Issues in Information Systems
Volume 17, Issue I, pp. 89-98, 2016

CHALLENGES TO MULTI-LAYER FEED FORWARD NEURAL NETWORKS IN INTRUSION DETECTION

Loye Ray, University of Maryland University College, loye.ray@faculty.umuc.edu

ABSTRACT

Today's network intrusion detection system is plagued with working in a dynamic environment. These systems must be able to cope with detecting new and unknown attacks. Current research has shown that the use of anomaly-based multi layered feed forward neural networks can provide help. However, these systems have issues that need to be resolved. These issues involve the architecture and preparation of using these systems in a network. This paper will describe the problems, what has been done to overcome these issues and address any future work needed.

Keywords: Feed Forward Neural Network, Intrusion Detection System, KDD 99 and Performance Rate

INTRODUCTION

Today's network intrusion detection systems are plagued with working in a dynamic environment. Intruders are using a wide variety of techniques that may include variations of old attacks to create new threats. These force network intrusion detection systems to overcome the difficulty of detecting new and unknown attacks (Joo, Hong & Han, 2003). These intrusion detection systems can use either misuse detection or anomaly-based detection methods. Misuse detection relies on having signatures of intrusions in order to detect attacks (Moradi & Zulkernine, 2004). They may not detect new and unknown attacks because of the lack of current signatures. Anomaly-based intrusion detection looks for abnormal behavior in a network to detect intrusions whether they are known or unknown. These traffic patterns can be seen as violations of acceptable events or when one abuses their legitimate profile of normal behavior (Moradi & Zulkernine, 2004). Current research has shown that the use of anomaly-based multi layered feed forward neural networks can detect this abnormal behavior. Joo, Hong and Han (2003) support this claim and found these types of networks work well in a dynamic environment. A neural network can also handle nonlinear data. This gives them the ability to self-adapt, self-organize, learn in real time and perform pattern recognition (Ding, Xu, Zhu, Wang & Jin, 2011). Multi-layer feed forward neural networks monitor network traffic and detect any abnormal behavior. However, these systems have issues that need to be resolved. This paper will examine these issues, what has been done to resolve them and address where future work may be needed.

MULTI LAYER FEED FORWARD NEURAL NETWORK

The multi-layer feed forward neural network architecture is a commonly used model for working with anomaly-based intrusion detection systems (Bhaskar, Kamath & Moitra, 2008; Wei, Hao-yu, Xu, Yu-xin & Aigo, 2010). This is because of the complexity of detecting intrusions and neural networks are good for resolving these kinds of problems. It works to determine irregularities within large amounts of data (Amer & Hamilton, 2009). This is very important when trying to find specific intrusions in today's global networks. Neural networks are well suited to finding intrusions in large datasets. This architecture model is used for pattern recognition by training it to recognize normal and attack patterns based on different input vectors. The neural network tries to identify the input and attempts to create an output based on what was taught (Moradi & Zulkernine, 2004). This training function uses an optimization process to find the best connection coefficients to find the best solution. It functions by using a series of steps that include: input a number of parameters to represent a vector, compare the output result to what was expected, and adjust the coefficients and rerun the process to obtain a better match (Moradi & Zulkernine, 2004). The paper will concentrate on this model.

Preprocessor

Before data is inputted to the multi-layer feed forward neural network, it must be converted to numerical data in order to process through the intrusion detection system. This is because the multi-layer feed forward neural network uses only numerical data and raw network traffic data may contain alphanumeric information (Moradi & Zulkernine, 2004). A conversion table is used to convert string data into numerical data. For example, protocol types such as Transmission Control Protocol, User Datagram Protocol and Internet Control Message Protocol can be converted to numerical values of 1, 2, and 3 respectively. The preprocessing is cumbersome due to the fact it is usually done by hand. Also the preprocessor needs to remove irrelevant and duplicate attributes in the dataset because it can make the model too complex and reduce the effectiveness of the intrusion detection system (Farid, Darmont, Harbi, Hoa & Rahman, 2009). After the preprocessor, the data is then sent to the input layer of the multi-layer feed forward neural network. Further examination of the preprocessor will not be part of this paper.

Architecture

The network structure of a feed forward neural network is simple. It consists of layers of interconnected neurons that feed information from one neuron to the next (Anyanwu, Keengwe & Arome, 2010). The architecture of a multi-layer feed forward neural network is composed of three layers as shown in figure 1. Each layer is constructed using neurons that act like the neurons in the human nervous system. Information propagates through the multi-layer feed forward neural network input layer then is forwarded to the hidden layer and passes into the output layer using a transfer function (Wei, Hao-yu, Xu, Yu-xin & Aigo, 2010). This is where the feed forward neural network gets its name by processing information in a forward direction (Anyanwu, Keengwe & Arome, 2010). In fact, the multi-layer feed forward neural network system, using back propagation uses a two stages process. There is a forward process where data is processed through each layer. Then an error signal is fedback to each layer and the process begins again (Gu, Shi & Wang, 2012). Back Propagation uses a gradient descent method to reduce the amount of error to a minimum amount (Ding, Xu, Zhu, Wang & Jin, 2011). The input layer works to take in numeric traffic data from the preprocessor. The number of neurons in the input layer is determined by the number of intrusion features to be processed. The hidden layer takes the output from the input layer and determines if the data is an intrusion or not. The number of hidden neurons is determined by trial and error and affects the detection rate of the intrusion detection system (Ray, 2013a). The result is sent to the output layer where it is classified according to the specific type of intrusion or as normal traffic. The number of different intrusions being identified determines the number of neurons in the output layer. Each neuron in all layers is given a random weight to calculate how close the neural network can recognize traffic as being normal or abnormal. After each iteration, the neural network has each neuron adjust their weights based on the output error difference between the calculated and actual target values (Ray, 2013b; Smith, 2002). The process continues until the difference or error rate is very small.

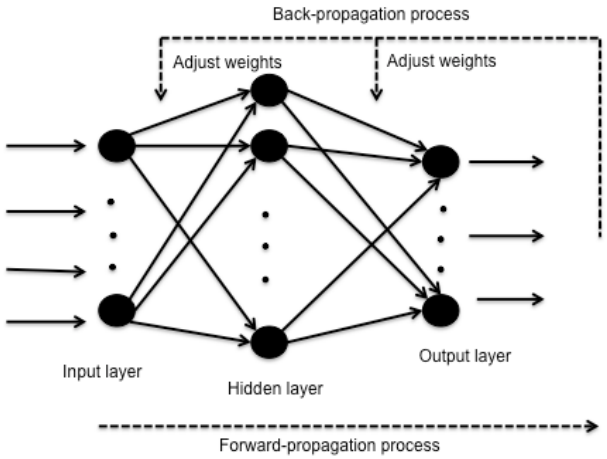

Figure 1. Feed forward neural network structure with back propagation algorithm

To get the best detection rate of abnormal traffic one needs to use an algorithm to adjust the weights and feed any error in the output back to the layers in the neural network. The back propagation algorithm is a common used method to perform this. It takes the detection error from the output layer and feeds it back to the hidden and input layers to adjust the weighting functions to improve the calculation of abnormal behavior being detected (Ray, 2014). According to Wei, Hao-yu, Xu, Yu-xin and Aiguo (2010) and Tian and Gao (2009), when the output layer doesn't provide expected results, an error signal is fed back along the original path to modify weights of each layers neurons. Then the process of detection through the multi-layer feed forward neural network begins again and continues until the error value matches a desired value. This training process is a continuous adjustment of the weights until the amount of error reaches a preset value (Ding, Xu, Zhu, Wang & Jin, 2011). The back propagation algorithm provides good detection capability but a slow training (convergence) rate. This slow training rate causes more time to be spent on getting the neural network to learn what is or is not an intrusion. This time keeps the intrusion detection system from being used to detect intrusion and reduces its efficiency. The reason is that training a neural network is done off-line (Ray, 2013b). The neural network using a back propagation algorithm may provide good performance but it can't guarantee the results will provide a global optimum solution.

An alternative common algorithm is the genetic algorithm that generates rules to improve multi-layer feed forward neural network learning. The genetic algorithm is a way of overcoming the poor training rate of the back propagation algorithm (Ray, 2013b; Ding, Su & Yu, 2011). The genetic algorithm works to discover the optimal solution to a given situation by adapting the probabilities according to different conditions (Li, 2004; Tian & Gao, 2009). It does this without the use of gradient error functions. These can be compared to biological evolution such as chromosomes in animals (Gu, Shi & Wang, 2012). As the neural network trains, rules are created to help it recognize abnormal behavior faster than the back propagation algorithm. It is used to produce the best architecture and weights (Hua & Xiaofeng, 2008). Thus, the genetic algorithm is a powerful tool for improving machine learning and optimizing neural networks (Ding, Su & Yu, 2011). However, the genetic algorithm doesn't detect intrusions as well as the back propagation algorithm.

Assessment

An important factor in looking at neural network intrusion detection systems is to determine how well it performs. These involve determining the convergence (training) rate and performance rate (detection and false alarms) of the multi-layer feed forward neural network intrusion detection system (Ray, 2013b). The convergence rate is a measure of the time it takes for the neural network to be trained. Each time a neural network goes through a training cycle is called epoch. To determine if the network has completed training is to set the epoch value to the amount of error that is acceptable from the output. To measure the performance of a neural network, one uses a confusion matrix. The confusion matrix is a table of calculations for determining the detection rate, false positive rate, and false negative rate measured as a percentage (Engen, Vicent & Phalp, 2008; Wu & Banzhaf, 2010). The detection rate is how well the network has accurately detected normal or abnormal traffic. The false positive rate is a measure of error where the network can't clearly determine if an intrusion exists or not. The false negative rate is a measure of intrusions that get falsely recognized as normal behavior. These measures play an important part in creating an effective multi-layer feed forward neural network intrusion detection system by ensuring that intrusion detection systems can successfully detect all types of intrusions. Any error can have a dramatic effect on an intrusion detection system by reducing its ability to identify intrusions and causing countermeasures to be used unnecessarily (Joo, Hong & Han, 2003). Therefore, an effective neural network-based intrusion detection system will have a high detection rate and low false alarm rate (Li, Fang, Guo & Chen, 2007).

ISSUES FOUND

The literature and research shows that a multi-layer feed forward neural network intrusion detection system provides a good solution for detecting intrusions in a network. However, it was found that there are some issues that need to be addressed before the solution can be effective. A summary of these issues can be found in Table 1. A significant consideration in providing accurate classification is affected by the architecture and algorithm used (Chaudhary & Swarup, 2009). These two major issues will be the focus of this paper.

Architecture Challenges

Modeling of a multi-layer feed forward neural network intrusion detection system is an important problem according to Bhaskar, Kamath and Moitra (2008). The basic design of a multi-layer feed forward neural network is simple and composed of input, hidden, and output layers. However, beyond this the architecture can be quite different. One reason for this is that there is no standard intrusion detection system model being used. Each researcher builds their model according to it specific use. Ahmad, Abdullah and Alghamdi 2009 constructed their model to detect denial of service attacks. Bi, Zhang and Cheng (2010) built their model to detect a series of specific attacks centered on a type of attack. A factor contributing to this is the number of neurons for each layer can vary depending on the use of the model. The number of neurons being used for each layer can also effect the performance of the intrusion detection system (Ding, Xu, Zhu, Wang & Jin, 2011). A standard configuration based on a set number of neurons per layers may be difficult to devise. Wei, Hao-yu, Xu, Yu-xin and Aiguo (2010) support this and add that the selection of different layers can influence how the neural network works. The size and features of the data coming into the network determine the number of input neurons. So if you have 41 features being submitted to the network, one must have 41 neurons in the input layer.

Another consideration is the amount of input data that can affect the efficiency of the neural network. Today's intrusion detection systems must be able to handle huge volumes of data from various points within a network (Chandola, Banerjee & Kumar, 2009). This means that these devices need to be efficient computation wise to do online analysis quickly. Too much input data or features can reduce convergence rate causing more time to train the network. On the other hand, too little data or features can reduce detection of new and unknown attacks. Amer and Hamiliton (2009) support this claim by describing that a balance must be maintained between using large datasets and controlling the amount of false alarms. Today's intrusion detection systems need to be able to detect various types of intrusions while keeping false alarms low to prevent attackers from getting through. With more input data comes an increase in the false alarm rate. With billions of data packets being analyzed, just two to three percent of false alarms can overcome the security analyst determining what's happening (Chandola, Banerjee & Kumar, 2009).

Determining the number of hidden layer and number of neurons for this layer can be difficult (Ray, 2013a). The universal approximation theorem, for this model, finds that one or more hidden layers can better approximate functions and produce precise outputs (Moradi & Zulkernine, 2004). However, this can result in more neurons being used in each hidden layer. These additional hidden layers and neurons increase the training time for only a slight improvement in computational accuracy. Moradi and Zulkernine (2004) found that using just one hidden layer reduces the complexity while increasing computational efficiency and decreased training time. The efficiency between the use of one or two hidden layered architecture was about 4% (Moradi & Zulkernine, 2004). Calculating the neurons for the hidden layer can be difficult (Ray, 2013a; Wei, Hao-yu, Xu, Yu-xin & Aigo, 2010). Wei, Hao-yu, Xu, Yu-xin and Aiguo (2010) found that too few neurons cause more errors while too many result in poor training time. Determining the number of neurons in the hidden layer is determined by trial and error because there is no mathematical equation to calculate a value. Researchers have used trial and error instead of seeking a mathematical or algorithm to help overcome this problem. Researchers Wang and Ma (2009) and Wei, Hao-yu, Xu, Yu-zin and Aiguo (2010) each varied the number of hidden layers in determining the optimum number.

Research has shown that the number of neurons in the output layer varies with individual researchers (Ray, 2014). Some use only one neuron while others use two, three, or more. One researcher has used 225 neurons. The issue here is the lack of a standard number of output neurons to identify all variations of old and new intrusion patterns. However, basically there didn't seem to be any limitation on detecting normal or attacks using any number of output neurons (Ray, 2014). It all depends on how specific you want the output to identify particular attack types. Ray (2014) found that one needs to determine what specific types of threats to show. One neuron is needed for each type of threat.

The use of various algorithms helps to improve the convergence and performance rates in multi-layer feed forward neural networks. The most common used algorithm is the BP. It can provide good detection capability but has a poor convergence rate. According to Bhaskar, Kamath and Moitra (2008) the problem is that it consumes too much time training the neural network. Also it was found that the back propagation algorithm becomes difficult to function effectively when the network is complicated such as with multiple hidden layers. Simpler architectures are

recommended when using the back propagation algorithm (Moradi & Zulkernine, 2004). Some researchers have tried to improve the back propagation algorithm to some degree but with little improvement. Research shows that using the genetic algorithm results in a good convergence rate but with a poor detection rate (Ray, 2014). However, multi-layer feed forward neural network intrusion detection systems need both a good convergence and performance rate to be effective. The issue here is how one can harness the strength of both algorithms while reducing the negative affects of any one algorithm.

Training and Testing Challenges

As mentioned earlier, a multi-layer feed forward neural network recognizes only numerical data. However, real traffic collected from live networks and simulated datasets such as Knowledge Discovery and Databases 99 (KDD 99) contain alphanumeric information Li, Kang, Guo and Chen (2007) found that this makes it incompatible with being processed by the multi-layer feed forward neural network without converting the nonnumeric data to numeric information (Ray, 2013b). Trying to use the nonnumeric data can cause errors and the intrusion detection system may not recognize an intrusion. This could be devastating to a network. Handling large volumes of data is another problem according to Kabiri and Ghorbani (2005). They describe that a way needs to be found to handle this amount of data without losing its importance. So researchers have relied on customizing the amount of samples used to train and test their models (Engen, Vicent & Phalp, 2011). They use this to maximize results and prove their model effectively detects intrusions. Bhaskar, Kamath and Moitra (2008) found that reducing the data to be tested would affect performance while improving training time. This is more apparent in models that detect a limited type or number of intrusions. Ahmad, Abdullah and Alghamdi (2009) specified data for denial of service attacks while Balram and Wiscy (2008) detected probe scans. The end result can make the model ineffective to detect new and unknown intrusions. Therefore, one must consider tradeoffs regarding data and performance (Bhaskar, Kamath & Moitra, 2008).

There are two types of data used to conduct training and testing of multi-layer feed forward neural network intrusion detection systems. These include simulate data (KDD 99) and real traffic data collected from a network. The KDD 99 dataset is a publicly available simulated dataset that is commonly used in research. It contains about 5 million records composed of 41 features describing different normal and attack connection properties (Ray, 2013b). These are broken down into four categories composed of denial of service, root, remote to user, and probe attacks (Chaudhary & Swarup, 2009; Kadeeban & Ghorbani, 2011). These 41 features have large dimensions and can result in long training times and results that are not close to the desired (Gu, Shi & Wang, 2012). However, this dataset is outdated with no publicly accessible substitute available (Ray, 2013b). The KDD 99 dataset was devised in 1999 and intrusions have dynamically changed since then. Choudhary and Swarup (2009) recommend expanding the KDD 99 dataset with additional scenarios. Even with these limitations, the KDD 99 dataset is still being used. Ray (2013b) found that an alternative to simulated data is the use of real traffic collected from a live network. Sit was found that researchers have collected small amounts of live data for specific types of intrusions but the majority of data comes from the KDD 99 dataset. The challenge to using real data is how to collect it and does the data contain enough of normal and attack information to be useful. A possible compromise is to use both KDD 99 dataset and real data to generate adequate samples for effectively training and testing a model (Wu & Banzhaf, 2010). Also today's attacks are ever changing and these can have similarities to normal traffic (Farid, Darmont, Harvi, Hoa & Rahman, 2009; Ibrahim, 2010). Also data being collected may be noisy (abnormal data within the normal traffic) making it harder to train the neural network-based intrusion detection system (Li, Fang, Guo & Chen, 2007). This can result in continuing training and testing of the multi-layer feed forward neural network intrusion detection system to detect these new variances.

RESULTS

Current research shows promising solutions to these issues. Table 1 briefly describes the results from this research. On the problem of detecting the number of neurons to use, one needs to maximize the number in order to ensure known, unknown and new intrusions are detected without too much sacrifice for convergence rate. However, this can negatively affect the convergence and performance rates. Wang and Ma (2009) built their model using a

different number of neurons in the input and hidden layers to test the best detection rate and training time values. To overcome the issues in the hidden layer, most researchers today use only one hidden layer with a balance of neurons to get an effective detection rate with low false positive rate and false negative rate. Also several researchers have varied the number of hidden layer neurons to see the effects of these rates. Ahmad, Abdullah and Alghamdi 2009 and Wang and Ma (2009) varied the number of neurons in their model's hidden layer. Wang and Ma (2009) found a nonlinear relationship between detection rate and number of neurons in the hidden layer. Some theoretical research has proposed a few equations but they have not been quantitatively tested. There seems to be many models that use one or more output neuron to indicate if the output is normal or an intrusion. Balram and Wiscy (2008) used one to indicate if a scan was being done. Bi, Zhang and Cheng (2010) however used seven output neurons to describe different specific attacks. These don't provide enough detail in their output layer to indicate the type of intrusion detected. Other researchers have used six output neurons to indicate normal and five types of attacks. Tain and Gao (2009) model encoded five neurons using ones and zeros to indicate intrusion 1, intrusion 2, etc. However, their model didn't define a specific intrusion type for each output. Still others have used a number of neurons to equal the number of intrusion categories in the KDD 99 dataset and one for normal traffic detection. Some have configured the output to indicate normal, known and unknown attacks. This is too basic and doesn't identify what the attack is so one can implement the correct response.

The use of a hybrid algorithm has been proposed to overcome weakness in the back propagation and genetic algorithms. The hybrid algorithm is composed of the back propagation and genetic algorithms. This hybrid algorithm enhanced both detection and convergence rates (Tian & Gao, 2009). The precise hybrid algorithm of Tain and Gao (2009) was found to be more stable and have higher accuracy than traditional back propagation algorithms. Ganatra, Kosta, Panchal and Gajjar (2011) propose a hybrid composed of back propagation and genetic algorithms but didn't provide any quantitative data to verify results. Ding, Su and Yu (2011) proposed using a hybrid algorithm to improve the training deficiencies of the back propagation algorithm. However, it was found that the hybrid algorithm is only good for some datasets. When the data is complex, this can stall either back propagation or genetic algorithms. The hybrid algorithm also has trouble providing valid results when the input data is complex (Ding, Su & Yu, 2011). To build models, researchers have used Java, C++ and Matlab. Matlab seems to be a favorite modeling tool by some researchers. Guangjun, Jialin, and Zhenlong (2008), Hua and Xiaofeng (2008), and Wang and Ma (2009) utilized Matlab to build and test their model.

When it comes to training and testing any multi-layer feed forward neural network intrusion detection system model, researchers seem to use a variety of sample sizes to improve detection. This is because they want to maximize their results to look good. Kandeeban and Rajesh (2011) used 50,000 samples for training and 10,000 for testing. In contrast, Shum and Malki (2008) utilized 20 to 200 samples. However, in a real network the number of samples needs to be as large as possible in order to detect a wide variety of intrusions. Most of these samples come from using the KDD 99 dataset. Other researchers use a small sample size based on specific types of attacks. All this limits the usefulness of the intrusion detection system in a real environment exposed to a dynamic range of intrusions. Researchers have recommended using real traffic with only a few describing how they collected data. Wang and Ma (2009) collected data on a network using Ethereal but the quantitative data provided was to little to validate their findings. Another method used was collecting real traffic data from the Internet using an Internet server and honeypot (Mafra, Moll, Fraga & Santin, 2010). According to Engen, Vincent and Phalp (2011) and Li (2004) there is still a need for a standard dataset for training and testing models.

Table 1. Issues Found and Results of Research

Category	Issue	Results from Research
Architecture	No standard intrusion detection model	The combination of single hidden layer, maximum # of hidden neurons, use of hybrid algorithm and maximum number of output neurons present good model
	Amount of input data negatively effects efficiency	Balance amount of input data to detection results of neural network IDS
	Difficult to determine # of hidden neurons needed	Use one hidden layer and maximize number of hidden neurons
	Difficult to determine right # of output neurons	Use at least one neuron for detection and a neuron for each type of specific threat being detected

	Difficult choosing right algorithm	Use hybrid algorithms to improve detection rate
Training and testing	Determining the amount of data samples difficult	Use maximum amount of samples for training and testing
	Choosing real or simulated input data to train IDS	Use real traffic for both training and testing

SUMMARY/CONCLUSION

In summary, this study has explained that the issues with using multi-layer feed forward neural network intrusion detection systems can be resolved. Using a hybrid algorithm based on back propagation and genetic algorithm may overcome the weaknesses of these individual algorithms. Researchers Tain and Gao (2009), Ganatra, Kosta, Panchai and Gajjar (2011), and Hua and Xiaofeng (2008) concur but still recommend further quantitative testing using both simulated and real traffic data. It was discovered that maximizing the multi-layer feed forward neural network intrusion detection system architecture to process many types of intrusions could improve detection of known, unknown, and new intrusions. However, the intrusion detection system architecture will need to maintain a balance of convergence and performance rates to avoid missing attacks. Amer and Hamiliton 2009 supports this claim and explains that an intrusion detection system must be carefully designed in order to be effective. Otherwise, the performance of the multi-layer feed forward neural network intrusion detection system will degrade according to Bhaskar, Kamath and Moitra (2008). Also, the use of a composite data sample of real and simulated data may overcome the flaws in using just the KDD 99 dataset. However, there is no quantitative data to support these claims. Wu and Banzhaf (2010) also found that such a dataset would need a method to measure its effectiveness.

In conclusion, further research is needed to optimize multi-layer feed forward neural network intrusion detection systems. These include devising and testing a robust model that can detect numerous types of new and unknown intrusions. Kabiri and Ghorbani (2005) found that building such a model is difficult. Matlab could provide a means to build and automate this model to reduce the difficulty. The model should maximize the number of neurons in each layer to detect a wide variety of intrusions. To support this, the input data should not be restricted to specific types but allowed to come from various different types of intrusions. Next, the model needs to be trained and tested using a large sample size in order to ensure the best detection rates and less false alarm rates. The number of neurons in the hidden layer should also increase as the samples increase (Wei, Hao-yu, Xu, Yu-xin & Aigo, 2010). To accomplish this requires devising a dataset composed of both simulated and real network traffic. Collecting data from real traffic could be combined with the KDD 99 dataset to create an updated source of data. The method used by Mafra, Moll, Fraga and Santin (2010) provides a viable means of collecting real traffic data to merge with the KDD 99 dataset.

Also, other algorithms should be tried in combination such as fuzzy logic. Garcia-Teodoro, Diaz-Verdejo, Macia-Fernandez and Vazquez (2009) and Ibrahim (2010) recommend using fuzzy logic with neural network intrusion detection systems. This is because the data is changing as attackers are varying their tactics using advanced persistent threat techniques. Therefore, fuzzy logic can be used to handle these variables as fuzzy data (Garcia-Teodoro, Diaz-Verdejo, Macia-Fernandez & Vazquez, 2009). One drawback to using fuzzy logic is its high consumption of resources (Garcia-Teodoro, Diaz-Verdejo, Macia-Fernandez & Vazquez, 2009). Lastly, quantitative research should be done to validate the findings from building, training and testing the model. The model architecture needs to be assessed on its effect on convergence and performance rates using a confusion matrix. This will help others to duplicate and improve on the model.

REFERENCES

Ahmad, I, Abdullah, A. B. & Alghamdi, A. S. (2009, October). *Application of artificial neural network in detection of DOS attacks*. Paper presented at the Second International Conference on Security of Information and Networks, North Cyrus, Turkey.

Amer, S. H. & Hamilton, J. A. (2009). Input data processing techniques in intrusion detection systems: Short review. *Global Journal of Computer Science and Technology*, *1*(2), 2-6.

Anyanwu, L. O., Keengwe, J. & Arome, G. A. (2010, April). *Scalable intrusion detection with recurrent neural networks*. Paper presented at the 2010 Seventh International Conference on Information Technology, Las Vegas, NV.

Balram, S. & Wiscy, M. (2008, November). *Detection of TCP SYN scanning using packet counts and neural network.* Paper presented at the Fourth International Conference on Signal Image Technology and Internet Based Systems, Bali, Indonesia.

Bhaskar, T., Kamath, N. & Moitra, S. D. (2008). A hybrid model for network security systems: Integrated intrusion detection system with survivability. *International Journal of Network Security*, *7*(2), 249-260.

Bi, J., Zhang, K. & Cheng, X. (2010, March). *A new method of data processing in the intrusion detection system with neural network.* Paper presented at the Second International Workshop on Education Technology and Computer Science, Wuhan, Hubei, China.

Chandola, V., Banerjee, A. & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 4*(3), 15:1-15:58.

Choudhary, A. K. & Swarup, A. (2009, November). *Neural network approach for intrusion detection.* Paper presented at the Second International Conference on Interaction Sciences: Information Technology, Culture and Human, Seoul, Korea.

Ding, S., Su, C. & Yu, J. (2011). An optimizing BP neural network algorithm based on genetic algorithm. *Artificial Intelligence Review, 36*(2), 153-162.

Ding, A., Xu, A., Zhu, H., Wang, J. & Jin, F. (2011). Studies on optimization algorithms for some artificial neural networks based on genetic algorithm (GS). *Journal of Computers, 6*(5), 939-946.

Engen, V., Vincent, J. & Phalp, K. (2008). Enhancing network based intrusion detection for imbalanced data. *International Journal of Knowledge-based and Intelligent Engineering Systems, 12,* 357-367.

Engen, V., Vincent, J. & Phalp, K. (2011). Exploring discrepancies in findings obtained with the KDD Cup ’99 data set. *Intelligent Data Analysis, 15*, 251-276.

Farid, D. M., Darmont, J., Harbi, N., Hoa, N. H. & Rahman, M. Z. (2009). World Academy of Science, Engineering and Technology, 60, 154-158.

Ganatra, A., Kosta, Y. P., Panchal, G. & Gajjar, C. (2011). Initial classification through back propagation in neural network following optimization through GA to evaluate the fitness of an algorithm. *International Journal of Computer Science & Information Technology, 3*(1), 98-116.

Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G. & Vazquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security, 28*, 18-28.

Gu, Y., Shi, Y. & Wang, J. (2012). Efficient intrusion detection based on multiple neural network classifiers with improved genetic algorithm. *Journal of Software,* 7(7), 1641-1648.

Guangjun, S., Jialin, Z. & Zhenlong, S. (2008, June). *The research of dynamic change learning rate strategy in back propagation neural network and application in network intrusion detection*. Paper presented at the Third International Conference on Innovation Computing Information and Control, Dalian, Liaoning, China.

Hua, J. & Xiaofeng, Z. (2008, December). *Study on the network intrusion detection model based on genetic neural network.* Paper presented at the 2008 International Workshop on Modeling, Simulation and Optimization, Hong Kong, China.

Ibrahim, L. M. (2010). Anomaly network intrusion detection system based on distributed time-delay neural network (DTDNN). *Journal of Engineering Science and Technology, 5*(4).

Joo, D., Hong, T. & Han, I. (2003). The neural network models for intrusion detection system based on the asymmetric costs of false negative errors and false positive errors. *Expert Systems With Applications*, *25*, 69-75.

Kabiri, P. & Ghorbani, A. A. (2005). Research on intrusion detection and response: A Survey. *International Journal of Network Security*, *1*(2), 84-102.

Kadeeban, S. S. & Rajesh, R. S. (2011). A genetic algorithm based elucidation for improving intrusion detection through condensed feature set by KDD99 dataset. *Information and Knowledge Management*, *9*(1).

Li, W. (2004, May). *Using Genetic Algorithm for Network Intrusion Detection*. Paper presented at the United States Department of Energy Cyber Security Group 2004 Training Conference, Kansas City, KS.

Li, Y., Fang, B., Guo, L. & Chen, Y. (2007, March). *Network anomaly detection based on TCM-KNN algorithm*. Paper presented at the 2nd ACM Symposium on Information, Computer and Communications Security, Singapore.

Mafra, P. M., Moll, V., Fraga, J. DaS & Santin, A. O. (2010, June). *Octopus-II Intrusion detection system: An anomaly based intelligent intrusion detection system.* Paper presented at the IEEE Symposium on Computers and Communications, Riccione, Italy.

Moradi, M., & Zulkernine, M. (2004, November). A neural network based system for intrusion detection and classification of attacks. In *Proceedings of the 2004 IEEE international conference on advances in intelligent systems-theory and applications*. Luxembourg-Kirchberg, Luxembourg, IEEE Press.

Ray, L. L. (2014). Improving Performance and Convergence Rates in Multi-Layer Feed Forward Neural Network Intrusion Detection Systems, *International Journal of Strategic Information Technology and Applications 5*(3). p. 24-36.

Ray, L. L. (2013a, June). Determining the Number of Hidden Neurons in a Multi Layer Feed Forward Neural Network, *Journal of Information Security Research 4*(2). p. 63-70.

Ray, L. L. (2013b, June). Training and Testing Anomaly-based Neural Network Intrusion Detection Systems, *International Journal of Information Security Science 2*(2). p. 57-63.

Shum, J. & Malki, H. A. (2008, October). *Network intrusion detection system using neural networks*. Paper presented at the Fourth International Conference on Natural Computation, Jinan, China.

Smith, R. (2002). *Network-based intrusion detection using neural networks*. Unpublished master thesis, Rensselaer Polytechnic Institute, Troy, NY.

Tian, J. & Gao, M. (2009, April). *Network intrusion detection method based on high speed and precise genetic algorithm neural network*. Paper presented at the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing, Wuhan, Hubei, China.

Wang, H. & Ma, R. (2009, March). O*ptimization of neural networks for network intrusion detection*. Paper presented at the First International Workshop on Education Technology and Computer Science, Wuhan, Hubei, China.

Wei, Z., Hao-yu, W., Xu, Z., Yu-xin, Z. & Ai-guo, W. (2010, August). *Intrusive detection systems design based on back propagation neural network.* Paper presented at the Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Hong Kong, China.

WuS. X. & Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, *10*, 1-35.