loops are initialized at any layer and the output does not affect the same layer in any manner. Most of pattern recognition experiments use this architecture.

Learning Process: Back Propagation

Back Propagation Neural Network (BPNN) generates complex decision boundaries in feature space. BPNN in specific circumstances resembles Bayesian Posterior Probabilities at its output. These conditions are essential to achieve low error performance for given set of features along with selection of parameters such as training samples, hidden layer nodes and learning rate. In else case, the performance of BPNN could not be evaluated. For W number of weights and N number of nodes, numbers of samples (m) are depicted to correctly classify future samples in following manner:

$$m \geq O\left(\frac{W}{\epsilon}\log\frac{N}{\epsilon}\right) \quad (1)$$

The theoretical computation of number of hidden nodes is not a specific process for hidden layers. Testing method is commonly entertained for selection of these followed in the constrained environment of performance [13].

IV. Simulation And Results

The performance of proposed algorithms has been studied by means of MATLAB simulation.

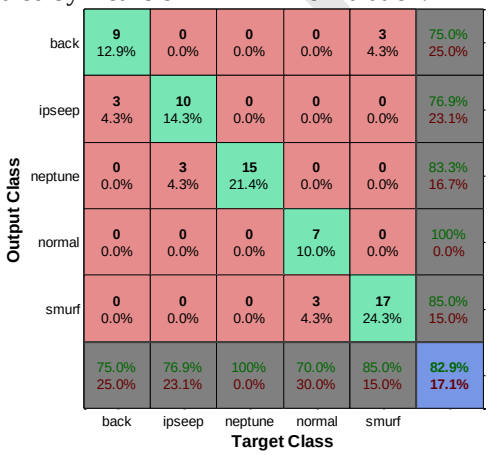

Figure 4: Confusion Matrix plot for intrusion classifier scheme using Neural Network

The row and column are the class of Intrusion in KDD99 cup database. There are 5 set of classes and each class having different set of detection. For training only 30 samples is taken from every set. The confusion plot indicates the accuracy i.e. 82.9% for proposed algorithm.

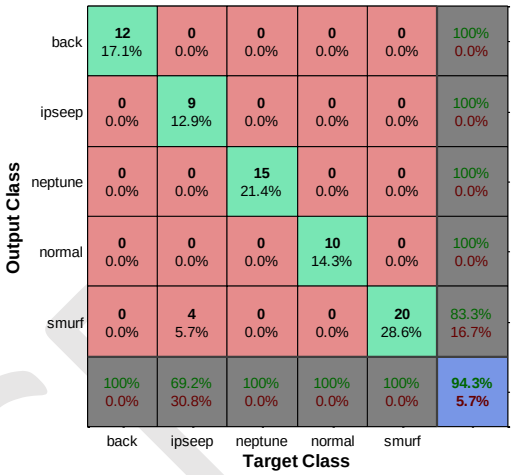

Figure 5: Confusion Matrix plot for intrusion classifier scheme using ABC-Neural Network

The row and column are the class of Intrusion in KDD99 cup database. There are 5 set of classes and each class having different set of detection. For training only 30 samples is taken from every set. The confusion plot indicates the accuracy i.e. 94.3% for proposed algorithm.

V. Conclusion

In this paper, an algorithm based on the Artificial Bee Colony and Neural Network for analysing program behaviour in intrusion detection is evaluated by experiments. The preliminary experiments with the 1999 KDD cup'99 database have shown that this approach is able to effectively detect intrusive program behaviour. The results also shown that a low false positive rate can be achieved. With the frequency-weighting method where each entry is equal to the number of occurrences of a system call during the TCP (Transmission Control Protocol) communication. Neural Network training process could be easily added to the training data set without changing the weights of the existing training samples. Artificial Bee Colony algorithm is used to optimize the output of our system, by appropriate selecting the input parameters through ABC.

Reference

[1] David Moore, "The Spread of the Code-Red Worm (Crv2)", September 2001. Online available at: http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 9, April 2016)

[2] Nicholas C Weaver, "Warhol Worms: The Potential for Very Fast Internet Plagues", August 2001. Online available at: http://www.cs.berkeley.edu/~nweaver/warhol.html
[3] Venkata Suneetha Takkellapati, G.V.S.N.R.V Prasad, "Network Intrusion Detection system based on Feature Selection and Triangle area Support Vector Machine", International Journal of Engineering Trends and Technology- Volume3, Issue 4, 2012.
[4] Esh Narayan, Pankaj Singh and Gaurav Kumar Tak, "Intrusion Detection System Using Fuzzy C-Means Clustering with Unsupervised Learning via EM Algorithms" VSRD-IJCSIT, Vol. 2 (6), 502-510, 2012.
[5] Deepika Dave, Prof. Vineet Richhariya, "Intrusion detection with KNN classification and DS- theory", IRACST Vol. 2, No.2, April 2012.
[6] P.S. Prabhu, "Network Intrusion Detection Using Enhanced Adaboost Algorithm", International Journal of Communications and Engineering Volume 3, No.3, Issue: 02 March 2012.
[7] Dalila BOUGHACI, Mohamed Lamine HERKAT, Mohamed Amine LAZZAZI, "A Specific Fuzzy Genetic Algorithm for Intrusion Detection", ICCIT, 2012.
[8] R. Shanmugavadivu, Dr.N.Nagarajan, "Network Intrusion Detection System Using Fuzzy Logic" IJCSE Vol. 2 No. 1, 2011.
[9] Nasser S. Abouzakhar And Abu Bakar, "A Chi-Square Testing-Based Intrusion Detection Model",CFET, 2010.
[10] Debdutta Barman Roy, Rituparna Chaki, Nabendu Chaki, "A New Cluster-Based Wormhole Intrusion Detection Algorithm for Mobile Ad-Hoc Networks", IJNSA, Vol 1, No 1, April 2009.
[11] Dianbo Jiang, Yahui Yang, Min Xia, "Research on Intrusion Detection Based on an Improved SOM Neural Network", IEEE 2009.
[12] Hazem M. El-Bakry, Nikos Mastorakis, "A Real-Time Intrusion Detection Algorithm for Network Security", Wseas Transactions on Communications Issue 12, Volume 7, December 2008.
[13] Christos Stergiou and Dimitrios Siganos, "Neural Networks", Report available at: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html