VHDL Modeling of Intrusion Detection & Prevention System (IDPS) – A Neural Network Approach

Tanusree Chatterjee  
Department of Computer Science  
Regent Education and Research Foundation

Abstract- The rapid development and expansion of World Wide Web and network systems have changed the computing world in the last decade and also equipped the intruders and hackers with new facilities for their destructive purposes. The cost of temporary or permanent damages caused by unauthorized access of the intruders to computer systems has urged different organizations to increasingly implement various systems to monitor data flow in their network. The systems are generally known as Intrusion Detection System (IDS). Our objective is to implement an artificial network approach to the design of intrusion detection and prevention system and finally convert the designed model to a VHDL (Very High Speed Integrated Circuit Hardware Descriptive Language) code. This feature enables the system to suggest proper actions against possible attacks. The promising results of the present study show the potential applicability of ANNs for developing practical IDSSs.

Keywords – Intrusion Detection System, artificial Neural network, attacks, Network Security

I. INTRODUCTION

An IDS can be a piece of installed software or a physical appliance that monitors network traffic in order to detect unwanted activity and events such as illegal and malicious traffic, traffic that violates security policy and that violates acceptable use policies. Intrusion Detection Systems (IDS) are now mainly employed to secure company networks. With the explosive growth of the network systems, information exchange became routine between computers around the world, thus the need for network security has become even more critical with the rise of information technology in everyday life. Meanwhile, the complexity of attacks is on the rise regardless of the beefed up security measures. Intrusion Prevention System (IPS) provides an in-line mechanism focus on identifying and blocking malicious network activity in real time. An IPS is a type of IDS that can prevent or stop unwanted traffic. The IPS usually logs such events and related information.

The IPS is unique in its ability to gather evidence of an attacker’s activity, remove the attacker’s access to the network, and reconfigure the network to resist the attacker’s penetration technique. The IPS stops attacks at the source of the threat and can proactively protect against future threats and vulnerabilities. In our present study, an intrusion detection system is implemented using Multi Layer Perceptron (MLP) ANN. The universal approximation theorem states that an MLP (with one or more hidden layers) can approximate any function with arbitrary precision and of course the price is an increase in the number of neurons in the hidden layer. Using more than one layer may lead to more efficient approximation or to achieving the same accuracy with fewer neurons in the neural network.

One of the objectives of the present study is to evaluate the possibility of achieving the same results with this less complicated neural network structure. Using a less complicated neural network is more computationally efficient. Also it would decrease the training time. While in some previous studies the implemented system is a neural network system with the capability of detecting normal or attack connections, in our present study a more general problem is considered in which the attack type is also detected. This feature enables the system to suggest proper actions against possible attacks. The promising results of the present study show the potential applicability of ANNs for developing practical IDSSs.

KDD 99 dataset are used as the input vectors for training and validation of our neural network. It was created based on the DARPA (Defense Advanced Research Project Agency) intrusion detection evaluation program which is publicly accessible via MIT Lincoln Lab[12][13].

II. RELATED WORK

There have been lots of traditional rule-based works to the design of IDS. Recently, an increasing amount of research has been conducted on applying artificial neural networks to detect intrusions in the network [3] [4]. This method proves to be advantageous as it goes through rigorous training, validation and testing phases before being actually fed to the network for detecting attacks. Comparing different neural network classifiers, the back-propagation neural network (BPN) has showed to be more efficient in developing IDS. In recent years, various approaches on Biological Inspired Intrusion Prevention and Self-healing System for network security have been proposed. These approaches are based upon data inspired by the human immune system (HIS) which applied to the autonomous defense system. The system integrates an artificial immune intrusion prevention system for network security inspired by the immunology theory known as danger theory and adaptive immune system. Attackers take the advantage of CGI (Common Gateway Interface) scripts to perform an attack by sending illegitimate inputs to the web server. A thesis in recent year also contains the findings and the results of the CGI related web server attacks and an explanation of the design and implementation of Intrusion Detection and Prevention System for CGI based web server attacks. In recent years, multilayer intrusion detection and prevention system architecture for VoIP infrastructures has been proposed. The key components of the approach are based on a VoIP-specific honeycomb network.
pot and on an application layer event correlation engine [5]. Some research in recent years has been done for IDPS using object oriented analysis method which describes the state’s overall requirements regarding the acquisition and implementation of IDPS with intelligence.

III. PROCEDURAL DETAILS

We have started our work towards designing an IDPS using Neural Network in MATLAB 2010b. We have used all 41 attributes of the KDD 99 dataset to train the ANN and then supplying it with other data sets which are yet to be classified to what type of attack it is. Data being used are live data from MIT Labs.

We started with the following activities before going to build the SIMULINK model of IDPS.
- Data collection
- Data Cleaning and Data Pre-Processing
- Classification of different attacks
- Data Feeding to ANN
- Use of basic nprttool
- Simulating Results.
- Observing Results.

Data Collection
KDD 99 data set based on the DARPA intrusion detection program, which is publicly accessible via MIT Lincoln Lab, is first of all collected at this block. We started our work with 5000 data of the total dataset, after that 25000 and then we proceeded with 1,00000 and 3,00000 dataset.

Data Cleaning and Data Pre-Processing
After collecting the original data from the MIT Lincoln Lab, we extracted the required features, and converted the data set into Matlab compatible format. This basically performs the data cleaning and pre processing procedure. First we clear the total dataset – we have numbered each element of the all 42 attributes. Then we converted all the elements to double and classified all the attacks according to the number resided in 42 th column.

The attributes given in the data set are converted into double data type to make it compatible with the ANN Tool box of Matlab. Then the feature variables have been converted e.g. the “protocol type” with values like tcp=0, udp=1, icmp=2. The attacks in the data set have been categorized as DOS=1, normal=0, probe=2, r2l=3, u2r=4.

Previous work has been done in the field of detection using limited number of attributes of KDD Dataset 99. We have used all 41 attributes (shown in figure 1.1) of the dataset to train the ANN and then supply it with other unclassified data sets which are yet to be classified.

---

### Table of Features

| Features          | Description                                                                 | Type     | Features          | Description                                                                 | Type     |
|-------------------|-----------------------------------------------------------------------------|----------|-------------------|-----------------------------------------------------------------------------|----------|
| 1. duration       | Duration of the connection                                                  | Cont.    | 11. hour          | 1 if the hour is in a 'poor' hour, 0 otherwise                            |          |
| 2. protocol type  | Connection protocol (e.g. tcp, udp)                                        | Dist.    | 12. Count         | number of connections to the same host in the current connection in the past two seconds | Cont.    |
| 3. service        | Destination service (e.g. smtp, ftp)                                       | Dist.    | 13. size count     | number of connections to the same service as the current connection in the past two seconds | Cont.    |
| 4. flag           | Flags of the connection                                                    | Dist.    | 14. err count      | % of connections that have SYST Error                                      | Cont.    |
| 5. source bytes   | Bytes sent from source to destination                                      | Cont.    | 15. err source rate| % of connections that have SYST Error                                      | Cont.    |
| 6. destination    | Bytes sent from destination to source                                      | Cont.    | 16. cur err source rate| % of connections that have REE Error                                      | Cont.    |
| 7. load           | If connection is from the same host, if not                                | Dist.    | 17. err            | % of connections that have REE Error                                      | Cont.    |
| 8. wrong fragments| Number of wrong fragmentation                                              | Cont.    | 18. err rate       | % of connections to the same service                                         | Cont.    |
| 9. avg            | Average of input packets                                                   | Cont.    | 19. err            | % of connections to different services                                      | Cont.    |
| 10. std           | Standard deviation of input packets                                        | Cont.    | 20. err            | % of connections to different hosts                                         | Cont.    |
| 11. failed login   | Number of failed logins                                                    | Cont.    | 21. err            | count of connections having the same destination host                       | Cont.    |
| 12. logged in     | 1 if successfully logged in, 0 otherwise                                    | Cont.    | 22. err            | count of connections having the same destination host                       | Cont.    |
| 13. is compromised | Number of "compromised" conditions                                         | Cont.    | 23. err            | % of connections having the same destination host                           | Cont.    |
| 14. root shell    | 1 if root shell in obtained, 0 otherwise                                    | Cont.    | 24. err            | % of different services on the current host                                | Cont.    |
| 15. in attempted  | 1 if "in" command attempted                                               | Cont.    | 25. err            | % of connections to the current host that have an 16 error                  | Cont.    |
| 16. rev          | Rev of the "in" command attempted                                          | Cont.    | 26. err            | % of connections to the current host that have an 16 error                  | Cont.    |
| 17. if successful | 1 if successful login                                                       | Cont.    | 27. err            | % of connections to the current host that have an 16 error                  | Cont.    |
| 18. port          | Port of the connection                                                     | Cont.    | 28. err            | % of connections to the current host that have an 16 error                  | Cont.    |
| 19. protocol      | Protocol number of the connection                                          | Cont.    | 29. err            | % of connections to the current host that have an 16 error                  | Cont.    |
| 20. service       | Service number of connection                                               | Cont.    | 30. err            | % of connections to the current host that have an 16 error                  | Cont.    |
| 21. is host login | 1 if the login belongs to the "host"                                      | Cont.    | 31. err            | % of connections to the current host that have an 16 error                  | Cont.    |

---

**Figure 1.1** All 41 features of KDDCUP’99 dataset

**Classification of different attacks**

To classify the different attacks we have found out the four main attack categories (Table 1.1):

- **a. DOS (DENIAL OF SERVICE):** An attacker tries to prevent legitimate users from using a service e.g. TCP SYN FLOOD, SMURF etc.
- **b. Probe:** An attacker tries to find information about the target host e.g. scanning the target host in order to get information about available resources.
- **c. 2R (User to Root):** An attacker has local account on victim’s host and tries to get the root’s privileges.
- **d. R2L (Remote to Local):** An attacker does not have local account on the victim’s machine and tries to obtain it.

And another category we have mentioned as **e. Other:** Those which are yet to be classified.

So we categorized the attacks in 5 categories and numbered them as – Dos (1), Probe (2), U2R (3), R2L (4) and other (5). We have numbered the normal data as 0.
Simulation was performed in MATLAB using dataset containing 311030 sample data was used as input to the IDS. Out of this 217720 samples were used for training, 46655 samples were used for validation and the rest 46655 samples were used for testing. The performance of the network taking into account training, validation and testing data in Matlab for the given dataset (using all the 41 features) is shown in the following figures. Total number of epochs taken is 243 where the validation check is 6, at epoch 243 and there is no validation failure up to this epoch (shown in figure 1.2).

We have obtained a least mean square error of $10^{-2}$ and the best validation performance was found to be 0.0088598 at epoch 237(Figure 1.3).

The next figure (figure 1.4) shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies. It shows the Success Rate vs. Error Rate in all the stages like training, validation and testing.

The following curves (Figure 1.5) represent the receiver operating curve. The colored lines in each axis represent the ROC curves.

| DoS       | U2R        | Probe     | R2L       | Other       |
|-----------|------------|-----------|-----------|-------------|
| Back      | Buffer     | Ipsweep   | FTP write | Named       |
| Land      | Load       | Nmap      | Guess passud | Xlock      |
| Neptune   | Perl       | Portswipe |           |             |
| Pod       | Satan      |           |           | Imap        |
| Smurf     | Teardrop   | Multihop  | Physpy    | Warezclient |
|           |            | Warezmaster|           |             |

Table 1.1 Different attacks of KDD dataset
The ROC curve is a plot of the true positive rate (sensitivity) versus the false positive rate (1 - specificity) as the threshold is varied. A perfect test would show points in the upper-left corner, with 100% sensitivity and 100% specificity. For this problem, the network performs very well.

Figure 1.5 Receiver operating characteristic graph – true positive vs. false positive

True and False alarms generated by IDS

- **True Positive**: A legitimate attack which triggers an IDS to produce an alarm.
- **False Positive**: An event signaling an IDS to produce an alarm when no attack has taken place.
- **False Negative**: A failure of an IDS to detect an actual attack.
- **True Negative**: When no attack has taken place and no alarm is raised.

The results are enumerated in the following table.

| Phase  | Success rate | Failure rate |
|--------|--------------|--------------|
| Training | 95.6%        | 4.4%         |
| Validation | 95.6%      | 4.4%         |
| Testing  | 95.6%        | 4.4%         |

Table 1.2 Success and failure rate of all stages

V. CONCLUSION AND FUTURE SCOPE

Our main objective is to convert the above design of the simulink model of the IDPS to VHDL code and downloaded next to FPGA (Sparatn 3 Starter Board). Now we are trying to convert the model of single layer ANN attack classifier model and next we will do the same using the multilayer structure described above. After converting the classifier model which will classify the normal or attack data separately, we will implement the VHDL of the total IDPS model which will not only detect the attack but also prevent them. Now we are working in offline scenario, after the successful completion of our work we will implement our architecture in online scenario.

REFERENCES

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

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

[3] Mehdi MORADI and Mohammad ZULKERNINE, "A Neural Network Based System for Intrusion Detection and Classification of Attacks", by Natural Sciences and Engineering Research Council of Canada (NSERC).

[4] Ray-I C hang et.al., "Intrusion Detection by Backpropagation Neural Networks with Sample-Query and Attribute-Query", International Journal of Computational Intelligence Research. ISSN 0973-1873 Vol.3, No. 1 (2007), pp. 6-10.

[5] Holistic VoIP Intrusion Detection and Prevention System by Mohamed Nassar, Saverio Niccolini, Radu State, Thilo Ewald.

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

[7] 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.

[8] “Guide to Intrusion and Prevention System”, csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf.

[9] Results of the DARPA 1998 Offline Intrusion Detection Evaluation, Richard P. Lippmann, Robert K. Cunningham, David J. Fried, Isaac Graf,Kris R. Kendall, Seth E. Webster, Marc A. Zissman.
[10] I. Mukhopadhyay, M. Chakraborty, S. Chakraborti, T. Chatterjee, “Back Propagation Neural Network Approach to Intrusion Detection System”, IEEE Conference on Recent Trend in Information System (ReTIS - 2011), ISBN No. 978-1-4577-0790-2.

[11] Artificial Neural Network Modeling of Intrusion Detection and Prevention System by M. Chakraborty, I. Mukhopadhyay, T. Chatterjee, IEMCON 2012, IEEE Conference on Innovative Techno Management Solutions for Social Sector.

[12] KDDCUP Data available on-
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, October 2007.

[13] Knowledge discovery in databases DARPA archive. Task Description.
http://www.kdd.ics.uci.edu/databases/kddcup99/task.html

[14] “Getting Started with Simulink - MathWorks”, www.mathworks.com/help/pdf_doc/simulink.pdf

[15] “Neural Network ToolBox - MathWorks”, www.mathworks.com/help/pdf_doc/nnet.pdf

[16] I Mukhopadhyay et.al, “HawkEye Solution” ICWET ’11 Proceedings of the International Conference & Workshop on Emerging Trends in Technology, pages 252-257, ISBN: 978-1-4503-0449-8.

AUTHORS PROFILE

First Author - Tanusree Chatterjee, M.Tech (CSE) from WBUT, Assistant Professor in Department of Computer Science, Regent Education and Research Foundation, Barrackpore, West Bengal.

Second Author – Abhishek Bhattacharya, M.Tech (CSE) from BIT, Mesra, Assistant Professor in Department of Computer Science, Institute of Engineering & Management, Saltlake, Kolkata, West Bengal.