A NEURAL NETWORK BASED DISTRIBUTED INTRUSION DETECTION SYSTEM ON CLOUD PLATFORM

Zhe Li[1], Weiqing Sun[2], Lingfeng Wang[1]

[1]Department of Electrical Engineering and Computer Science,
[2]Department of Engineering Technology,
University of Toledo, 2801 W. Bancroft St., Toledo, OH 43606, USA
Zhe.Li@rockets.utoledo.edu, Weiqing.Sun@utoledo.edu, Lingfeng.Wang@utoledo.edu

Abstract: Intrusion detection system (IDS) is an important component to maintain network security. Also, as the cloud platform is quickly evolving and becoming more popular in our everyday life, it is useful and necessary to build an effective IDS for the cloud. However, existing intrusion detection techniques will be likely to face challenges when deployed on the cloud platform. The pre-determined IDS architecture may lead to overloading of a part of the cloud due to the extra detection overhead. This paper proposes a neural network based IDS which is a distributed system with an adaptive architecture so as to make full use of the available resources without overloading any single machine in the cloud. Moreover, with the machine learning ability from the neural network, the proposed IDS can detect new types of attacks with fairly accurate results. Evaluation of the proposed IDS with the KDD dataset on a physical cloud testbed shows that it is a promising approach to detecting attacks in the cloud infrastructure.

Keywords: Distributed IDS; Neural network; Cloud security; Anomaly detection.

1 Introduction

Nowadays, cloud computing is known by more and more people due to its advantages such as high scalability, high flexibility and low operational cost. Cloud service users usually do not need to know how the cloud based software or platform runs; instead, they only need to send the requests to the cloud provider and then wait for the results, which is a much easier and more efficient way to access the needed computing resources [1]. However, there are several issues for the current cloud platforms. According to Ref. [2], security issues such as information leakage, unreliable data and unauthorized access are the most concerned problems by the majority of cloud users. Other issues such as stable operations, support systems and user friendliness have received less attention.

To address the security problem with the cloud, it is a natural choice to deploy a distributed IDS system on the cloud to protect the virtual machines (VMs) and virtual networks against potential attacks. The major issue with such a choice is that the IDS could overload some busy nodes in the cloud and slow down the detection efficiency if no special arrangements are made. On the one hand, the IDS should not use too many resources to affect the performance of the major computing tasks; On the other hand, the deployed IDS should detect attacks efficiently. Therefore, it is desirable to equip the distributed IDS with the flexibility feature in that it can dynamically adjust its architecture based on the real-time resource usage information across the cloud. Moreover, it is important for the IDS system to be capable of detecting unknown (new) attacks in the cloud. Hence, anomaly detection will be more suitable, but it can be more demanding for resources [3, 4]. Thus, a balance needs to be achieved to satisfy cloud customers as well as provide the reasonable performance of intrusion detection simultaneously.

Some approaches have been proposed to address the security issues in the context of cloud computing. A multiple dimensional result [5] has been presented by using an artificial neural network (ANN) based approach. The work was based on a single machine instead of the cloud platform. In Ref. [6], the authors presented an immune system in both anomaly and misuse detection methods and compared the two methods. The immune system is based on the combination of positive and negative characterizations which come from several features defined as normal or abnormal states. A trusted agent based approach was proposed in Ref. [7], which determines whether a machine in a network is malicious based on the experiences and its previous operations. In Ref. [8], Vieria and Schulter proposed an ANN based function to realize an IDS on the cloud, and a feed-back structure ANN is used to create a behavior-based system and an expert system to build a knowledge-based system. And in Ref. [9] the authors concentrated on alleviating the network traffic when realizing an IDS based on a MapReduce framework.

Here a distributed IDS architecture is proposed which consists of nodes running backpropagation (BP) based ANNs on the cloud platform. By design, it is expected to achieve better flexibility, scalability and performance. The proposed IDS system has two main characteristics:

978-1-4673-1857-0/12/$31.00 ©2012 IEEE

1) It has a flexible distributed architecture which could adjust its configuration based on real-time resource usage information to avoid overloading any node in the cloud.

2) It provides multiple dimensional results which could be used to not only recognize malicious activities but also find what malicious activities are taking place.

The remainder of the paper is organized as follows. In Section 2, the BP-based neural network is introduced. The design of ANN based intrusion detection in a cloud environment is detailed in Section 3. The implementation of the proposed algorithm in a physical cloud experimental testbed is discussed in Section 4, coupling with the related experimental results and analysis. Conclusions and future work are given in the final section.

2 Backpropagation (BP) algorithm based neural network

As shown in Figure 1 [10], the whole neural network is composed of three layers: input layer, hidden layer and output layer

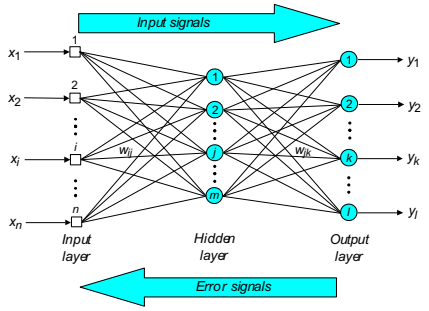

Figure 1 General architecture of the backpropagation algorithm based neural network

Now, we can regard a cloud as a lot of virtual machines which offer services to users. Each machine can be used to simulate a couple of nodes in a neural network so that several virtual machines in the same cluster will constitute a neural network. The following procedure will show how the ANN algorithm works. Here are some notations used in introducing the algorithm: *x*, *y*, *w* represent the input data, output result, weight value respectively, θ is correction needed only in the hidden and output layer, it will be continuously updated after each iteration, *e* is the error value, σ is error gradient and *p* is the number of iterations:

1) Initialize all the weights and threshold levels of the network to random numbers which are distributed inside a small range (*-2.4/Fi*, *2.4/Fi*), where *Fi* is the total number of inputs of a neuron *i* in the network.

2) Calculate the outputs of the neurons in the hidden layer:

$$y_j(p)=\text{sigmoid}\sum_{i=1}^{n}[x_i(p)*w_{ij}(p)-\theta_j]$$

where *n* is the number of inputs of neuron *j* in the hidden layer, and *sigmoid* is sigmoid activation function (sigmoid(s)=$1+\frac{1}{e^{-s}}$, here *e* is the base of the natural logarithm).

3) Calculate the actual outputs of the neurons in the output layer:

$$y_k(p)=\text{sigmoid}\sum_{j=1}^{m}[x_{jk}(p)*w_{jk}(p)-\theta_k]$$

where *m* is the number of inputs of neuron *k* in the output layer.

4) Calculate the error gradient for the neurons in the output layer:

$$\sigma_k(p)=y_k(p)*[1-y_k(p)]*e_k(p)$$

where $e_k(p)=y_{d,k}(p)-y_k(p)$. $y_{d,k}$ is the desired output value.

5) Calculate the weight corrections:

$$\Delta w_{jk}(p)=\alpha*y_j(p)*\sigma_k(p)$$

then update

$$w_{jk}(p+1)=w_{jk}(p)+\Delta w_{jk}(p)$$

where α is termed learning rate.

6) Calculate the error gradient for the neurons in the hidden layer:

$$\sigma_j(p)=y_j(p)*[1-y_j(p)]\sum_{k=1}^{I}w_{jk}(p)*\sigma_k(p)$$

7) Calculate the weight corrections:

$$\Delta w_{ij}(p)=\alpha*x_i(p)*\sigma_j(p)$$

then update

$$w_{ij}(p+1)=w_{ij}(p)+\Delta w_{ij}(p)$$

8) Increase iteration *p* by 1, go back to step 2 and repeat the process until the selected error criterion is satisfied.

The trained ANN acquired the knowledge of normal activities and attacks for performing anomaly detection tasks. In our research, KDD database is used in the training phase. For each network connection, 41 different quantitative and qualitative features were extracted. So after training, ANN learns what all the feature values are like in normal activities and in various attack scenarios. When any event is coming into the network, it will be treated as at least 41 input values corresponding to 41 different features of the event, then all these inputs will pass through the hidden layer and output layer in the ANN, the output node will get the result. When a malicious activity is detected, the output layer will raise an alarm and disallow the malicious activity. Every activity followed will be recorded in case the origin of the attack needs to be tracked. The supervisor of the cloud will fetch this information from the output layer. The output layer resides in the cluster leader machine, and the leader is the only machine which is allowed to communicate with the outside world. These 41 dimensional vectors make the detections more accurate in the complex cyber environment.

3 System architecture

We used Ubuntu Enterprise Cloud (UEC), which is Ubuntu's Eucalyptus-powered cloud platform, to build the cloud on our servers. Eucalyptus is one of the most popular cloud platforms which is well developed and feature-rich. Also it is designed to provide an Amazon EC2 compatible API. The Eucalyptus cloud platform is composed of five major components as shown in Figure 2:

1) The CLC (Cloud controller) is used to manage the underlying virtualized resources.
2) The Walrus provides an S3-like service to perform scalability and access control of virtual machines.
3) The CC (Cluster controller) controls the whole cluster by managing executions and networking.
4) The SC (Storage controller) handles storage in a cluster.
5) At least one NC (Node controller) controls activities in VM instances.

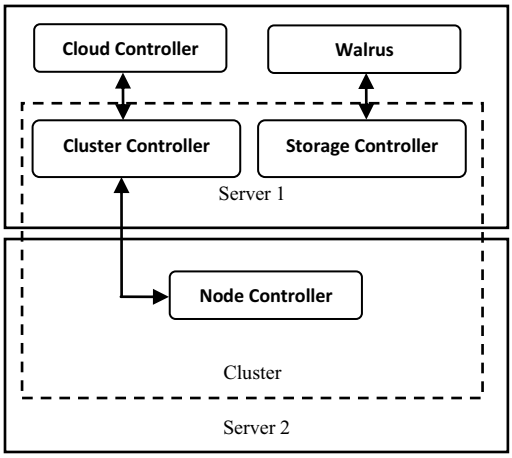

Figure 2 Architecture of the cloud

Based on this cloud platform, the ANN-based IDS will be established. In the architecture, there is one manager VM and multiple worker VMs in the network. The manager VM monitors the load information for the worker VMs and decides the mapping of ANN on the worker VMs dynamically. That is, those worker VMs having certain amounts of resources available will be chosen to perform the intrusion detection task, and the worker VMs are assigned to the input layer, hidden layer and output layer to form an ANN.

The input layer in the proposed ANN structure is responsible for collecting data from the network. All the requests or data flow in the network should first be collected by those nodes and then be passed through the whole neural network for any malicious activities. The hidden layer receives the raw data from the input layer and processes them based on the ANN mechanism discussed in Section 2, and forwards the results to the output layer. This layer will also modify weight values of the input layer after each iteration and pass those updated values to the input layer. The output layer derives the final result based on the intermediate results received from the hidden layer. It also updates weight values for the hidden layer and sends them to the hidden layer to improve the overall network behavior.

As mentioned previously, the architecture shown in Figure 3 is proposed for improving the system flexibility, which is also important to enhance the robustness of IDS [11-13]. When one node in the IDS is unavailable due to situations such as deadlock, poweroff, and scarce resources, the IDS is able to adjust itself accordingly to form a new capable architecture.

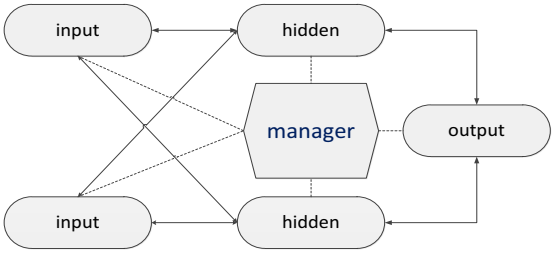

Figure 3 Architecture of the proposed IDS

Figure 4 shows the process flow for the multi-threaded manager process. When a client joins the IDS, it will raise a thread and connect to the server, the server will then store the thread into the queue with the address and port number. Once the network connection is established, all the clients will send the resource usage information periodically to the manager so as to select the most appropriate nodes to construct the IDS. After the IDS is built, all the other IDS nodes will receive the message from the manager and run the corresponding (input, hidden, output or wait) function based on the conditional statement. In addition, the IDS nodes will update the resource usage information to the manager every 10 seconds, and all other nodes will do the same every 10 minutes.

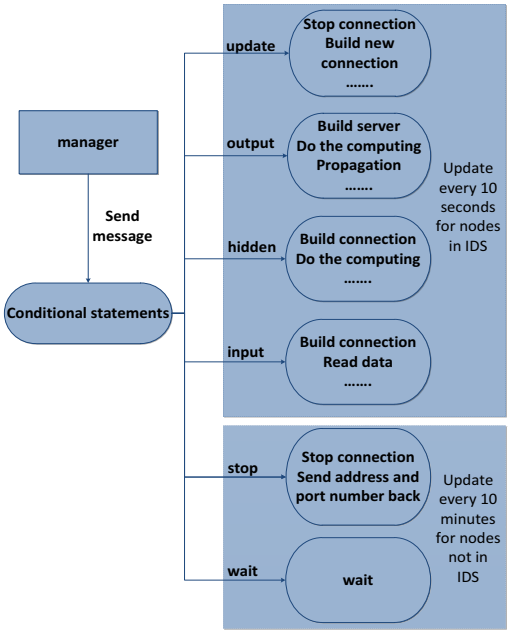

Figure 4 Process flowchart for the manager

When some nodes in the IDS become unavailable (busy or power off), the manager will be informed of this

event within 10 seconds based on the system design. The manager will then choose new nodes based on the most recent resource usage information. It will send messages to stop the old connections and ask to build new ones. Thus, the whole IDS could continue to function. Also, when some nodes outside the IDS become unavailable, the manager will be notified of this change within 10 minutes. So the manager will not choose them as candidates for IDS nodes. Further, the structure of the IDS can be adjusted through the manager, which sends messages to the candidate nodes to build a new IDS structure as requested. All the models are trained off-line before they are deployed.

4 Experimental results

Several experiments were performed to demonstrate the effectiveness of the proposed IDS. As shown in Figure 5, the servers used to build the cloud platform are two Dell PowerEdge R710 server machines and one Dell PowerEdge R610 server machine with Quad-core Intel® Xeon® CPU, 20 GB RAM and 500GB hard disk. 30 virtual machines were created each with 512MB RAM to emulate a cloud environment.

Figure 5 The experimental cloud testbed based on Dell® PowerEdge® R710 and R610 Servers

The first step in the experiment is to train the neural network. As a flexible IDS is desired, three different models are pre-trained (the number of models could be more depending upon the specific applications) by 10 percent of KDD dataset with respective 3, 5, 7 nodes to achieve the intrusion detection function. The IDS receives instructional signals from the manager, and then forms the corresponding architecture for the intrusion detection tasks. Table I shows the performance results for different models/structures of the IDS in terms of training time, detection time and detection accuracy. The numbers reported are the average across 10 runs. As can be seen from the table, the average detection accuracy is around 99% for all the three models, and the training detection time increases as the number of IDS nodes increases because of additional communication overhead.

Here we chose the 5-node architecture as an example and some resultant experiment results will be discussed. In this 5-node neural network, the ANN is distributed as a 2-2-1 structure, which means there are 2 nodes in both input and hidden layers and 1 node in the output layer. The learning rate chosen is 0.1 and correction in the two hidden node machines are 0.8 and -0.1 respectively and correction in the output node machine is 0.3.

Table I Performance results of different IDS models

Model	Structure	Average training time	Average detection time	Average detection accuracy
3	1-1-1	5m3s	20.73s	98.3%
5	2-2-1	5m45s	36.95s	99%
7	3-3-1	6m35s	53.62s	99.7%

In the input layer, those weight values keep changing in the training phase to adapt to the training data so that the performance of the whole IDS will be constantly improved. As there are 41 inputs (based on KDD dataset, every data flow in the network has 41 different feature values), after the training phase each input will have a corresponding weight value. In total 41 weights will be saved in the input layer, which are ready to be used to conduct the detection task.

In the output layer, the iteration numbers and the error between the real result and desired result can be obtained. In the whole training process, it was found that though the error value did not keep decreasing after each single training circle, the general trend did decrease which means the IDS performance is being improved. Technically speaking, when the error becomes less than 0.001, the ANN is considered ready to be used.

The total time consumed in the training phase is between 5 to 6 minutes. As an example, Table II below shows the value change in one of the 41 input weights (W1), one of the hidden weights (H1) and the error value in the output layer (Error).

Table II Results of the training phase

	Before training	During training	After training
W1	0.028706280142065916	0.028726335 91204304	0.0287554975 56997633
H1	0.216313726 1439795	0.216286438 9576207	0.2162630008 1037786
Error	0.247077420 93260648	5.738612462 453663E-4	1.0171468101 10198E-15

Following the training phase, we tested the ANN performance using KDD no-label dataset and corrected dataset. From the experimentation results, it was found that every different state corresponds to a small range of values. Thus, according to the value obtained from the output layer, we can not only determine whether there are malicious actions but also know what kind of attack is transpiring. The sample results from a test are shown in Figure 6, and it can be seen that the IDS is able to classify every abnormal activity and normal activity without any wrong detection. The accuracy is 100% in this case. But it does not mean the accuracy can always be this high. Initial values like weights, corrections, learning rate are randomly generated and picked so every time when the ANN is trained, different results and value intervals of the system states may be yielded.

According to all the tests carried out, a 99% or higher accuracy can be oftentimes achieved.

Figure 6 A screenshot demonstrating the testing results

To illustrate the result more clearly, Table III below is used to show multiple dimensional results with different value intervals based on the same test shown in Figure 6.

Table III Value intervals of different states

Value interval	State
0.17~0.181	Ipsweep
0.062~0.067	Normal
0.81~0.82	Snmpgetattack
0.093~0.094	Xsnoop
0.048~0.051	Satan
0.024~0.027	smurf or xterm
0.61~0.63	Warezmaster
0.076~0.079	apache2
0.182~0.185	neptune or back
0.156~0.159	Nmap
0.41~0.43	Portsweep
0.037~0.039	Xlock
>1	buffer_overflow
0.71~0.73	guess_passwd
0.29~0.31	Multihop
0.25~0.27	Mailbomb
0.055~0.057	Mscan
0.067~0.069	Named

Only two sets of states (i.e., smurf and xtem, Neptune and back) fall in the overlapped intervals. So the overall performance is satisfactory. We also evaluated the recovery cost when a number of IDS nodes become unavailable. The results shown in Table IV are based on the 7-node model with a 3-3-1 architecture.

Table IV Recovery time for proposed IDS

Number of unavailable nodes	Average recovery time
1	3.4s
2	5.2s
3	6.5s

5 Conclusions and future work

In this paper, a neural network based IDS is built on a cloud platform. The accuracy of the implemented IDS is shown to be high and the time expense is acceptable. Implementation of the neural network in the cloud is a promising direction. There is still much room left for further improving the current work. For example, the KDD dataset used is based on every message passing through a single machine in the network. In fact, there are various ways to attack a network by compromising several machines simultaneously [14]. So an enhanced algorithm should be developed to detect those kinds of attacks. Also, the anomaly detection algorithm can be further enhanced by adding misuse detection functions. The idea is to build an expert database to achieve knowledge based detection.

References

[1] Ramgovind, S. Eloff and M.M. Smith, E., "The management of security in Cloud computing", in Information Security for South Asia, 2010, pp. 1-7.

[2] M. Okuhara, T. Shiozaki, T. Suzuki, Security architectures for cloud computing, FUJITSU Sci. Tech. J., vol. 46, no. 4, (2010) October, pp. 397-402.

[3] A. K. Ghosh and A. Schwartzbard, a study of using neural network for anomaly and misuse detection, Proceedings of the 8th USENIX Security Symposium, page 12, Washington, D.C., USA, August, 1999.

[4] W. Lee and D. Xiang, Information-Theoretic Measures for Anomaly Detection, Proceedings of 2001 IEEE Symposium on Security and Privacy, page 130.

[5] S. Mukkamala, G. Janoski, and A. Sung, Intrusion Detection Using Neural Networks and Support Vector Machines, Neural Networks, Proc. of the 2002 International Joint Conference, pp. 1702-1707.

[6] D. Dasgupta and F. Gonz´alez， An Immunity-Based Technique to Characterize Intrusions in Computer Networks, IEEE Transactions on Evolutionary Computation, 6(3), pp. 1081-1088, June 2002.

[7] S. Pal, S. Khatua, N. Chaki, and S. Sanyal, A New Trusted and Collaborative Agent Based Approach for Ensuring Cloud Security, Annals of Faculty Engineering Hunedoara International Journal of Engineering, Vol. 10, Issue 1, February, 2012.

[8] K. Vieira, A. Schulter, C. Westphall, and C. Westphall, "Intrusion detection techniques in grid and cloud computing environment," IT Professional, vol. 99, 2009.

[9] M. D. Holtz, B. M. David, and R. T. de Sousa Junior, "Building Scalable Distributed Intrusion Detection Systems Based on the MapReduce Framework", REVISTA Telecomunicacoes, no. 2, pp. 22-31, 2011.

[10] N. Michael, Artificial Intelligence - A Guide to Intelligent Systems-2nd edition, Addison Wesley, 2005.

[11] V. Kotov, V. Vasilyev, "A Survey of Modern Advances in Network Intrusion Detection", 13th International Workshop on Computer Science and Information Technologies (CSIT'2011), pp. 18-21, 2011.

[12] P. Guan and X. Li, "Minimizing distribution cost of distributed neural networks," Scalable Software Systems Laboratory, Department of Computer Science, Oklahoma State University, Stillwater, pp. 1-5, 2007.

[13] Y. Chen, V. Paxson, and R. Katz, "What's New About Cloud Computing Security?" Technical Report No. UCB/EECS-2010-5.

[14] S. Bharadwaja, W. Sun, M. Niamat, F. Shen, Collabra: A Xen Hypervisor based Collaborative Intrusion Detection System, Eighth International Conference Information Technology: Next Generations, pp. 695-700, 2011.