A Thesis

Entitled

A Neural Network Based Distributed Intrusion Detection System on Cloud Platform

By

Zhe Li

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Master of Science Degree in Engineering

_____

Dr. Lingfeng Wang, Committee Chair

_____

Dr. Weiqing Sun, Committee Co-Chair

_____

Dr. Richard Molyet, Committee Member

_____

Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo

May 2013

An Abstract of

A Neural Network Based Distributed Intrusion Detection System on Cloud Platform

By

Zhe Li

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Masters of Science Degree in Engineering

The University of Toledo

May 2013

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 thesis 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.

Acknowledgements

This work would not be possible without the help and support of several people.

First, I would like to thank Dr. Lingfeng Wang and Dr. Weiqing Sun for their guidance. Their attitude towards research and their insights in this field inspired me to reach a new academic level. They taught me how to solve problems and analyze them in an effective way. I believe the experience of being one of their students will have a big influence in my life.

I would like to thank Dr. Richard Molyet for agreeing to serve on my committee.

I would also like to thank all the people who helped me to finish this project.

Finally, I would like to express my gratefulness and respect to my parents and friends. Their encouragement are responsible for all that I have and will achieve.

Contents

List of Tables

List of Figures

List of Abbreviations

ANN ……. Artificial Neural Network
API ..... Application Program Interface

IDS …….... Intrusion Detection System

UEC …….... Ubuntu Enterprise Cloud

VM ……..... Virtual Machine

Chapter 1

1 Introduction

1.1 Background and Motivation

Cloud computing is developed based on the increasing demand of Internet using, interacting and other related aspects; it usually involves providing dynamic expand Internet service by virtualized resources. The cloud is a metaphor for describing networks or Internet. In the past pictures of clouds are often used to represent telecommunication network, and then it is also used to refer to abstraction of the Internet and the underlying infrastructure. The narrow cloud computing concept refers to the rent and use mode of IT infrastructure. It indicates the needed resources are obtained through the network, based on rules like on-demand and easy to expand; Generalized cloud computing refers to the rent and use mode of computing. This kind of service can be IT, software, Internet related, or other services. It means computational ability can be treated as a kind of commodity and be traded through the Internet just like other utilities such as water, gas, electricity

and so on. Cloud computing has the following main features:

1) Dynamic resource allocation. According to the real time demand of consumers, cloud could dynamically divide or release different physical and virtual resources. When a request is raised, cloud would match it rapidly by increasing the available resources to realize elasticity of resources. If the user no longer needs this part of the resources, it can release these resources for free. Thus, cloud computing is regarded as infinite resources combined together which realizes the scalability of IT resources.
2) On demand self-service. Cloud computing provides a self-service mode as resources service, users can get resources automatically without interacting with providers. At the same time, the cloud system provides a certain application service directory; the customer can select the right service to meet their own needs.
3) Convenience of network access. Users may access the network through different terminal equipment which makes the network accessible from anywhere.
4) Measurable service. In cloud computing, according to different types of services, resources can be automatically controlled and the allocation is optimized. It is a kind of pay-as-you-go service model.

5) Virtualization. With the virtualization technology, it is possible to reorganize computing resources distributed in different areas for realizing the sharing of infrastructure.

Nowadays, cloud computing is rapidly developed and 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 concerning 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 (Intrusion Detection System) on the cloud to protect the virtual machines (VMs) and virtual networks against potential attacks. An intrusion detection system is a piece of software that is usually used to monitor system performance to avoid unintended behaviors and send report to the manager [3]. Intrusion detection system is a kind of

network transmission real-time monitoring system. It would raise an alarm or take the initiative reaction to protect the network from attacks when suspicious actions are found. The differences between intrusion detection system and other network security systems is that IDS provides an effective way to keep the whole network safe. And there are two main methods to build an IDS.

The first method is intended for anomaly detection. Anomaly detection refers to the actions of looking up unexpected behaviors or data that do not meet the exist model. These behaviors or data that do not conform to the requirements of the model is often referred to as abnormalities, abnormal values, the disharmony of observation, exceptions, aberrations, surprise, peculiarities, or contaminants in different application areas [4]. In this approach, there is no established normal activities set into the IDS; instead, the IDS will be designed to learn what kind of actions are malicious and what kind of actions are normal based on a well-planned training program with plenty of data. The nature of entered data is an important aspect of any anomaly detection technology. General input data is always a collection of the data instance (also known as the object, record, point, vector, pattern, case, sample, observation, entity) [5]. The advantage of this method is it has the ability to explore new species of attacks; but on the other hand, it may cause a lot of inaccurate judgment, such as raising an alarm when the network is working normal or

ignoring an attack as it considers the attack as normal action. The second way to build an IDS is called signature-based detection which is dependent on a knowledge base. The signature-based detection method is very useful, because it is very effective to detect known threats through signatures of observed events to determine possible attacks [6]. This approach could accurately report and defend attacks which are already known in the knowledge base, but the disadvantage is it has a limited effect to new kinds of attack, and the knowledge base should be updated frequently to make sure the IDS has good performance.

In this thesis, the first way is chosen to build the IDS. The major issue with building it on a cloud platform 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 [7, 8]. Thus, a balance

needs to be achieved to satisfy cloud customers as well as provide the reasonable performance of intrusion detection simultaneously.

1.2 Existing IDS architectures and algorithms

In Ref. [9], the authors propose a hybrid intrusion detection system that combines K-Means. They use K-nearest neighbor and Naive Bayes as the two key factors for anomaly detection. An entropy based algorithm is used to select the important attributes and removes the redundant attributes. In Ref. [10], a misuse intrusion detection system is founded by a genetic algorithm that based on the knowledge of a set of intrusion behavior classification rules. An adaptive network intrusion detection system is shown in Ref. [11], which uses a two-stage architecture. In the first stage, some possible malicious connections in the traffic are detected by a probabilistic classifier. In the second stage, the authors try to minimize the possible IP addresses of attacks through an HMM based traffic model. In Ref. [12], the researchers propose a distributed IDS by using multi-agent methodology which is combined with accurate data mining techniques. Those intelligent agents are responsible for collecting and analyzing the network connections, and the performance is really good. The authors in Ref. [13] use evolution theory to explain the evolution of data and connections in the network and thus reduce the complexity. The

proposed Intrusion Detection System (IDS) is based on this theory. Ref. [14] discusses an IDS with a new alert clustering and analyzing facility. This mechanism could help all cooperating nodes get a better understanding of whole system which helps them to find false alarms and detect those damaged nodes in the system. Attacks in one node will spread to the network to alert other cooperating nodes to update themselves about new attack patterns. This will lead to early detection and prevention of attacks. In Ref. [15], the authors choose 19 key features to describe all the various network visits. Then they use a gradual feature removal method and combine it with a clustering method, ant colony algorithm and support vector machine (SVM) to build an intrusion detection system to determine whether a visit in the network is normal or not. It is shown in Ref. [16] that a high-throughput intrusion detection system (IDS) is represented. This IDS is based on a comparison architecture. It includes a bloom filter-based header comparison and parallel pattern matching method which means it can parallel sequence compare packet content with the Snort rules. Ref. [17] proposes an effective intrusion detection system. It uses a Particle Swarm Optimization (PSO) as a feature selection algorithm and a decision tree as a classifier. This would help accelerate the speed of detection and make the result more accurate. In Ref. [18], authors claim that the Hidden Naïve Bayes (HNB) is a data mining model that relaxes the Naïve Bayes method's conditional independence

assumption which could help to solve intrusion detection problems as it has attributes such as dimensionality, highly correlated features and large stream volumes.

1.3 Related Work

Some approaches have been proposed to address the security issues in the context of cloud computing. A multiple dimensional result [19] 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. [20], the authors presented an immune system based on 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. [21], which determines whether a machine in a network is malicious based on the experiences and its previous operations. In Ref. [22], 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. In Ref. [23] the authors concentrated on alleviating the network traffic when realizing an IDS based on a MapReduce framework. In Ref. [24], a framework of Collaborative Intrusion Detection

System is proposed to counter a variety of attacks, especially large-scale coordinated attacks. In the proposed system, there are actually more than one IDS in the cloud computing area, but the work is on one large and effective IDS. All the IDSs in the cloud share information such as new kinds of attacks and send alert to each other so that the users could get notifications in time. Thus, it could handle a variety of attacks even attacks of large-scale very well. And in Ref. [25], an IDS module consisting of Snort and a signature apriori algorithm is built which generates new rules from captured packets. After the new rules are generated, they will be stored in the Snort configuration file to improve the efficiency of Snort. This IDS exhibits a good performance in recognizing known attacks and the attacks that are deduced from the original knowledge. Most kinds of architectures of IDS today in the cloud environment are deployed on the network periphery of each guest OS, this architecture will increase the threat of attacks and give the hackers chance to compromise all guests from one compromised guest. In Ref. [26], a hybrid architecture for deployment of an intrusion detection system is shown. This architecture has a security mechanism on both sides of the cloud to ensure the "trustful score" is good so that no compromised guest exists in the cloud. In Ref. [27], the authors propose a Distributed, Collaborative, and Data-driven Intrusion Detection and Prevention system (DCDIDP), which aims to provide whole protection and detection for all the

cloud providers that work together in different layers. This system is composed of three layers: network, host and global so that it can be well distributed in different cloud providers. In Ref. [28], cloud intrusion detection with a new statistical waveform based classification is proposed, which records network connections in a period of time and then draws a waveform based on them. Then it analyzes the waveform to find the possible doubtful characteristics of the waveform and classifies intrusion type based on features of the waveform.

1.4 Objectives of Project

Based on the issues listed above, 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:

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.

After the IDS is built, it will be tested in different scenarios to see the performance of it, which gives the information of whether this IDS is good enough and how to improve it in the future.

1.5 Synopsis of Thesis

This thesis describes and discusses the creation and testing of the IDS on a cloud platform. It begins by reviewing a neural network based algorithm which is called backpropagation algorithm to provide the basic information, knowledge and ideas about how the IDS could distinguish normal or malicious actions so that it would work in cloud platform to defend attacks from network.

In the next step, the architecture of the IDS is designed, which includes details of how the IDS would react to different requests under different situations in complex network. This is done by first figuring out the concept of cloud and how a cloud works so that a cloud could be built right to develop an IDS on it. Then it is time to think about how many functional parts are needed in this IDS based on backpropagation algorithm.

The next thing to do is considering the real time contexts that could happen to a cloud and modify the IDS architecture gradually until it could react properly, corresponding to each of the contexts so that it could protect cloud in most cases. Next, a cloud is built using 4 Dell Poweredge servers, then IDS is embedded on it and several different scenarios are simulated to test it. After that, the test results are collected and analyzed to draw a conclusion of the research and then the probable future work is pointed out.

1.6 Outline of Thesis

The remainder of the thesis 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, coupled with the related experimental results and analysis. Conclusions and future work are given in the final section.

Chapter 2

2 Literature Review

2.1 Outline

This chapter consists of three parts. The first two present the knowledge of neural network and backpropagation algorithm, which are important to help understand this study. Backpropagation algorithm is one of the most popular neural network based algorithms that include feedbacks in the network. The third part indicates how to adjust backpropagation algorithm in IDS on cloud platform.

2.2 Neural Network Concepts and Applications

The computer can learn knowledge through experience, which is called machine learning. This learning process constitutes of learning by examples and learning by analogy. Machine learning researches are often independent from the practical applications. A researcher might develop a new classification method, and then compare

its performance (such as accuracy or AUC) with existing publicly available data set of classification models to assess its effectiveness [29]. With machine learning ability, a computer can automatically adapt itself to the complex environments, this machine learning ability can be improved along with the time increase and more cases study. At present the most popular machine learning algorithms are based on neural network or genetic algorithm. This thesis is committed to neural network.

Artificial neural networks are designed to simulate biological neural networks. Every neuron in the network is well programmed based on their properties and works together to solve artificial intelligence problems without creating a model of a real system [30]. A neural network can be defined as a system based on human brain structure model. Human brain is composed by intensive nerve cells with mutual communication capacity. These nerve cells are the basic units of the information processing mechanism of brain, which are called neurons. A normal human brain can contain nearly 10 billion neurons and 60 trillion connections and synapses between these neurons. The brain processes information through these neurons, the information processing ability of human brain is much faster and stronger than any computer existing today. Even though each neuron has a very simple structure, a large number of neurons together can form a huge and mature

processing mechanism. As is shown in figure 2-1 [31], a neuron has cell body, soma, a lot of fiber called dendritic, and a long fiber called axon.

Human brain can be thought as a highly complex, nonlinear, parallel information processing system. The information process and information store procedures in the brain are not completely separated, instead, they could happen simultaneously in the same neural network. In other words, in the neural network, these two procedures happen globally, not locally. Learning ability is the biggest characteristic of a biological neural network, based on this idea, computers are used to simulate biological learning processes so as to realize the functions that needed.

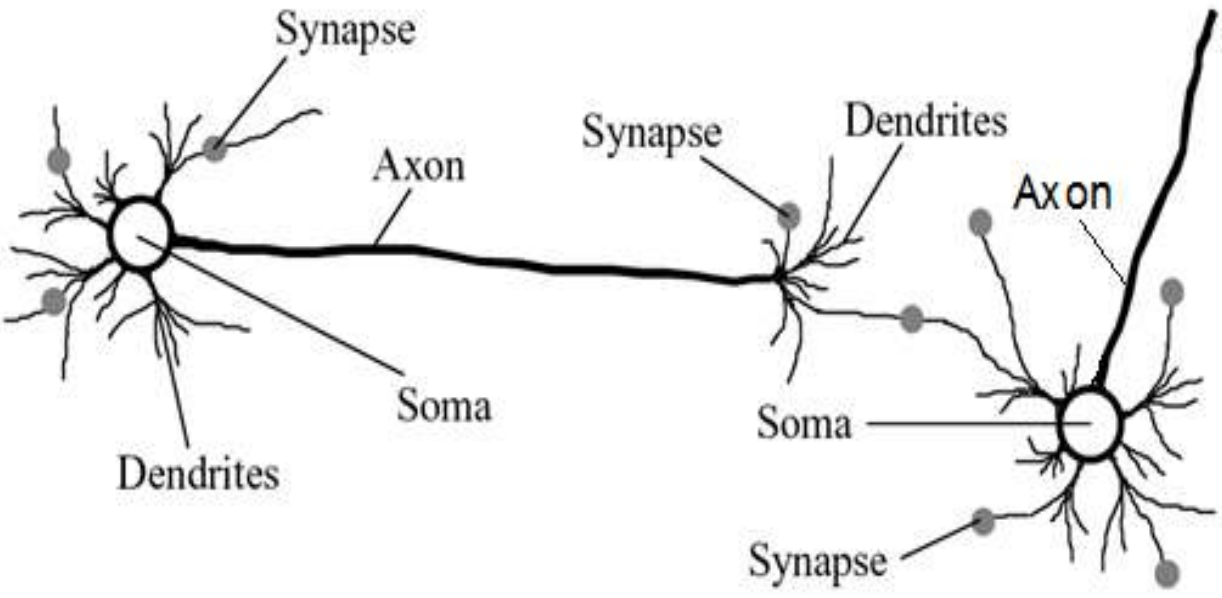

Figure 2- 1: Biological neural network [31]

An artificial neural network consists of a number of very simple processors which are called neurons that are similar to the biological neurons in the brain. Between neurons, there are weighted links combining them together as a whole network. The signals thus transfer from one neuron to another. The output signal of the neurons will be split into many branches that transmit the same signal. The outgoing branches terminate at the incoming connections of other neurons in the network.

Neural network has been widely used thus far. There are various kinds of algorithms and work based on neural network in computer science area that were applied to many different aspects. As long as there are multiple variables randomly coming and the user wants to determine or clarify something according to them, neural network would be a good choice due to its self-learning and organizing ability. Here are some instances. As shown in Ref. [32], an interesting research is done by inputting variables such as moisture, titratable acidity, free fatty acids, tyrosine, and peroxide value into an ANN and helping to develop the model of radial basis (exact fit) artificial neural network for estimating the shelf life of burfi stored at 30ºC; the output value of this model will be the acceptability, and the results turn out quite well. Another relevant job is done in Ref. [33] that presents the potential of Cascade Backpropagation algorithm based ANN

models in detecting the shelf life of processed cheese stored at 30 ºC. The authors accelerate learning in ANNs by using the Cascade backpropagation algorithm (CBA), and the Bayesian regularization algorithm was used for training the network. In Ref. [34], the authors describe a new approach to analyzing road images which often contain vehicles and extract license plate (LP) from natural properties by finding vertical and horizontal edges. An algorithm based on artificial neural network (ANN) is used for recognition of Korean plate characters in this paper. Ref. [35] introduces a radial basis function artificial neural network, in which a multilayer feed forward network is used to deal with hydrological data. In RBFANN, spread and center values are the model parameters which are estimated by inducing the suitable weight values. In Ref. [36], the authors are aiming at the problem of the uncertainty of the calorific value of coal; a soft measurement model for the calorific value of coal is proposed based on the RBF neural network. And combined with the thought of k-cross validation, the genetic algorithm constructed a fitness function to optimize the RBF network parameters. An BP based neural network method is distributed in Ref. [37] which is efficient for solving the traffic flow problem--a complex non-linear prediction of a large-scale system. This is effective because Neural Network Model has adaptive and self-learning ability. Also Ref. [38] shows classification of different types of targets (vehicles) in an Intelligent Transport

System. Supervised Artificial Neural Network is used as the soft computing tool for classification, and here targets are classified on the basis of returned energy to the Radar or Radar Cross Section (RCS) values taken at different aspect angles. In Ref. [39], the authors analyze and determine the factors influencing the lifeless-repairable spares consumption. They use the BP neural network to forecast the consumption and combine it with a genetic algorithm which could optimize the weights and thresholds of the BP neural network. In Ref. [40], the research aims to develop, reference image quality measurement algorithms for JPEG images, and to classify the image based on its quality an Elman neural network has been developed. A new approach using the Modular Radial Basis Function Neural Network (M–RBF–NN) technique is presented in Ref. [41] to improve rainfall forecasting performance coupled with appropriate data–preprocessing techniques by Singular Spectrum Analysis (SSA) and Partial Least Square (PLS) regression. Like these listed examples, IDS is also a promising direction to adopting the neural network approach. In this thesis, a backpropagation algorithm will be discussed in the next section, which is based on neural network and applied in IDS.

2.3 Backpropagation Algorithm

As shown in Figure 2-2 [31], the whole neural network is composed of three layers: input layer, hidden layer and output layer.

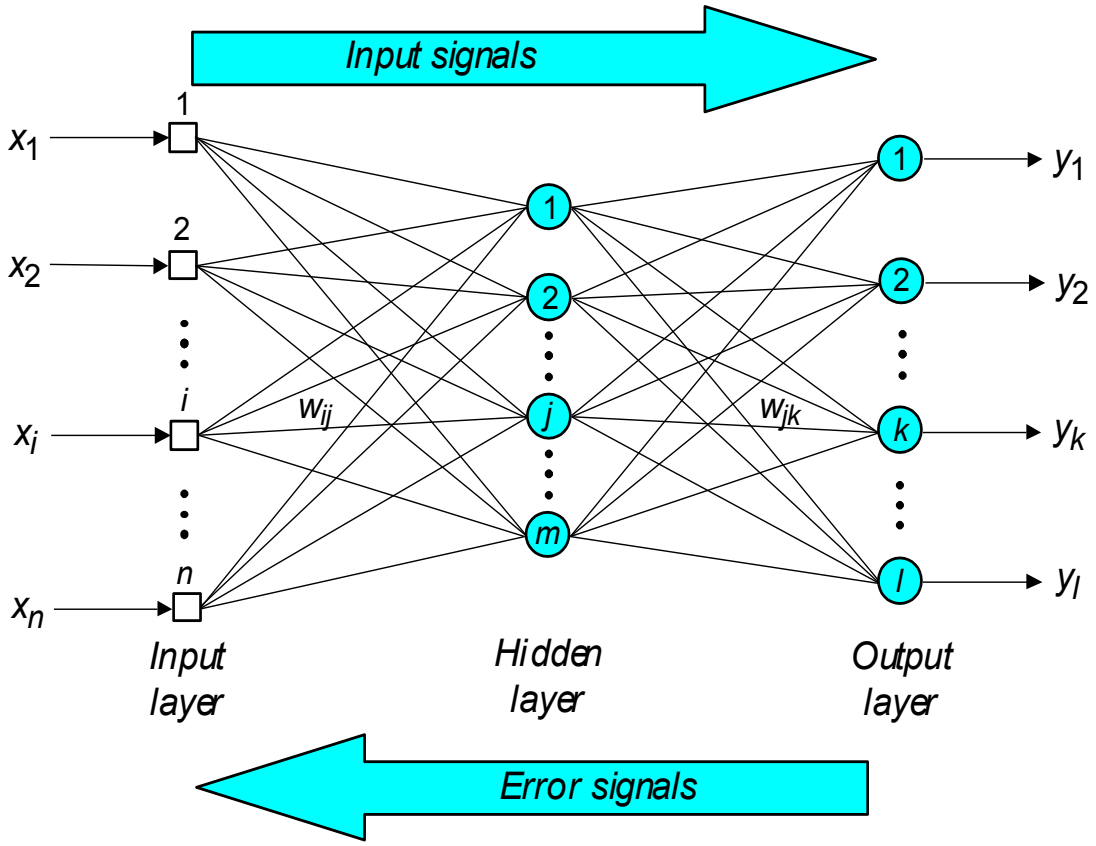

Figure 2- 2: General architecture of the backpropagation algorithm based neural network

The following procedure will show how the ANN (Artificial Neural Network) 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) Initialization, set all the weights and threshold levels of the network to random numbers uniformly 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 one, go back to step 2 and repeat the process until the selected error criterion is satisfied.

Next section includes a discussion about how to apply this algorithm in an IDS.

2.4 Backpropagation Algorithm in IDS

Now, a cloud can be regarded as many 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 trained ANN acquired the knowledge of normal activities and attacks for performing anomaly detection tasks. In this research, the KDD data package is used in the training phase. This data package is designed and collected by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military

network environment, was provided. In their design, for each network connection, 41 different quantitative and qualitative features were extracted. Here are some data samples that are contained in KDD:

'0', 'icmp', 'ecr_i', 'SF', '1032', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '511', '511', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.00', '255', '255', '1.00', '0.00', '1.00', '0.00', '0.00', '0.00', '0.00', '0.00', 'smurf.'

'0', 'tcp', 'http', 'SF', '307', '468', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '9', '14', '0.00', '0.00', '0.00', '0.00', '1.00', '0.00', '0.14', '255', '255', '1.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.00', 'normal.'

'0', 'tcp', 'http', 'SF', '54540', '8314', '0', '0', '0', '2', '0', '1', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '5', '5', '0.00', '0.00', '0.20', '0.20', '1.00', '0.00', '0.00', '220', '220', '1.00', '0.00', '0.00', '0.00', '0.00', '0.00', '0.06', '0.06', 'back.'

where the 42nd record determines the status of the connection.

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, and 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.

Chapter 3

3 System Architecture

3.1 Outline

At first, An introduction about cloud computing and some tools that are used to build cloud on the servers, such as Ubuntu Enterprise Cloud, Eucalyptus, and KVM are given. Then in the third subsection, details of architecture of the proposed IDS are presented to show the protecting mechanism and working principles of the intrusion detection system which was built for the cloud. The last two sections discuss about the programming language and database that are used to realize the IDS.

3.2 Cloud Computing

Cloud computing is the way to use computing resources as an utility. Both hardware and software can be delivered as a service over a network (typically the Internet) [42]. According to Ref. [43], there are two types of cloud: public cloud and private cloud.

A public cloud is designed to apply a pay-as-you-go manner to the general public to provide services. In Ref. [44], Adobe Creative cloud is a good instance of this: the users pay a certain amount to get the resources and tools they want. And a private cloud is usually used to deal with the inside data of an organization which are not open to the public. For example, Ref. [45] shows that the IBM smart cloud can provide private cloud service by giving threat protection for every layer of virtual infrastructure, limiting access to critical data, tracking user access and getting virtual infrastructure reports. Three classic services are provided in cloud computing right now [46]----HaaS, SaaS, DaaS. HaaS stands for hardware as a service which supports users to buy infrastructure in a pay-as-you-go manner so that users can get the computing resource they desire. A good example of this service is the Amazon EC2 cloud [47]. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides flexible compute resources in the cloud. It is designed to make web-scale computing easier for developers. SaaS means software as a service. In this mode, software or an application is hosted as a service and provided to customers across the Internet. Good examples of this service are Microsoft "Software+Service" cloud service [48] and the Google cloud platform [49]. The Microsoft cloud service called Windows Azure enables the user to quickly build, deploy and manage applications across a global network of Microsoft-managed datacenters. The

user can build applications using any operating system, language or tool. And the Google App Engine is a platform for traditional web applications in Google-managed data centers which support multiple languages. It handles deploying code to a cluster, monitoring, failover, and launching application instances as necessary. Developers have read only access to the file system on App Engine. DaaS denotes data as a service. Many applications such as Adobe Buzzwords are interested to use it to get data from the cloud wherever and whenever is necessary. Also, two kinds of open source software are popularly used to realize all these functions in cloud: OpenStack [50] and Eucalyptus [51]. Openstack aims to deliver solutions for all types of clouds by being simple to implement, massively scalable, and feature rich. Details of Eucalyptus are introduced later in this chapter because it is used to build cloud in this research.

3.3 Building the Cloud

As shown in Figure 3-1, the servers used to build the cloud platform are two Dell PowerEdge R710 server machines and two Dell PowerEdge R610 server machines with Quad-core Intel® Xeon® CPU, 20 GB RAM and 500GB hard disk. 45 virtual machines were created each with 256MB RAM to emulate a cloud environment. Ubuntu Enterprise Cloud is used to build the cloud platform, and KVM to create instances in the cloud.

Figure 3- 1: The experimental cloud testbed based on Dell® PowerEdge® R710 and R610 Servers

3.3.1 Ubuntu Enterprise Cloud

Ubuntu Enterprise Cloud (UEC) is a new type of open source tool powered by Eucalyptus proposed by the Ubuntu company. The UEC is used to further simplify the Eucalyptus based cloud infrastructure for its deployment, configuration, and use.

The following contents are the aspects that UEC has simplified:

1) Create the Eucalyptus public cloud that could run on the Amazon EC2 infrastructure.
2) Build a private cloud which could run on the internal data center infrastructure under the firewall.

So far, it is easy to install and use Eucalyptus. The users just need to download a CD version of the cloud server and install it in any place they want to. UEC is also the first open source item that allow the users to create cloud services in a local environment and then utilize the powerful functions of the cloud easily.

3.3.2 Eucalyptus

Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems (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 3-2:

1) The CLC (Cloud controller) is used to manage the underlying virtualized resources. This is the main controller that responsible for managing the whole

system. It is the main access entrance to the cloud for all users and administers. CLC will help transfer the requests to the right components, then collect the responses from those components and send them back to clients. CLC is the window which connects Eucalyptus cloud and outside world.

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. CC maintenance the information of all the related Node Controller that are running in the cluster, and is responsible for controlling life cycle of instances on those nodes. It will pass virtual instances' requests to the Node Controller with available resources.

4) The SC (Storage controller) handles storage in a cluster. SC and Walrus work together to store and access the virtual machine images, kernel images, RAM disk images and user data.

5) At least one NC (Node controller) controls activities in VM instances. NC controls operation system on the host machine and the corresponding hypervisor, such as KVM or Xen. In this thesis, KVM is used as the hypervisor.

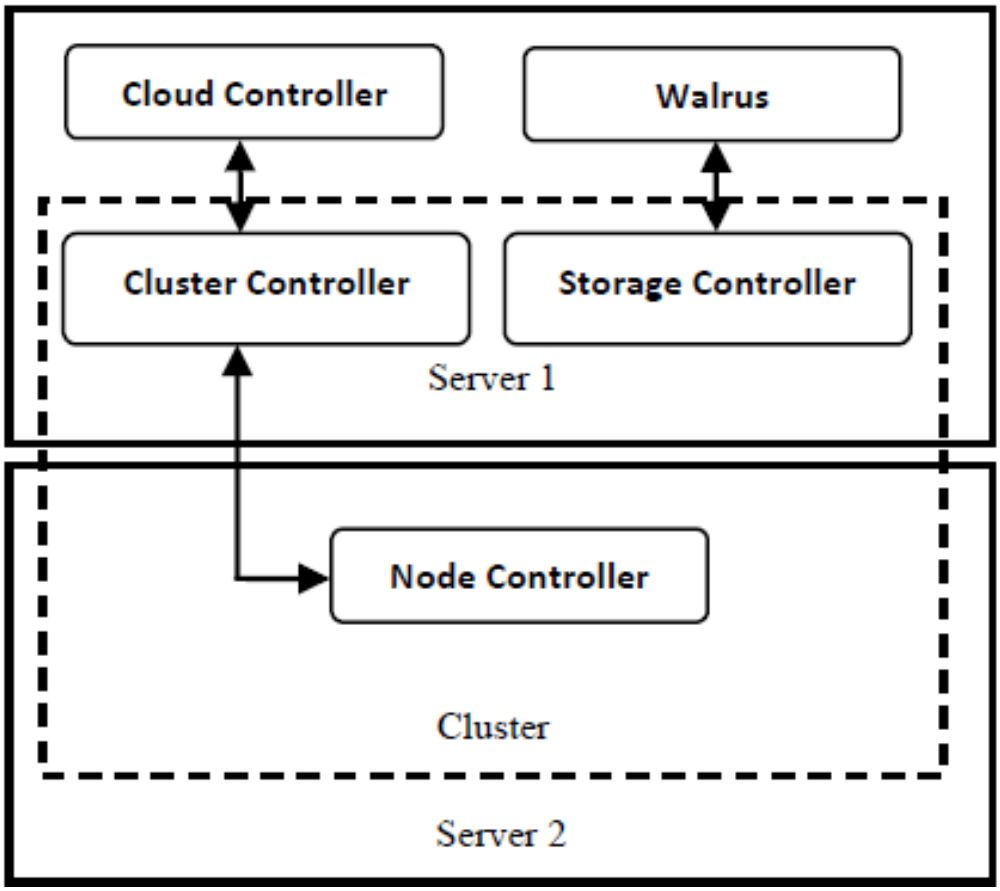

Figure 3- 2: Architecture of the Eucalyptus cloud

~/.euca/eucarc

euca-describe-availability-zones verbose

AVAILABILITYZONE	myowncloud	192.168.1.1			
AVAILABILITYZONE	|- vm types	free / max	cpu	ram	disk
AVAILABILITYZONE	|- m1.small	0048 / 0048	1	128	2
AVAILABILITYZONE	|- c1.medium	0048 / 0048	1	256	5
AVAILABILITYZONE	|- m1.large	0024 / 0024	2	512	10
AVAILABILITYZONE	|- m1.xlarge	0024 / 0024	2	1024	20
AVAILABILITYZONE	|- c1.xlarge	0012 / 0012	4	2048	20

Figure 3- 3: Availability of the cloud

Figure 3-3 shows the availability of the cloud that how many virtual machines it could support after a cloud is created successfully.

3.3.3 KVM

KVM is the abbreviation for Kernel-based Virtual Machine. It is an open source system virtualization module. It uses Linux's own scheduler to realize management, so when compared to Xen, its core source code is small. KVM has become one of the most popular options to build virtual machines in academic circles. The KVM virtualization needs hardware support (such as Intel VT technology or AMD SVM technology). It is the full virtualization based on hardware.

3.4 The Proposed IDS

Based on the cloud platform has been introduced above, 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 3-4 is proposed for improving the system flexibility, which is also important to enhance the robustness of IDS [52-54]. 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.

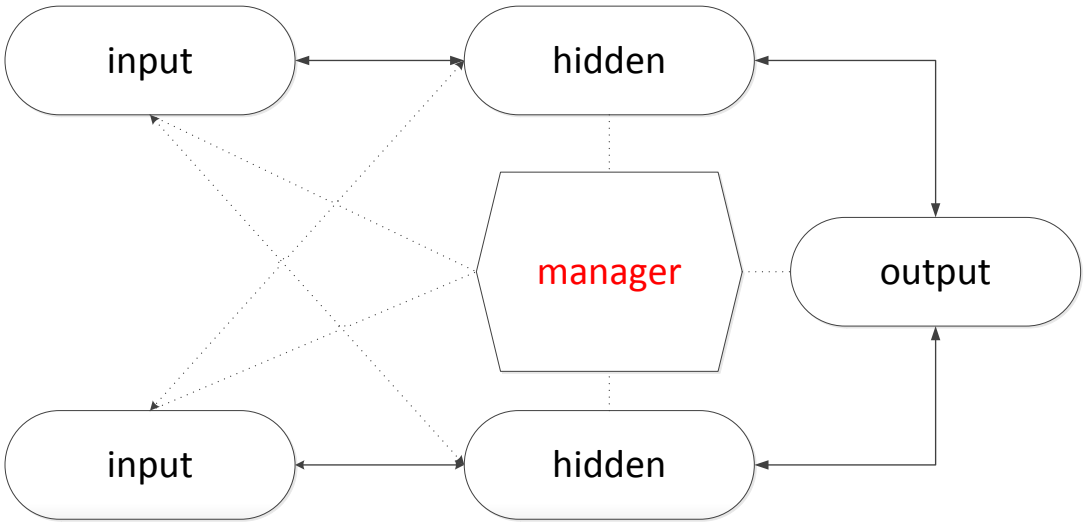

Figure 3- 4: Architecture of the proposed IDS

Figure 3-5 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.

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.

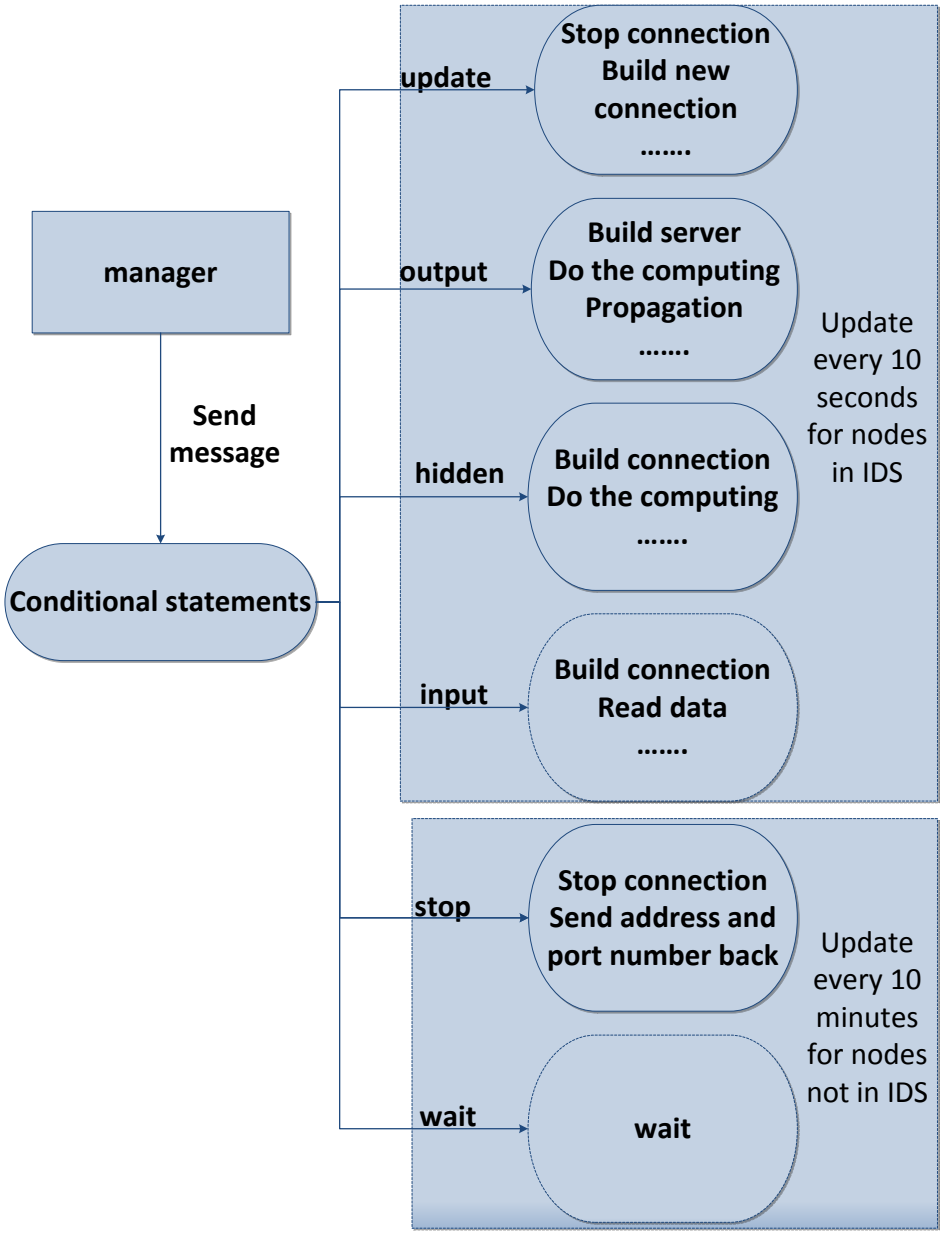

Figure 3- 5: Process flowchart for the manager

3.5 Network Programming

Network programming could be simply understood as communicating and transmitting data between different computers. To the programmers, it is important to

master a program interface and use it in a network programming model. Java SDK provides some relatively simple API to do this.

These API exist in the Java.net package. So as long as this package is imported, network programming can be started. The basic model of network programming is the Client-Server model. Simply put, it is a communication between two processes, and one of them has to provide a fixed socket, while the other one only needs to know this socket and to establish a connection between them. Then, data communication can be set up by using this connection. Here, the provider of the fixed socket is usually referred to as the server, and the connection-builder is often called the client. Based on this simple model, the network programming can be realized properly.

There are many APIs in Java which support this model. Here, only the Java-Socket programming is introduced which is used in the IDS. Java has quite a simple socket programming interface. First, the permanent socket provider and the server are introduced. Java provides the ServerSocket class to support this function. Actually, when the user creates an object of this class and provides the port number of the socket, the user, in fact, has already set up a permanent interface for other computers to let them access the server.

Here, let’s see a simple example of it:

“ServerSocket ssock1=new ServerSocket(30002);”

Every port is unique, which means that allocating the same port twice to different servers is not allowed. A port is a unique mark on each individual computer that provides different services. In addition, the port number is between 0 to 65535, and the first 1024 numbers have been reserved for Tcp/Ip connections, so the user has to assign port numbers after 1024.

Next, a link needs to be established. The link is always requested by the client first, and Java also provides a socket object to support this task. As long as the client creates an instance of socket, the link is built:

“ssock =new Socket("127.0.0.1",30000);”

As can be seen in the example, the client must know the server's IP address. Here, “127.0.0.1” means the server host is the same computer itself.

After the connection is built between computers, Input and Output streams are needed to receive or send data through the network. Two methods from socket: getInputStream() and getOutputStream() are needed. With all of the basic knowledge and methods introduced so far, a simple sample of socket programming could be built and tested.

But there is still a problem. When the server accepts the first request from a client, the client will occupy the port after the link is built, thus the subsequent clients can no longer connect to the same server. There will be an exception popped up when more than one client is running at the same time. So the server needs to be adjusted to fix this problem because in the design of the IDS, there will always be many clients running simultaneously. The way to solve this problem is to use the technology of "thread." Every time a server finds a connecting request from a client, it will assign the processes that deal with the request to an individual thread. That way the server can deal with multiple requests from clients.

3.6 Database

Database is an indispensable component in modern IT systems. Without advanced database technologies, tedious work will be needed, and some tasks would be difficult to accomplish, especially in big organizations such as banks, colleges and libraries.

In this work, as the KDD data package is used to test the IDS, those data in the KDD package have to be stored first into a database so that it could be read and then sent into the cloud to simulate network connections or requests in the network. Thus, MySQL database is chosen to realize it.

Generally speaking, MySQL has the following chief characteristics:

1) There is no limit on the number of clients that can access the database at the same time.
2) It can store more than 50,000,000 records.
3) It is one of the fastest database systems at present.
4) It is simple to set up the permissions of users.

Because here the IDS is used to provide protection for a Ubuntu Enterprise Cloud which is powered by Eucalyptus under the Ubuntu system, the database is deployed the under linux command-line environment. Below are the major steps:

First of all, MySQL should be installed by typing in a command like:

"sudo apt-get install mysql-server"

The second step is to log in by a command like:

"mysql -h hostname -u username -p[password]"

Then some information will be displayed on the screen like the following:

shell> mysql -h host -u user -p

*Enter password: ********

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 25338 to server version: 5.1.2-alpha-standard

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

When the above information is displayed, it indicates the user has logged in the database successfully, and can then start to use it. The instructions in mysql are the same as SQL database query language.

Chapter 4

4 Results and Discussion

4.1 Outline

In this chapter, the basic performance of the IDS based on some simple tests is given to examine the accuracy and time efficiency of the IDS. Then some detection results from a running IDS are presented and analyzed. After that, testing results from different scenarios are discussed to understand the proposed IDS more comprehensively and deeply.

4.2 Basic IDS Performance

The first step in the experiment is to train the neural network. As a flexible IDS is desired, six different models are pre-trained (the number of models could be more depending upon the specific applications) by the 10 percent of KDD dataset with respective 3, 5, 7, 13, 19, 45 nodes to achieve the intrusion detection function. Actually

there is a data package in KDD called “corrected” which not only has the 41 features of a network connection, but also has the “answer” indicating the status of this collection. This “corrected” package is the one used to train the IDS. After that, the IDS is tested with a package called “unlabel” which has the exactly same data as the “corrected” package in same order, but only it does not have the “answers” indicating the status of collections. As the “unlabel” package is used, the IDS can determine the status of the connections and compare them with the right “answers” in the “corrected” package so that knowledge can be gained to evaluate the performance. There are more than 300,000 records in these two packages. The IDS receives instructional signals from the manager, and then forms the corresponding architecture for the intrusion detection tasks. Table 4.1 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 the 5-node architecture is chosen 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 4. 1: Performance results of different IDS models

Number of damaged nodes	*Average recovery Time*
1	*3.6s*
2	*5.3s*
3	*6.1s*
10	*15.4s*

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 12 minutes. As an example, Table 4.2 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 4. 2: Results of the training phase

	Before training	During training	After training
W1	0.028706280142065916	0.02872633591204304	0.028755497556997633
H1	0.2163137261439795	0.2162864389576207	0.21626300081037786
Error	0.24707742093260648	5.738612462453663E-4	1.017146810110198E-15

4.3 Detecting Result of IDS

Following the training phase, ANN performance is evaluated 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, the IDS 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 4-1, 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 4- 1: A screenshot demonstrating the testing results

To illustrate the result more clearly, Table 4.3 is used to show multiple dimensional results with different value intervals based on the same test shown in Figure 4-1.

Only two sets of states (i.e., smurf and xtem, Neptune and back) fall in the overlapped intervals. So the overall performance is satisfactory.

Table 4. 3: 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

4.4 IDS Performance in different Scenarios

As can be seen in the performance table given in the previous section, the average training time and detection time increase when the model becomes larger. That maybe

due to two principal reasons: the first reason is that when there are more VMs involved in the IDS, there are more communications needed to be established which increases the running time of the system. And the second reason is the cross-server communications. Hence some more tests related to this topic are presented for trying to address this problem. Table 4.4 proves that communications between the same or different physical machines are not closely associated with the time increase. There is no big difference of time efficiency between the same structure with different distributions in physical servers. So the main reason for the time increase should be the communication cost itself.

Table 4. 4: Performance of IDS in different distributed structures

Model	*structure*			*Average training time*	*Average detect time*	*Average detect accuracy*
	Server 1	*Server2*	*Server3*			
3	***1***	***1***	***1***	***3m32.6s***	***12.64s***	***99%***
3	***1***	***1-1***		***3m39.0***	***13.44s***	***99.2%***
3	***1-1-1***			***3m35.1s***	***13.53s***	***99.1%***
5	***2***	***2***	***1***	***4m01.9s***	***26.65s***	***99.1%***
5	***2-2***		***1***	***3m57.2s***	***23.74s***	***99.7%***
5	***2-2-1***			***4m00.8s***	***24.75s***	***99.4%***
7	***3***	***3***	***1***	***4m21.8s***	***36.33s***	***99.5%***
7	***3***	***3-1***		***4m27.9s***	***37.28s***	***99.1%***
7		***3-3-1***		***4m25.0s***	***36.02s***	***98.9%***
15	***7***	***7***	***1***	***6m16.8s***	***83.58s***	***99.3%***
15		***7***	***7-1***	***6m20.2s***	***80.99s***	***99.6%***
15	***7-7-1***			***6m15.1s***	***82.20s***	***99.5%***

As can be seen in Table 4.5, when the CPU is very busy, the time required increases dramatically. For instance, in the case that the usage of CPU is 70%, 3-node structure spends 53.72s to accomplish the detection which is longer than 7- node structure spends in the case of 30% usage of CPU. This makes sense for distributing IDS in large scale some time since the IDS is distributed in larger structure by involving more VMs to avoid high CPU usage in each VM which dominates the time consumption more than the communication does.

Table 4. 5: Performance of IDS in different occupancy of CPU

Model	*Structure*	*Average Detect time*									
		Different VMs					*ALL in one VM*				
		idle	*30%*	*50%*	*70%*	*98%*	*Idle*	*30%*	*50%*	*70%*	*98%*
3	*1-1-1*	*13.53s*	*17.54s*	*24.20s*	*53.72s*	*134.33 s*	*10.04s*	*13.01s*	*17.81s*	*42.20s*	*101.10s*
5	*2-2-1*	*24.75s*	*32.08s*	*44.27s*	*98.64s*	*246.65 s*	*20.03s*	*39.35s*	*54.01s*	*120.34s*	*300.91s*
7	*3-3-1*	*36.02s*	*46.69s*	*64.43s*	*143.03s*	*357.58 s*	*39.23s*	*50.89s*	*70.23s*	*155.90s*	*389.76s*
13	*6-6-1*	*69.79s*	*90.45s*	*124.82s*	*277.1s*	*692.75 s*	*110.51s*	*142.91s*	*197.21s*	*437.82s*	*1094.55 s*
19	*9-9-1*	*102.58s*	*132.96s*	*183.48s*	*407.32s*	*1018.2 5s*	*252.28s*	*327.08s*	*451.36*	*1002.01 s*	*2504.90 s*
45	*22-22-1*	*248.68s*	*318.33s*	*439.29s*	*975.22s*	*2438.0 5s*	*983.85s*	*1260.58 s*	*1581.44 s*	*3510.79 s*	*8776.98 s*

Table 4. 6: Comparison of Performances of IDS deployed in different number of VMs

Model	*Structure*	*Average training time(different structure)*	
		Different VMs	*ALL in one VM*
3	*1-1-1*	*3m35.1s*	*1m36s*
5	*2-2-1*	*3m58.3s*	*3m02s*
7	*3-3-1*	*4m25.0s*	*4m56s*
13	*6-6-1*	*5m48.8s*	*9m21s*
19	*9-9-1*	*7m11.4s*	*17m40s*
45	*22-22-1*	*12m.58.8s*	*51m18s*

Table 4.6 shows different training time when the same structure is used but is distributed in only one VM compared with that when it is distributed in different VMs. And during training, usage of CPU is not a key factor.

When there are bigger data packages passing through the IDS, it doesn't affect the IDS a lot since the time spent with respect to the size of data package exhibits a linear growth. It is shown in the Table 4.7. So this result does not support the view for distributing the IDS in large scale in real time, although it was supported by the previous test results.

Table 4. 7: Performance of IDS with different size of large datasets

Model	*structure*	*Average Detect time*		
		100000 data	*200000 data*	*400000 data*
3	*1-1-1*	*13.53s*	*25.70s*	*46.26s*
5	*2-2-1*	*24.75s*	*47.02s*	*84.63s*
7	*3-3-1*	*36.02s*	*68.44s*	*123.20s*
13	*6-6-1*	*69.79s*	*132.60s*	*238.68s*
19	*9-9-1*	*102.58s*	*194.90s*	*350.82*
45	*22-22-1*	*248.68s*	*472.49*	*850.48s*

So in summary, when the cloud is not busy, communication is the most important factor that dominates the time usage; but if the VMs are very busy (usage of CPU higher than 50%), it seems that computation starts to dominate the time consumption. This also explains why when a lot of nodes are embedded in one VM, the training time exceeds that in the case when those nodes are distributed in different VMs (see the training time

of 7 or 13 nodes in table 4.6). Thus, it is not recommended to simulate more than 5 nodes in a single VM.

The proposed IDS is tested in several different scenarios. There are three points to be claimed here:

1) In an extreme case, when all the VMs are considered busy, the IDS uses every VM in the cloud while occupying fewer resources in each VMs than usual cases.
2) The system could handle somehow serious damage in the cloud with 10 VMs (1/4 of the total number) being unavailable. The recovery time is dependent on how many VMs are damaged. The more new VMs that need to be found to substitute the unavailable VMs, the more communications and new connections needed to establish. Therefore, more time will be consumed.
3) When a lot of VMs are busy, the manager prefers to make the IDS size smaller This is different from what it did in the extreme case when all the VMs are busy. However, when the size is too small such that all the VMs involved are busy, the manager starts to enlarge the IDS to involve more VMs so that each VM could contribute fewer resources to the IDS.

Table 4. 8: Recovery time for proposed IDS

Number of damaged nodes	*Average recovery Time*
1	*3.6s*
2	*5.3s*
3	*6.1s*
10	*15.4s*

The recovery cost is also evaluated when a number of IDS nodes become unavailable. The results shown in Table 4.8 are based on the 45-node model with a 22-22-1 architecture.

The more the new VMs needed to find to substitute the unavailable VMs, the more communications and new connections needed to establish; therefore, more time is consumed.

Chapter 5

5 Conclusion and Future work

5.1 Summary and Conclusions

Advanced soft computing and artificial intelligence methods/techniques are being used widely in Intrusion. Detection Systems (IDS) for acquiring the ability to learn and evolve, which makes them more accurate and efficient in the presence of enormous number of unpredictable attacks. In this thesis, 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 for intrusion detection is a promising direction.

5.2 Future Work

There is still much room left for further improvement of 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 such as by compromising several machines simultaneously [55] or starting an attack inside the network by a compromised node. So an enhanced algorithm should be developed to detect those kinds of attacks. Also, larger data sets and more complex and realistic scenarios should be developed and tested.

And there is another slight complication that ANN lacks in certain areas, which are detection precision for low frequent attacks and detection stability. So it is necessary to think about a way to address this problem [56]. 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] **Wikipedia**, http://en.wikipedia.org/wiki/Intrusion_detection_system.

[4] **Varun Chandola, Arindam Banerjee, Vipin Kumar**, Anomaly Detection : A Survey, ACM Computing Surveys, September 2009.

[5] **Tan, P.-N., Steinbach, M., and Kumar**, Introduction to Data Mining, Addison-Wesley, V. 2005, Charper 2.

[6] **Ankita Tuteja, Ravi Shanker,** Optimization of Snort for Extrusion and Intrusion Detection and Prevention, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 3, May-Jun 2012, pp.1768-1774.

[7] **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.

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

[9] **Om Hari, Kundu Aritra**, A hybrid system for reducing the false alarm rate of anomaly intrusion detection system, Information Technology (RAIT), 2012 1st International Conference on, pp.131-136.

[10] **Mayank Kumar Goyal, Alok Aggarwal**, Composing Signatures for Misuse Intrusion Detection System Using Genetic Algorithm in an Offline Environment, Intelligent Systems and Computing Volume 176, 2012, pp 151-157.

[11] **Karthick, R. Rangadurai**, Adaptive network intrusion detection system using a hybrid approach, Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference, pp.1-7.

[12] **Imen Brahmi, Sadok Ben Yahia, Hamed Aouadi, Pascal Poncelet**, Towards a Multiagent-Based Distributed Intrusion Detection System Using Data Mining Approaches, 7th International Workshop on Agents and Data Mining Interation, ADMI 2011, Taipei, Taiwan, May 2-6, 2011, pp. 173-194.

[13] **Mohammad Sazzadul Hoque, Md. Abdul Mukit, Md. Abu Naser Bikas**, An Implementation of Intrusion Detection System Using Genetic Algorithm, International Journal of Network Security & Its Applications, Volume 4, Number 2, pages 109 - 120, March 2012.

[14] **Deepa Krishnan, Madhumita Chatterjee**, An Adaptive Distributed Intrusion Detection System for Cloud Computing Framework, International Conference, SNDS 2012, Trivandrum, India, October 11-12, 2012. Proceedings, pp. 466-473.

[15] **Yinhui Li, Jingbo Xia, Silan Zhang, Jiakai Yan, Xiaochuan Aj, Kuobin Da**, An efficient intrusion detection system based on support vector machines and gradually feature removal method, Expert Systems with Applications Volume 39, Issue 1, January 2012, Pages 424–430.

[16] **Yi-Mao Hsiao, Ming-Jen Chen, Yuan-Sun Chu, Chung-Hsun Huang**, High-throughput intrusion detection system with parallel pattern matching, IEICE Electronics Express Vol. 9 2012, pp. 1467-1472.

[17] **Levent Koc, Thomas A. Mazzuchi, Shahram Sarkani**, A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier, Expert Systems with Applications Volume 39, Issue 18, 15 December 2012, Pages 13492–13500

[18] **Ahmed A. Elngar, Dowlat A. El A. Mohamed, Fayed F. M. Ghaleb**, A Fast Accurate Network Intrusion Detection System, International Journal of Computer Science and Information Security Volume 10, Issue 9, 2012, pp. 29-35.

[19] **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.

[20] **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.

[21] **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.

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

[23] **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.

[24] **Nguyen Doan Man, Eui-Nam Huh**, A Collaborative Intrusion Detection System Framework for Cloud Computing, Proceedings of the International Conference on IT Convergence and Security 2011, Lecture Notes in Electrical Engineering Volume 120, 2012, pp 91-109.

[25] **Chirag N. Modi, Dhiren R. Patel, Avi Patel, Muttukrishnan Rajarajan**, Integrating Signature Apriori based Network Intrusion Detection System (NIDS) in Cloud Computing, Procedia Technology Volume 6, 2012, Pages 905–912.

[26] **Sanchika Gupta, Susmita Horrow, Anjali Sardana**, A Hybrid Intrusion Detection Architecture for Defense against DDoS Attacks in Cloud Environment, 5th International Conference, IC3 2012, Noida, India, August 6-8, 2012. pp. 498-499.

[27] **Taghavi Zargar, Saman**, **Takabi, Hassan**, **Joshi, James B.D.,** DCDIDP: A Distributed, Collaborative, And Data-Driven Intrusion Detection And Prevention Framework For Cloud Computing Environments, CollaborateCom 2011, October 15-18, 2011.

[28] **Liu Yiming**, **Tseng Kuo-Kun, Pan Jeng-Shyang**, Statistical Based Waveform Classification for Cloud Intrusion Detection**,** Computing, Measurement, Control and Sensor Network (CMCSN), 2012 International Conference, pp 225-228.

[29] **Carla E. Brodley, Umaa Rebbapragada, Kevin Small, Byron Wallace**, Challenges and Opportunities in Applied Machine Learning, AI Magazine Vol 33, No 1, 2013.

[30] **Wikipedia**, http://en.wikipedia.org/wiki/Neural_network.

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

[32] **Sumit Goyal, Gyanendra Kumar Goyal**, Radial Basis (Exact Fit) Artificial Neural Network Technique for Estimating Shelf Life of Burfi, Advances in Computer Science and its Applications (ISSN 2166-2924) 93 Vol. 1, No. 2, June 2012.

[33] **Sumit Goyal , Gyanendra Kumar Goyal**, A Novel Method for Shelf Life Detection of Processed Cheese Using Cascade Single and Multi-Layer Artificial Neural Network Computing Models, ARPN Journal of Systems and Software, VOL. 2, NO. 2, February 2012.

[34] **Kaushik Deb, Ibrahim Khan, Anik Saha, Kang-Hyun Jo**, An Efficient Method of Vehicle License Plate Recognition Based on Sliding Concentric Windows and Artificial Neural Network, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[35] **K.S. Kasiviswanathan, Avinash Agarwal**, Radio Basis Function Artificial Neural Network: Spread Selection, International Journal of Advanced Computer Science, Vol. 2, No.11, pp. 394-398, 2012.

[36] **Yuan Jing, Minfang, Qi, Zhongguang, Fu**, Prediction of coal calorific value based on the RBF neural network optimized by genetic algorithm, Natural Computation (ICNC), 2012 Eighth International Conference, pp. 440-443, 2012.

[37] **Anuja Nagare, Shalini Bhatia**, Traffic Flow Control using Neural Network, International Journal of Applied Information Systems (IJAIS), Volume 1– No.2, January 2012.

[38] **Priyabrata Karmakar, Bappaditya Roy, Tirthankar Paul, Shreema Manna**, Target Classification: An application of Artificial Neural Network in Intelligent Transport System, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 6, June 2012.

[39] **Feng Guo, Su-qin Zhang, Deng-bin Zhang, Wei Gao**, Application of Genetic Neural Network on Lifeless-Repairable Spares Consumption Forecasting, Computer Science & Service System (CSSS), 2012 International Conference, pp.1313-1315, 2012.

[40] **Paulraj M. P, Mohd Shuhanaz Zanar Azalan, Hema C.R., Rajkumar Palaniappan**, Image Quality Assessment using Elman Neural Network Model and Interleaving Method, International Journal of Human Computer Interaction (IJHCI), Volume 3, Issue 3, 2012.

[41] **Jiansheng Wu Yu, Jimin Yu**, Rainfall time series forecasting based on Modular RBF Neural Network model coupled with SSA and PLS, Journal of Theoretical and Applied Computer Science, Vol. 6, No. 2, 2012, pp. 3–12.

[42] **Wikipedia**, http://en.wikipedia.org/wiki/Cloud_computing.

[43] **Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, Dav id Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia**, A View of Cloud Computing, Communications of the ACM, Vol. 53 No. 4, Pages 50-58.

[44] **Adobe Creative Cloud**, http://www.adobe.com/products/creativecloud.

[45] **IBM Smart Cloud**, http://www.ibm.com/cloud-computing.

[46] **Yadong Gong, Zongquan Ying, Meihong Lin**, A Survey of Cloud Computing, Proceedings of the 2nd International Conference on Green Communications and Networks 2012 (GCN 2012): Volume 3, 2013.

[47] **Amazon EC2**, http://aws.amazon.com/ec2/.

[48] **Windows Azure**, http://www.windowsazure.com.

[49] **Google Cloud Platform**, https://cloud.google.com/products/cloud-storage.

[50] **OpenStack**, http://www.openstack.org.

[51] **Eucalyptus**, http://www.eucalyptus.com.

[52] **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.

[53] **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.

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

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

[56] **D.P. Gaikwad, Sonali Jagtap, Kunal Thakare, Vaishali Budhawant**, Anomaly Based Intrusion Detection System Using Artificial Neural Network and Fuzzy Clustering, International Journal of Engineering Research & Technology, Vol. 1, Issue 9, Nov. 2012.