ISSN 2395-8618

Detecting Simulated Attacks in Computer Networks Using Resilient Propagation Artificial Neural Networks

Mario A. Garcia and Tung Trinh

***Abstract*—In a large network, it is extremely difficult for an administrator or security personnel to detect which computers are being attacked and from where intrusions come. Intrusion detection systems using neural networks have been deemed a promising solution to detect such attacks. The reason is that neural networks have some advantages such as learning from training and being able to categorize data. Many studies have been done on applying neural networks in intrusion detection systems. This work presents a study of applying resilient propagation neural networks to detect simulated attacks. The approach includes two main components: the Data Pre-processing module and the Neural Network. The Data Pre-processing module performs normalizing data function while the Neural Network processes and categorizes each connection to find out attacks. The results produced by this approach are compared with present approaches.**

***Index Terms*—Computer security, artificial neural network, resilient propagation.**

I. Introduction

The number of web attacks in the United States was 413,622,456, which shows the importance of detecting and preventing intrusions [1]. Moreover, according to Shum and Malki [2], from July 2004 to August 2004, the number of network attacks increased 55%. Those statistics alarm network security communities to develop more secured solutions that could protect the tenets of information security: confidentiality, integrity, and availability [3]. There are many proposed methods to develop an intrusion detection system; however, the neural network is considered as an alternative solution to detect zero day attacks. An advantage of neural networks for intrusion detection is that they can "acquire knowledge through learning and store it in inter-neuron connections known as synaptic weights" [4]. In other words, neural networks can detect attacks after they were trained with a subset of network traffic representing the signatures of the attacks to be detected. This method is called "Supervised Learning" because the neural network needs to be trained. (Unsupervised Neural Network can detect attacks without training.) There are several training algorithms to train neural networks such as back propagation, the Manhattan update rule, Quick propagation, or Resilient propagation.

This research describes a solution of applying resilient propagation artificial neural networks to detect simulated attacks in computer networks. The resilient propagation is a supervised training algorithm. The term "supervised" indicates that the neural networks are trained with expected output. The resilient propagation algorithm is considered an efficient training algorithm because it does not require any parameter setting before being used [14]. In other words, learning rates or update constants do not need to be computed. The approach is tested on eight neural network configurations and the results are compared with other approaches found in the literature.

A. Neural Networks and Intrusion Detection

An intrusion is defined as "an attempt to gain unauthorized accesses to network resources" [5]. External people or internal users of networks can be responsible for an intrusion. There are many types of intrusions that are being used by hackers such as viruses, Trojan, attempt break in, successful break in, and Denial-of-Service [6]. An intrusion detection system is 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 [7]. Based on the trigger, intrusion detection systems can be divided into two types: misuse detection and anomaly detection. Misuse detection is a method of using attack databases to identify attacks. Every activity is compared with known attacks to figure out if that activity is an attack or not. In contrast, anomaly detection finds intrusions by keeping track of characteristics of profiles of authorized users or groups in the network and alerting on discrepancies. Intrusion detection systems use alarms to label those profiles.

A neural network is "an information processing system that is inspired by the way biological nervous systems, such as the brain, process information" [8]. In other words, a neural network consists of a number of elements which work together to solve a given problem. In additional, a neural

Manuscript received on January 15, 2015, accepted for publication on May 10, 2015, published on June 15, 2015.

The authors are with Texas A&M University–Corpus Christi, Computer Science, 6300 Ocean Dr., Corpus Christi, TX, USA (e-mail: {mario.garcia, tung.trinh}@tamucc.edu).

ISSN 2395-8618

network also can be trained to gain knowledge before being used. A neural network contains two main components: the input layer and the output layer. Depending on the complexity of the problem, a neural network can have several hidden layers between the input layer and the output layer. The number of neurons in the input layer should match the size of the input. Normally, the neuron in the output layer is one. The hidden layer plays a role of a data processing station. These layers handle data from the input layer and transfer processed data to the output layer. Neurons in two adjacent layers are connected by the weights. These weights are used to compute the output and to minimize the error produced by the neural networks.

There are two algorithms used to define the structure of a neural network: the topology algorithm and the training algorithm. The topology algorithm refers to the way neurons are connected and how data is transferred between neurons, while the training algorithm denotes the method to adjust weights between neurons to produce accurate output and minimal error. The function to calculate the output is the activation function attached in hidden layers and the output layer. Equation 2.1 shows how to compute the output from a node *j*.

$$output_j = f(x_i) = f\left(\sum_{i=1}^{n} o_i w_{ij}\right) \qquad (2.1)$$

where output *j* is the output of node *j*, x_j is the data of node *j*, o_i is the output of node *i* connected to *j* with the corresponding weight w_{ij}, and $f()$ is the activation function. There are three main activation functions used in neural networks: linear, sigmoid, and hyperbolic tangent. Each activation function scales output in a specific range. Input used in neural networks should be normalized into numbers in that range.

II. State of the Art

In last few years, networking researchers have developed intrusion detection systems using various neural network types. Shum and Malki [2] described a feedforward neural network using the back propagation algorithm implemented with three layers: an input layer, a hidden layer, and an output layer. Similarly, Poojitha, Naveen kumar, and JayaramiReddy [6] introduced an intrusion detection system using an artificial neural network using the back propagation algorithm. This proposed approach uses two phases, training and testing, to detect intrusion activities. First, the intrusion detection system is trained to "capture the underlying relationship between the chosen inputs and outputs" [6].

After that, the system is tested with an available data set. Mukhopadhyay, Chakraborty, Chakrabarti, and Chatterjee [9] presented a study of applying the back propagation algorithm in intrusion detection systems. This approach detects intrusions in four steps: collect data, convert data into MATLAB format, convert data into double data type. This data is used as input to the neural network. Yao [10] expressed a combination of the back propagation neural network and the genetic algorithm. 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 the alert response module. This approach is proposed to "overcome the blindness of optimization" and "avoid occurring local convergence". Jiang, Yang, and Xia [4] introduced an intrusion detection system based on the improvement of the Self-Organizing Maps 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 [4].

Han [11] proposed an improved model of the Adaptive Resonance Theory 2-A neural network which can "handle data directly". 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 filter 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 [11]. Bashah, Shanmugam, and Ahmed [7] presented a host-based intrusion detection system using both anomaly detection and misuse detection trigger with the SOM algorithm. Ahmad, Abdullah, and Alghamdi [5] described another proposed intrusion detection system. This system uses resilient back propagation algorithm to compute weights between neural neurons.

A. The Knowledge Data Discover KDD Cup 1999 Data Set

Intrusion detection systems have been grabbing attention 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 provide a completed source for implementing and testing intrusion detection systems. This database contains 22 different attacks and normal connections [12]. The data set contains around five million TCP/IP connections which are labeled as normal or attacks connections. Each connection record contains 41 features in a TCP/IP packet and the "Label" feature denoting the category into that the connection falls. Table I shows 22 intrusion categories included in the KDD Cup 1999 data set. Besides, the "normal" value is assigned to normal connections. Each attack is classified in one of four groups: DoS, U2R, R2L, and Probe. According to [12], these groups are described as follows: DoS: denial-of-service; U2R: unauthorized access to local root privileges; R2L: unauthorized access from remote machines; Probe: surveillance or other probing.

ISSN 2395-8618

TABLE I.
ATTACKS IN THE KDD CUP 1999 DATA SET

Group	Attack Names
DoS	Back, Land, Neptune, Pod, Smurf, Teardro
U2R	Buffer_overflow, loadmodule,
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient warezmaster,
Probe	ipsweep, nmap, portsweep, satan,

B. The Encog Framework

In 2008, a neural network and machine learning framework named Encog was published and developed for C/C++, Java and .NET programming languages by Heaton Research, Inc. [13]. This framework provides the library for creating neural networks and normalizing data. Developers can implement various types of neural network such as feedforward neural networks, adaptive resonance theory 1 neural networks, and self-organizing map neural networks. Moreover, Encog also contains several training techniques like backpropagation, genetic algorithm, Manhattan update rule propagation, and resilient propagation. In addition, multiple activation functions are included in Encog such as Bipolar function, linear function, sigmoid function, and hyperbolic tangent function. In this research, the Encog is used to build the neural network in .NET framework.

III. NEURAL NETWORK – INTRUSION DETECTION SYSTEM DESIGN

According to Heaton [14], neurons in a feedforward neural network are connected forward. In essence, data is transferred from the input layer to the hidden layer 1 and so on, but there are no backward connections. The weight connecting two neurons in two adjacent layers is calculated randomly in the initialization of the neural network. Then this weight is adjusted during the training process. The computation of the output in a Feedforward neural network is described as follows. First, input from the input layer is transferred to each neuron of the hidden layer 1. Then by using Equation 2.1, output of neurons in the hidden layer 1 is transferred to neurons in the hidden layer 2. Similarly, each neuron in this layer generates and then distributes its output to the output layer. Finally, the neuron in the output layer computes the final output.

The resilient propagation is a supervised training algorithm. The term "supervised" indicates that the neural networks are trained with expected output. The resilient propagation algorithm is considered the most efficient training algorithm because it does not require any parameter setting before being used [14]. In other words, learning rates or update constants do not need to be computed. As it was previously discussed, the training algorithm is used to adjust weights to produce accurate output and minimal error rates. The change in weight between two neurons is calculated using Equation 3.1:

$$\Delta w_{ij}^{k} = \begin{cases} -\Delta_{ij}^{k} \; if \; \dfrac{\partial E(w^{k})}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{k} \; if \; \dfrac{\partial E(w^{k})}{\partial w_{ij}} < 0 \\ 0 \text{ otherwise} \end{cases} \tag{3.1}$$

where $\partial E(w^k)/\partial w_{ij}$ refers to the partial derivative of the error with respect to the weight w, w_{ij} is the weight of neurons i and j, Δ_{ij}^{k} is the update value, and k expresses the index of iteration [16]. The update value Δ_{ij}^{k} is updated using Equation 3.2:

$$\Delta_{ij}^{k} = \begin{cases} \eta^{+}\,\Delta_{ij}^{k-1} \text{if } \dfrac{\partial E(w^{k-1})}{\partial w_{ij}} \dfrac{\partial E(w^{k})}{\partial w_{ij}} > 0 \\ \eta^{-}\,\Delta_{ij}^{k-1} \text{if } \dfrac{\partial E(w^{k-1})}{\partial w_{ij}} \dfrac{\partial E(w^{k})}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{k-1} \text{ otherwise,} \end{cases} \tag{3.2}$$

where $0 < \eta^{-} < 1 < \eta^{+}$, $\eta^{+}$is the increase factor and $\eta^{-}$is the decrease factor. In default, η^{+} is equal to 1.2 while η^{-} is equal to 0.5 [14]. If this result is greater than 0, it means the sign has not changed, so the update value Δ is increased by multiplying the previous update value with the increase factor. Nevertheless, if the result is smaller than 0, it means the sign has changed and the previous Δ is too large. Hence, the update value is decreased by multiplying the previous update value with the decrease factor. In order to evaluate how well the neural network is trained, the mean square error method is applied to calculate the error between actual output and expected output. Equation 3.3 describes the mean square error:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}\left(actual_{t} - ideal_{t}\right)^{2} \tag{3.3}$$

where MSE is the mean square error value, n is the size of actual output, actual t is the t^{th} actual output and the ideal t is the corresponding ideal output.

The activation function used in the neural network is the hyperbolic tangent function, which is shown in Equation 3.4.

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3.4}$$

A. The Data Pre-processing Module

As it was mentioned, neural networks are trained using the activation function. Depending on the activation function used in the neural network, data is normalized in different ranges which can be (0, 1) or (–1, 1). Because the activation function used in the neural network in this research produces output between -1 and 1, training data and testing data are normalized to values in the range from –1 to 1. In the KDD Cup 1999 data set, there are four features in an individual TCP connection in text format: protocol_type, service, flag, and label. Because each of these features has various groups of values, each group is converted into a number between –1

ISSN 2395-8618

and 1. Other features are in numeral format, so they are normalized by Equation 3.5:

$$f(x) = \frac{(x - dL)(nH - nL)}{dH - dL} + nL \qquad (3.5)$$

where *x*: value needed to be normalized; *dL*: the lowest value of the data; *dH*: the highest value of the data; n_H: the highest value of the normalization range; *nL*: the lowest value of the normalization range.

IV. Neural Network Intrusion Detection System Testing

Before describing the evaluation plan used in this thesis, a brief review of evaluations of previous and related work is provided. The KDD Cup 1999 data set is used for creating training and testing data sets in every approach. In [2], the training set contains 196 records while three testing sets are the normal traffic set, the known attack set, and the unknown attack set which consist of 50, 25, and 25 records respectively. The records in normal traffic and known attack testing set are extracted from the training set while the unknown attack set contains different data. The results of the evaluation are: 100% of normal traffic and know attacks are detected only 76% of unknown attacks are classified. The results seem very accurate, but the detection rate of new attack is quite low, only 76%. Moreover, the detection rate could increase if the size of testing data increments.

In [4], the approach uses the KDD Cup 1999 data set for training and testing data sets. Each category of attacks is trained and tested individually [4]. Detection rates are only good in three types, normal, dos, and probe, while the other two groups have very low accuracy. Furthermore, most of the testing sets have smaller sizes than their corresponding training sets. In general, the proposed solution does not perform well in detecting all types of attacks. The results are quite better than the results computed in [4]: 99.76% of normal, 100% of DoS and Probe, 67.77% of U2R and 36.84% of R2L. Nevertheless, the classifier accuracies of U2R and R2L are not acceptable.

Another similar evaluation method is described in [11]. The system is trained 10 times with the same 10% subset of the entire KDD Cup 1999 data set and then is tested with the entire data set. In other words, the number of training record is equal to the number of testing records. The lowest detection rate is 90.385% while the highest is 99.946%. The results are accurate, but in order to get that achievement, the neural network is trained with the same amount of data of the testing data. Hence, they do not fully express the capability of neural networks that is detecting new objects based on training objects. In [5], the work focuses on detection of DoS attacks.

The proposed system produces 96.16% detection rate in case of attack detection with the highest ratio is 100% and the lowest scale is 79%. However, the testing data set is extremely small, 11 back attacks and 5 normal packets while the numbers of records of other attacks are not mentioned. Meanwhile, the training data set contains full feature packet of DoS attacks of the KDD Cup 1999 data set [5]. Hence, it is hard to estimate the performance of this system. The method proposed in [9] uses a different method to evaluate neural networks: 2 testing levels. At level 1, the system is trained and then tested with the same data set. Consequently, the success rate is very high: 95.6%. However, at level 2, when the system is tested with the new data set, the success rate reduces to 73.9% [9]. Therefore, this proposed method is considered not accurate.

A. Evaluation Plan

As discussed in the previous section, some prior studies do not produce accurate results. In some cases, the results are lower than 70% which is unacceptable. Only the approach presented in [11] can generate over a 90% detection rate. However, this method uses the same amount of data for training and testing the neural network which may not be practical. Therefore, this study focuses on producing accurate testing results using a small amount of training data. In order to evaluate this approach, several neural networks are tested with the same training and testing data for comparing the efficiency of different structures of neural networks. All neural networks have the same input layer and output layer, resilient propagation training algorithm, and hyperbolic tangent activation function. The only different configuration is the number of neurons in hidden layers. The first number in the Structure column refers to the number of neurons of the hidden layer 1 while the second number is the number of neurons of the hidden layer 2. For example, the neural network NN1 has 9 neurons in the hidden layer 1 and 10 neurons in hidden layer 2. Although there are more choices, due to the limitation of time, only 8 neural networks are trained and tested.

The training data set contains 73,249 records extracted from the entire KDD Cup 1999 data set. The training set is then normalized using the Data Pre-processing module. There are 37,860 records of normal traffic included in the training data. At an individual iteration of training process, the input of the neural network is the set of 41 TCP/IP features of a particular record while the ideal output is the "Label" feature of that record as well. Each element of the input is fed into each neuron of the input layer respectively. Then the training process continues until the end of the training set is reached.

This is to ensure that the neural networks have knowledge of all attacks in the training data. If the neural networks are trained using an error rate, the training process stops immediately after that error rate is achieved. Hence, the neural networks may not be trained with attacks in the rest of the training data set. After being trained, each neural network is tested with several testing data sets. Unlike previous works, in this study, the overall detection rates are considered instead of individual attacks' detection rates. First, the neural networks

ISSN 2395-8618

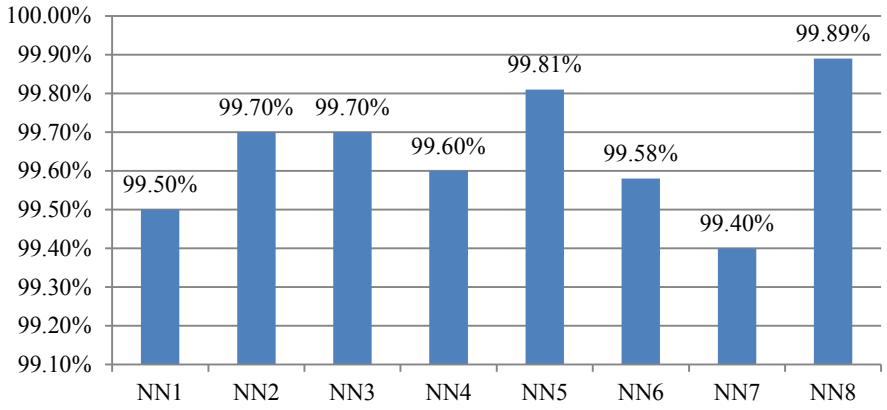

Fig. 1. Training detection rates

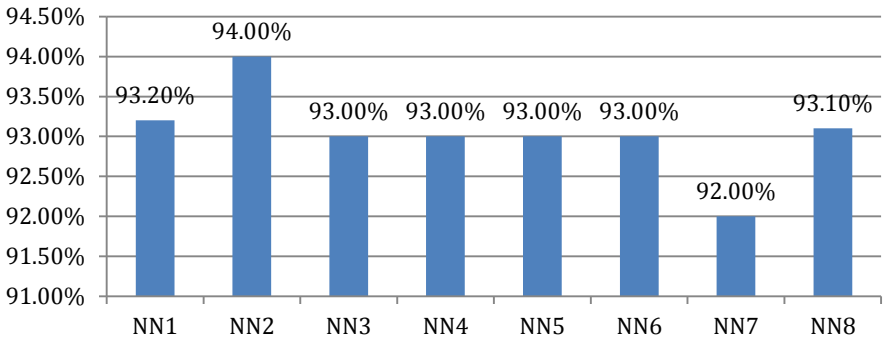

Fig. 2. Detection rates in 10-percent subset test

are tested with the same training set to verify they can detect records used to train them. Then, they are tested against a normal data set, which consists of 83,538 records.

This is to verify that the neural networks can detect normal traffic correctly to prevent false positives. After that, each neural network is tested with the 10-percent subset of the KDD Cup 1999 data set. This subset contains 494,021 records. The input and the ideal output in each test case are computed similarly as presented in the training process. The detection is considered correct if the absolute value of the actual output and the expected output is less than 0.04. Otherwise, it is counted as an incorrect detection. Similarly, the output is compared with each individual connection, including both attack values and normal values, in succession.

B. Results

In this study, the training data set contains more than 70,000 records and 23 different types of output. Hence, the smaller the error is, the better trained neural networks are. The best training error rates among all neural networks was 0.00005 and the worst was 0.00029. The average time it took to train each neural network was 12 hours. The obtained error rates denote that the actual output is very close to the ideal output.

After being trained, the neural networks were tested against the training data set itself. The best result was 99.89% generated by the Neural Network 8, while the worst was 99.4% computed by the Neural Network 7. After being trained and tested with the training data set, each neural network was tested using the normal traffic data set extracted from the KDD Cup 1999 data set. Figure 1 shows the results of this test.

The worst result is 95% produced by the Neural Network 2 while the Neural Network 8 once again produces the best result. In essence, the Neural Network 8 can detect up to 99.99912% normal traffic. Finally, the neural networks are tested with the 10-percent subset of the KDD Cup 1999 data set. This subset consists of 494,021 records, which is equal to 6.7 times of the training data set. There are 8 types of intrusions in the 10-percent subset have similar numbers of records in the training data set. Meanwhile, the rest has the big differences with the training data set.

As an illustration, the neptune attack type has only 20,482 records in the training data set, but it has 10,7201 records in the 10-percent subset. Figure 2 shows the results after testing eight neural networks with the 10-percent subset. The differences between the testing neural networks are really small. The best result is 0.06% and the worst one is 0.08%. Hence, the average result is about 93%. This result is acceptable because the training data set is much smaller than the testing data set. Moreover, the result also shows what is happening in real life where the network traffic is much larger

ISSN 2395-8618

than any data sets. After several tests, the Neural Network 8, which contains 14 neurons in each hidden layer shows the outstanding performance among 8 neural networks. Testing results generated from this neural network are used to compare with other studies.

V. 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.

VI. Future Work

Training a neural network is a time consuming process. In this research it took twelve hours to train the resilient back-propagation neural network. The amount of traffic generated in a real computer network can be extremely large. More efficient techniques such as parallel computing may be applied to speed the training.

In order to enhance the detection rate and efficiency of the neural network, different configurations of neural networks such as Back-propagation, Feed-forward, Redial-base neural networks, etc., need to be implemented and analyzed.

An enhanced version of the KDD Cup 1999 data set, NSL-KDD [16] could be used as a training dataset. This data set removes duplicate records in the original KDD Cup 1999. Even better, a huge contribution to the intrusion detection community will be the creation of a new training dataset that includes the most recent attacks.

Another task will be to evaluate the combination of this approach with other traditional methods used in intrusion detection systems including statistical and mathematical models.

One of the most relevant tasks to be performed is the application of this methodology to a real computer network in order to make the research more practical.

Acknowledgments

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Visiting Faculty Program (VFP).

References

[1] M. Baldonado, C.-C.K. Chang, L. Gravano, and A. Paepcke, “The Stanford Digital Library Metadata Architecture,” Int. J. Digit. Libr. 1 (1997) 108–121

[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] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

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

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

[15] 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.

[16] 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

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