Applied Neural Networks inIntrusion Detection System

Tung Trinh and Mario A. Garcia

College of Science & Engineering, Texas A&M University
Corpus Christi, Texas, USA

Abstract

Detecting attacks has been arising as the most important issue in security communities. In very large networks, it is impossible for administrators or security personnel to detect which computers are being attacked and from where attacks come. Hence, intrusion detection systems using neural networks are considered as the best solution to detect attacks. The reason is that neural networks have some advantages such as learning from training and being able to categorize data. This paper presents an implementation of an intrusion detection system using neural networks in .Net framework. This approach contains a data pre-processing module and a neural network. The neural network consists of one input layer, two hidden layers, and one output layer. The system is also tested in various architectures to compare the efficiency.

Introduction

The rapid growing of large computer networks is creating more and more opportunities for hackers to attack the networks. Besides, the popularity of intrusion tools allows more people to attempt a computer network. The raise in attacks alarmed network security communities to develop more secured solutions that could protect the tenets of information security: confidentiality, integrity, and availability [2]. In large networks which could contain more than 1000 computers, it is impossible for network administrators to figure out which computers are being attacked or to oppose attacks that are happening. Therefore, the intrusion detection system arises as the most efficient solution that can automatically detect attacks and then report attack information to network administrators. This system also monitors network traffic, identifies any unusual activities that can access the network without authorization and permission, and then notifies responsible people.

There are many proposed methods to develop an intrusion detection system; however, the neural network is considered a better approach among other approaches. The main advantage of neural networks is they can “acquire knowledge through learning and store it in inter-neuron connections known as synaptic weights” [3]. In other words, neural networks can detect attacks after they were trained with a sample database. In this paper, an intrusion detection system is implemented in .Net framework in various architectures to compare the efficiency of different neural networks.

The rest of this paper is organized as follows. Important definitions will be expressed in section 2 while section 3 discusses some related works. Section 4 presents the implementation of the intrusion detection system using neural networks. Section5 will describe the experiments and

results. In section 6, future works and conclusions are expressed.

An intrusion is defined as "an attempt to gain unauthorized accesses to network resources" [2]. An intrusion can be done by external people or internal users of a network. There are many types of intrusions are being used by hackers such as inference, viruses, Trojan, attempt break in, successful break in, and Denial-of-Service [5]. *An intrusion detection system* is a corporation of software and hardware components to perform three network defense functions: prevention, detection, and response. There are two main criteria to classify intrusion detection systems: the trigger and the source of data used by intrusion detection systems [5]. According to the source of data, intrusion detection systems can be categorized into three classifications: network-based intrusion detection systems, host-based intrusion detection systems, and hybrid intrusion detection system implementation [2]. Network-based intrusion detection systems identify attacks by analyzing all packets transmitting in the network. Instead of capture packets, host-based intrusion detection systems examine host properties and activities such as "system calls, application logs, and file system modifications (binaries, password files, capability/acl databases)" to detect intrusions [6]. The hybrid intrusion detection system implementation is the combination of the network-based intrusion detection system and the host-based intrusion detection system to take advantages and eliminate disadvantages of both.

A *neural network* is stated as "an information processing system that is inspired by the way biological nervous systems, such as the brain, process information" [7]. In other words, a neural network consists on a huge number of elements which work together to solve a given problem. In additional, a neural network also can be trained to gain experiences before being used. A neural network contains two main components: input layer and output layer. Depends on the complexity of the problem, a neural network can have one or more hidden layers between the input layer and output layer. The input layer gets input data while the output layer produces output data. The hidden layer plays a role of a data processing station. This layer handles data from the input layer and transfers processed data to the output layer. Neurons in a neural network are connected by the weights which are computed by using the activation function. There are three activation functions used in neural networks: linear, sigmoid, and hyperbolic tangent. Each activation functions scale data in different ranges.

The KDD Cup 1999 Data set

Intrusion detection systems have been receiving great attentions of computer science researchers in recent years. There are many approaches have been proposed and presented to network security communities. Instead of using real data, most of proposed approaches use either the DARPA 1998 data set or the KDD Cup 1999 data set or both as the input. The KDD Cup 1999 data set is a huge database that contains 14 different kinds of attacks and 42 features of each connection record [11a]. Each connection record contains a label that points out what type of that connection is. Based on the label, attacks are categorized. This data set is the most widely used for the intrusion detector learning task. More information about the KDD Cup 1999 data set is expressed in Appendix B.

In last few years, networking researchers have been developed intrusion detection systems using various neural network types. In [1], a feedforward neural network using the back propagation algorithm is developed with three layers: an input layer, a hidden layer, and an output layer. Similarly, Poojitha et al. describe an intrusion detection system using an artificial neural network

trained by back propagation algorithm in [5]. This proposed approach uses two phases, training and testing, to detect intrusion activities. Firstly, the intrusion detection system is trained to "capture the underlying relationship between the chosen inputs and outputs" [5]. After that, the system is tested with an available data set. Another work that applies the back propagation algorithm in intrusion detection system is presented in [8]. This approach detects intrusions in four steps: collect data, convert data into MATLAB format, convert data into double data type, and finally feed output data into the neural network. A combination of a back propagation neural network and the genetic algorithm is introduced in [9].

This intrusion detection system has eight modules including: a network packet capture device, the preprocessing module (a), the normal data detection module, the misuse detection module, a statistical module, the preprocessing module (b), the abnormal data detection module, and an alert response module. This approach is proposed to "overcome the blindness of optimization" and "avoid occurring local convergence". In [3], Jiang et al. introduce an intrusion detection system which is based on the improvement of the SOM algorithm. This approach can "increase detection rate and improve the stability of intrusion detection" by modifying the strategy of "winner-take-all" and using interaction weight which is the effect between each neuron in the output layer [3]. Han proposed an improved model of the Adaptive Resonance Theory 2-A neural network which can "handle data directly" in [10]. This implementation consists of three layers: F0, F1, and F2. The F0 layer takes input data and transfer to the layer F1 which "performs a Euclidean normalization" to filer only acceptable data to send to the F3 layer. The F3 layer then computes the activation value and labels the winning node as "normal" or "one of the 22 attack types" based on the classification of the data [10]. In [6], a host-based intrusion detection system using both anomaly detection and misuse detection trigger is implemented in neural networks with the SOM algorithm. Another proposed intrusion detection system is presented in [4]. This system uses resilient backpropagation algorithm to compute weights between neural neurons.

Since 2008, a neural network and machine learning framework named Encog has been published and developed for C/C++, Java and .NET by Heaton Research, Inc. [12]. This framework not only provides the library for creating neural networks but also normalizing and processing data. Everyone can download and use Encog for free for personal and noncommercial purposes. In this research, the Encog is used to build the neural network in .NET framework.

Implementation

Data normalization is a very important part in building a neural network because the neural network only process numeral data in some ranges. Unlikely, network traffic contains both numeric and alpha characters. Therefore, the need of normalizing network traffic data arises as one of the most challenging problem in neural network applications.The KDD Cup 1999 dataset is the most widely used database for training and testing neural networks. This dataset contains two different types of data: discrete and continuous. While continuous data consists of numeric values, discrete data may comprise alpha characters or boolean values (i.e., 1 for true and 0 for false). Hence, numeric and alpha characters in the KDD Cup 1999 dataset must be normalized. In order to normalize numeric data, the Equation 1 is used [1]:

$$f(x) = \frac{(x - d_L)(n_H - n_L)}{d_H - d_L} + n_L \quad (1)$$

where: x is the normalizing value, $d_L$is the lowest value of the dataset, d_H is the highest value of the dataset. $N_h$represents the highest value while n_L is the lowest value of the normalization range. Consequently, the highest and lowest value of each feature containing continuous data in the KDD Cup 1999 dataset must be found to perform the Equation (1). The approach firstly searches for highest and lowest values of each feature in the dataset. Then, when a particular value is normalized, the corresponding highest and lowest values are applied in the Equation (1). Table 1 shows some features with their according highest and lowest values.

Table **1**. Highest and lowest values of some features in KDD Cup 1999 dataset

Features	Highest Value	Lowest Value
duration	58329	0
wrong_fragment	3	0
urgent	14	0
hot	77	0
num_failed_logins	5	0

With discrete data, the solution is to collect every single value of each feature and then convert that value into a number in between -1 and 1. By doing this, the normalized values are consistent when different data files are used to train the neural network. Table 2 explains how some features like protocols and flag are normalized from discrete data type into numbers.

Table 2. Discrete data normalization.

Protocol		Flag	
Discrete data	Numeric data	Discrete data	Numeric data
udp	0	SF	0
tcp	0.5	S2	0.1
imcp	1	S1	0.2

The most important feature in the KDD Cup 1999 dataset is Label which denotes the category that each connection falls in. As mentioned, there are 23 types of network connection in the training data, one is normal connection and the others are attacks. However, in the testing dataset, there are some records that do not fall in any category in the training data. Hence, the "other" label is considered to refer to uncategorized records. Table 3 presents the normalized values of 23 labels in the training dataset and the "other" connection type.

Table 3. Label normalized values

Labels	Normalized values	Labels	Normalized values
back	-0.99	perl	0.09
buffer_overflow	-0.9	phf	0.18
ftp_write	-0.81	pod	0.27
guess_passwd	-0.72	portsweep	0.36
imap	-0.63	rootkit	0.45
ipsweep	-0.54	satan	0.54
land	-0.45	smurf	0.63
loadmodule	-0.36	spy	0.72
multihop	-0.27	teardrop	0.81

neptune	-0.18	warezclient	0.90
nmap	-0.09	warezmaster	0.99
normal	0	other	1

After implementing the proposed normalization function, results produced are much better than the method in [1]. There are no extra data in the normalized file. Every column in the dataset is not divided to several columns. Table 4 shows an example of a back attack connection is normalized and converted to numeric data.

Table 4: Back attack connection

Original data	Normalized data	Original data	Normalized data
0	-1	0	0
Tcp	0.5	1	-1
http	-0.9	2	-0.992
SF	0	0	0
54540	-0.999	0	0
8314	-0.999	0	0
0	0	0.5	0.5
0	-1	1	1
0	-1	0	0
2	-0.948	1	1
0	-1	1	-0.992
1	1	1	-0.992
1	-0.999	1	1
0	0	0	0
0	0	1	1
0	-1	0	0
0	-1	0	0
0	-1	0	0
0	-1	0	0
0	0	0	0
0	0	back	-0.99

Neural Networks' Architecture

As discussed, neural networks can have different architectures based on the number of neurons and the activation function used in each layer. Figure 1 shows the general architecture of neural networks implemented in this research. The neural network is designed as follows: the input layer contains 41 neurons; the hidden layer is divided in two sub-layers: hidden sub-layer 1 and hidden sub-layer 2; and finally, the output layer is made of 1 neuron. The input layer has 41 neurons to fit with the KDD Cup 1999 data set because each connection in the data set has 41 features, exclude the label. The number of neurons of each hidden sub-layer is changed in each experiment for comparing the efficiency of different architectures. The output layer only provides Boolean result: attack or non-attack; so it only needs one neuron.

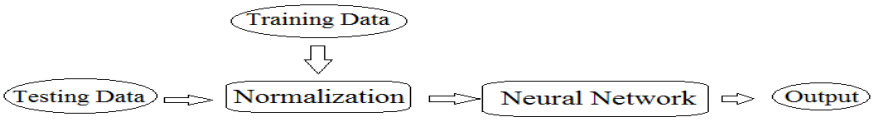

Figure 1. Neural network's architecture

Experiments and Results

In order to experiment with different architectures of neural networks to find out the most efficient solution, the 10% subset of the entire KDD Cup 1999 dataset is used. This subset contains about 400,000 records of network traffic. The error rate is set at 0.02, i.e. 2%, to ensure that the neural network is trained efficiency. As mentioned, the data is normalized in the range between -1 and 1, so two activation functions used are: sigmoid and hyperbolic tangent. The reason is those functions work with both negative and positive numbers. The method used to train neural networks are propagation training, a supervised training, because expected outputs are provided. There are two forms of propagation training are applied in the approach: backpropagation and resilient propagation. In order to compare the efficiency of each architectures, training and testing time caculated in miliseconds are used. There are total 32 different architectures in the experiments as described in Table 5. Because the numbers of neurons in the input layer and the output layer are remained, only numbers of hidden layers are presented in Table 5.The first number is the number of neurons of the hidden layer 1 while the second one refers to the number of neurons of the hidden 2. And then the activation and the propagation training method are addressed. Morover, each architecture is assiged a code for ease of expression.

Table 1. Architectures involved in the expertiment

Architecture	Code	Architecture	Code
9 – 10, Sigmoid, Resilient	1	9 – 14, Sigmoid, Resilient	17
9 – 10, Tangent, Resilient	2	9 – 14, Tangent, Resilient	18
9 – 10, Sigmoid, Back	3	9 – 14, Sigmoid, Back	19
9 – 10, Tangent, Back	4	9 – 14, Tangent, Back	20
9 – 11, Sigmoid, Resilient	5	9 – 15, Sigmoid, Resilient	21
9 – 11, Tangent, Resilient	6	9 – 15, Tangent, Resilient	22
9 – 11, Sigmoid, Back	7	9 – 15, Sigmoid, Back	23
9 – 11, Tangent, Back	8	9 – 15, Tangent, Back	24
9 – 12, Sigmoid, Resilient	9	10 – 10, Sigmoid, Resilient	25
9 – 12, Tangent, Resilient	10	10 – 10, Tangent, Resilient	26
9 – 12, Sigmoid, Back	11	10 – 10, Sigmoid, Back	27
9 – 12, Tangent, Back	12	10 – 10, Tangent, Back	28
9 – 13, Sigmoid, Resilient	13	10 – 11, Sigmoid, Resilient	29
9 – 13, Tangent, Resilient	14	10 – 11, Tangent, Resilient	30
9 – 13, Sigmoid, Back	15	10 – 11, Sigmoid, Back	31
9 – 13, Tangent, Back	16	10 – 11, Tangent, Back	32

Conclusion

This study already proves a reliable and efficient solution for detecting simulated attacks in computer networks. The system includes two components: the Data Pre-Processing module and the Neural Network. The Data Pre-processing module plays a role of processing data in the KDD Cup 1999 data set before data is used in the Neural Network. Meanwhile, the Neural Network is to detect simulated attacks. There are total eight different structures used to evaluate the Neural Network. These structures are the combinations of different numbers of neurons in two hidden layers of the Neural Network. These eight neural networks are built using the feedforward algorithm and trained using the resilient propagation algorithm. Each neural network is trained with a 73,249-records training data set. Then it is tested against three different testing data sets: the training data set, the normal traffic data set, and the 10-percent subset of the KDD Cup 1999 data set. The detection rates are 99.89%, 99.9%, and 93% respectively. The proposed approach produces highly accurate results compared with other approaches. However, while most previous studies use large training data and small testing data, the ratio of training data and testing data in this study is 0.15. Nevertheless, this study still needs to be improved in the future as discussed in the following section.

Future work of this study should include improving the executing time by applying parallel computing. Currently, it takes about 12 hours to train the neural networks using the discussed training data set. Hence, parallel computing may be applied to improve the training speed. Another important issue is to increment the detection rate by improving the training algorithms or using the enhanced version of the KDD Cup 1999 data set, NSL-KDD [[17]. This data set removes duplicate records in the original KDD Cup 1999 data set. The results presented in [17] denote that this data set is more reliable than the KDD Cup 1999 data set. Another work is to evaluate the combination of this approach with other methods using in intrusion detection systems. The cooperating method could be intelligent agents or data mining technique. Finally, a method to apply the neural networks into real computing networks should be addressed to make the research more practical.

References

[1] https://www.securelist.com/en/analysis/204792255/Kaspersky_Security_Bulletin_2012_The_overall_statistics_for_2012#6

[2] J. Shum and H. A. Malki, "Network intrusion detection system using neural networks," *Fourth International Conference on Natural Computation*, vol. 5, p. 242-246, Oct. 2008.

[3] R. Weaver, "Guide to network defense and countermeasures," Jan. 2006.

[4] D. Jiang, Y. Yang, and M. Xia, "Research on intrusion detection based on an improved SOM neural network," *Fifth International Conference on Information Assurance and Security*, vol. 1, p. 400-403, Aug. 2009.

[5] I. Ahmad, A.B. Abdullah and A.S. Alghamdi, "Application of artificial neural network in detection of DOS attacks," *2nd International Conference on Security of Information and Networks*, 2009.

[6] G. Poojitha, K. Naveen kumar and P. JayaramiReddy, "Intrusion detection using artificial neural network," *International Conference on Computing Communication and Networking Technologies*, p. 1-7, Jul. 2010.

[7] N. Bashah, I. B. Shanmugam and A.M. Ahmed, “Hybrid intelligent intrusion detection system,” *World Academy of Science, Engineering and Technology*, 2005.

[8] R. Beghdad, “Critical study of neural networks in detecting intrusions,” *Computers and Security*, p. 168-175, Jun. 2008.

[9] I. Mukhopadhyay, M, Chakraborty, S. Chakrabarti and T. Chatterjee, “Back propagation neural network approach to intrusion detection system,” *International Conference on Recent Trends in Information Systems*, p. 303-308, Dec. 2011.

[10] X. Yao, “A network intrusion detection approach combined with genetic algorithm and back propagation neural network,” *International Conference on E-Heath Networking, Digital Ecosystems and Technologies*, p. 402-405, Apr. 2010.

[11] X. Han, “An improved intrusion detection system based on neural network,” *International Conference on Intelligent Computing and Intelligent Systems*, p. 887-890, Nov. 2009.

[12] kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[13] www.heatonresearch.com

[14] J. Heaton, “Programming neural networks with Encog 3,” Oct. 2011.

[15] J. Heaton, “Introduction to neural networks for C#,” 2008.

[16] A. D. Amastasiadis, G. D. Magoulas, and M. N. Vrahatis,”New globally convergent training scheme based on the resilient propagation algorithm," *Neurocomputing*, vol. 64, p. 253-270, 2005.

[17] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani , "A detailed analysis of the KDD CUP 99 data set," *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on* , p. 1,6, 8-10, July 2009