INTRUSION DETECTION USING ARTIFICIAL NEURAL NETWORK

Prepared by

GAN SZE KAI
WEK 990008

Under Supervision of
MR. WOO CHAW SENG

Under Moderation of
PN. ROHANA MAHMUD
ASSOC. PROF. DR. SELVANATHAN

INTRUSION DETECTION USING ARTIFICIAL NEURAL NETWORK

By

GAN SZE KAI

WEK 990008

A thesis presented to the Faculty of Computer Science and Information Technology of University of Malaya in partial fulfillment of the Requirement for the Degree of

BACHELOR OF COMPUTER SCIENCE

UNIVERSITY OF MALAYA

JANUARY 2002

ABSTRACT

According to this thesis title – Intrusion Detection Using Artificial Neural Network is whereby a neural network technology must be built inside the Intrusion Detection System (IDS). IDS is one of the security techniques, which is being used currently. It can be said that, Intrusion Detection System is a companion of the firewall, (of is beyond the firewall). The firewall is analog to the locker in (our houses to lock the windows and doors) to prevent intruders' break-in. Therefore, the IDS is a burglar alarm system to alert the user when an intruder successfully get through the firewall. In addition, it is also a vulnerable scanner to scan the network traffic to detect any existing intruder at that period of usage.

In this project, neural network is used to analyze the network traffic, which will be implemented into the IDS to improve its ability to distinguish its authorized user and the intruder. Neural network is itself, a classification system that will enable the IDS to classify between its user and system activities. (This project is only focus on analysis statistical network traffic).

Hence, in my Chapter II - Literature Review, there will be a discussion providing details of the neural network and the IDS. Lastly, in Chapter III, report on the system design of the entire system will be discussed too. System implementation and system testing is discuss in chapter IV and chapter V. A conclusion as well as presentation will then be made preceding the end of the above report.

ACKNOWLEDGEMENT

The development of this intrusion detection using artificial neural network is developed form collecting many information and advice of many individuals. The project will not be success without the help and cooperation from many parties.

First, I would like to thank my supervisor Mr. Woo Chaw Seng. He has given a lot of advised, comments and guideline throughout the completion of the project proposal. Next, I would like to take this opportunity to thank my moderator Pn. Rohana Mahmud and Assoc. Prof. Dr. Selvanathan kindly to become my moderator and giving a lot of suggestions and comments. He's opinion is one of my important guideline to develop this project.

I would like to convey special thanks to my friend and members of networking discussion group from network labs, FSKTM. Especially Lim Shiau Hong, Jimmy Tan Kai Hong. I also want to thank my friend form our neighbor country-Singapore, Low Lee Hong - giving me a lot of help in guiding to writing a good thesis with proper grammars.

Lastly, sincere thanks to all lectures in Faculty of Computer Science and Information Technology, especially Mr. Phang Keat Keong, Mr. Ling Teck Chaw, Mr. Ang Tang Fong and Dr. Sameen Abd. Kareem for sharing their time and knowledge with me.

Gan Sze Kai

17 January 2002

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

CHAPTER I
INTRODUCTION

1.1 Project Overview

This chapter describes the purposes of the project and the problems to be solved. The system functions, limitation as well as the significance and rationale of the project will be discussed too later in this chapter.

The aim of this project is to devise an Intruder Detection System (IDS) using Artificial Neural Network (ANN). The main purpose of this project is to make an intelligent security system. Neural network is use to classify user, system and intruder. Therefore the IDS detection ability will be improved.

1.2 Project Definition

The birth of the Internet triggered the innovation of a whole new generation of telecommunication technology. The invention of the telephone system had defined today's networking world. However, as companies connect their networks to the Internet, they become more vulnerable to virus attack and espionage. Since there is a need to safeguard the users' security and privacy, therefore the protection of the integrity of data communication is very important in avoiding our personal information and confidential data from being stolen.

Firewall-perimeter defense is, in other words, acting like a locker that locks our doors and windows to prevent intrusion by outsiders. It can help the corporate or community to block as well as preventing the outsider from identifying the structure of their network by filtering the unnecessary information. However, firewall had a lot of limitations. It fails to alert authorized users at the time when the intruder is trying to break into their systems. Users or system administrators in companies may then be required to monitor the network system themselves but this will lead to an increased

in security cost. Furthermore, if companies are using manual labour to safeguard their network systems, there may be cases of labour shortages when under some unavoidable cases, such as the staff-in-charged taking a day off and companies cannot find any suitable staff immediately to fill up the temporary post. The loss of time in this case, will provide as an opportunity for any intruder to attack their systems easily.

Thus, IDS (intrusion detection system) is being referred as a burglar alarm and a vulnerability scanner to scan the network system and alert when locates any attacking signature. It can be considered as a replacement of human work to monitor the network system as well as alerting user whenever there is any breaking in into the security system. System administration is only required to check the log or the report from the IDS in certain time. In these ordinary IDS, system administration needs to be updated and change the hacking signature or security rules. But the conventional IDS has some limitations. For instance, it always triggered off a lot of false alarms to the user. It will therefore create a lot of additional work for the particular user to filter out the unnecessary information and changing the rules.

To overcome this problem, an intelligent IDS is needed to help user makes decisions on the security status. An administrator will only need to check the log file in his or her working time and whenever they may find some problems like abnormal login, they can teach the IDS to recognize this type of signal. Therefore this method not only save the time and cost but it also can help the development of the computer intelligent.

1.3 Objective

The main objective of this project is to develop an intrusion detection system using artificial neural network based on the user's requirement. This is achieved by building an IDS concentrated on the implement of neural network technology on the security system. This project will integrate the artificial neural network in the security system and train the neural network to recognize the signal or information flaws in the network system or the personal LAN. In the beginning stage, a basis monitoring system will be built and some basis features will be used in the system itself. For example, logging the activity of the LAN, auditing every machine and detecting the signature (scanning) or style of the user's using the resource in the LAN.

Preceding the building of the monitoring system, a neural network will then be built into this system. This neural network will be trained to employ and analyzes the data provided by the users or from the network system itself. These two learning process can then be employed under supervision or non-supervision conditions.

1.4 Scope

The function of this system is to make the security system more efficient. Nevertheless, this system has its own limitations. The domain covered in this system is on integration of the neural network with the IDS so as to train the system to recognize the distinction between the normal network traffic and the abnormal network traffic, which will then supply the appropriate information to the user.

The achieve of this project:

1. Build a basic IDS (with simulation).
2. Implement the Neural network technology in the IDS.
3. Train the agent to analyze network traffic.
4. Use Pattern matching to distinct normal signal and abnormal signal.
5. Highlight the abnormal signal.

1.5 Project Schedule

Prior to achieving the objectives of this system, a milestone of the whole system is drawn. The milestone will arrange the time for each stages of the system development and after which, a guideline will be prepared in developing the system. A project must be managed properly since it may involve immense effort. Project management is the coordination of all aspects of a project so that it can be completed under the constraints defined. Since this final project has to be completed within a short time span, planning need to be done to:

- Define the goals of the project
- Define and allocate the resources
- Establish timetable and schedules workloads
- Trace and monitor progress of the project
- Report and document the project

Task	*Jun 01*	*July 01*	*Aug 01*	*Sept 01*	*Oct 01*	*Nov 01*	*Dec 01*	*Jan 02*	*Feb 01*	*Mar 01*
Literature Review										
System Analysis										
System Design										
Learn Software										
System Coding										
Integrate System										
System Testing/ Evaluation										
Documentation			3181						3182	

Figure 1.1 Gantt Chart for The Project

1.6 Chapter Overview

Chapter 1:Introduction

This chapter introduces the project in a general view and the rationale of the project. It also states the scope of the system.

Chapter II: Literature Review

This section discusses the coverage and understanding of this project. It includes the detail of the field that our learning has involved. Two major topics will discuss about AI and networking.

Chapter III: System Design And System Analysis

In this chapter, those methods that are being used to gather data and information for this project will be reviewed. The analysis and design of this project will be included in this chapter too.

Chapter IV: Conclusion

This is the conclusion for the chapters above, it include the summary and problem facing in this project.

1.7 Conclusion

This chapter is an overview of the project. The roughly description of the entire project is discuss in this chapter. A brief history of security and current security weakness and the purpose of using artificial intelligent in the IDS bring a very clear concept of the project. The project limitation and project scope set a guideline for the whole development. Finally the chapter overview gives some summary of each chapter.

CHAPTER II
LITERATURE REVIEW

2.1 Introduction

This chapter is about the topics the have been surveyed, learned for gain the information and knowledge to complete this project. This chapter discusses all the related topics for the project, which will aid in narrowing down the area of research. Some unnecessary information will be obliterated to provide a more specific project. The major areas that will be covered are as follow:

Artificial intelligence
Artificial Neural Network
Intrusion detection System
Unified Modeling Language

2.2 Artificial Intelligence

Artificial intelligence (AI) is the study of allowing computers to accomplish tasks in replaced of employing human intelligence. At this moment, human still manages to perform better than computers [1]. Artificial Intelligent is concerned with developing and improving algorithms by employing problem solving techniques and at the same time, perform cognitive, in which humans are currently better at [5][4]. Human is still more superior than computers in learning as well as in gaining unconstrained abilities to perform tasks from any examples and training.

An AI system must be capable to doing three things:

(1) Store knowledge
(2) Apply the knowledge stored to solve problems.
(3) Acquire fresh knowledge through experiments.

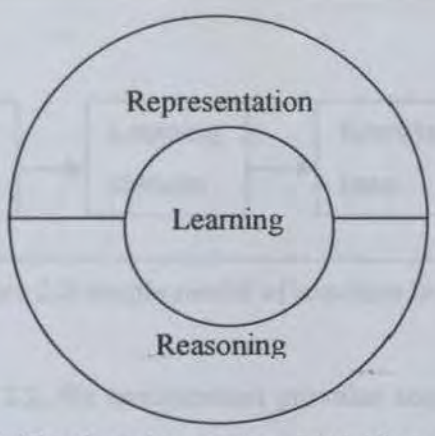


Figure 2.1 The three key components of an AI system.

An AI system has three key components [3].

(1) Representation

AI used symbol structure to represent both general knowledge about a problem domain of interest and specific knowledge about the solution to the problem. Normally, familiar terms are used as in symbols to enable the symbolic representation of AI easier to comprehend.

Knowledge has two categories, declarative and procedural. In a declarative representation, knowledge is represented as a static collection of facts, with a miniature set of general procedures used to manipulate the facts. On the other hand, in a procedural representation, knowledge is embodied in an executable code that acts out the meaning of the knowledge.

(2) Reasoning

Reasoning is the ability to solve problems. A reasoning system must satisfy certain conditions (Fischler and Firschein, 1997) [4].

- Able to express and solve a broad range of problems and problems types.
- Able to make explicit and implicit information known to it.
- Have a control mechanism that determines which operations to apply to a particular problem, when a solution to the problem has been obtained, or when further work on the problem should be terminated.

(3) Learning

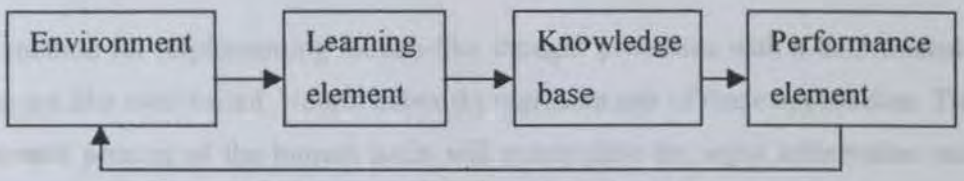


Figure 2.2 simple model of machine learning

As depicted in figure 2.2, the environment provides some information to a learning element. The learning element then employs this information to make improvements to a knowledge base and finally the performance element will apply this knowledge base to perform its task. To formulate the learning element regarding the filling up of missing details or to ignore those that is unimportant, the machine will operates by guessing and a receive feedback from performance element. The feedback mechanism enable machine to evaluate its hypotheses and revise them if necessary.

In addition, there are two types of information processing which are inductive and deductive. In inductive information processing, general patterns and rules are determined from raw data and experience. In deductive information processing, general rules are used to determine specific facts.

2.3 Artificial Neural Network

AI is a method for implementing human-like thought processes with a deterministic machine are like wise varied. Neural networks represent one of these approaches. The function and process of the human brain will manipulate the input information and being stimulated later into the machine [3].

2.3.1 Introduction

A neural network is a massively parallel-distributed processor that has a nature of learning and storing experiential knowledge that had been learn. The fundamental processing element of a neural network is a neuron. A biological neuron is a single cell capable of a sort of crude computation. It is stimulated by one or more inputs, and it generates an output that is being sent to other neurons.

The actual relationship between input and output can be enormously complex. There can be significant time delays between application of the input stimulus and generation of the output response. Fatigue and random events can influence the operation of a neuron. This building block of human awareness encompasses a few general capabilities. Much is still unknown about how the brain trains itself to process information. Presently, researcher is still focusing on the simple model, which accounts mainly the most basic neural processes.

The first artificial neural networks were created in 1943 by McCulloch and Pitts. These were two electrical engineers who wanted to model electronic hardware from the human brain. They came up with what they called a neurode, shown in Figure 2.3. Today the form of the neurode has not changed much, as shown in Figure 2.4.

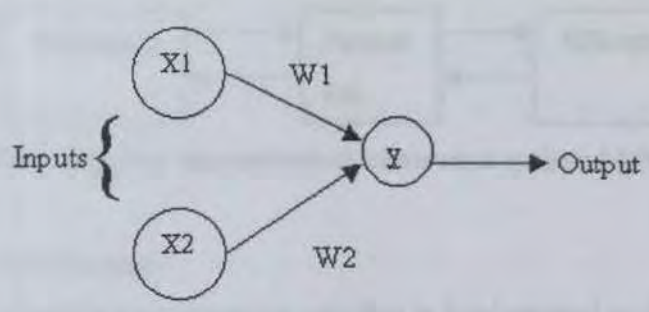


Figure 2.3 The McCulloch-Pitts neurode 1940's

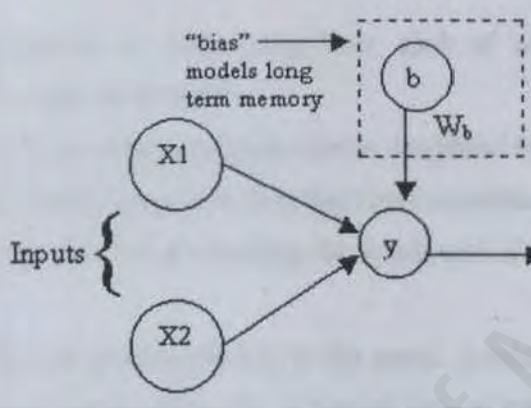


Figure 2.4 A more contemporary neurode mode 1 with bias.

2.3.2 Human Brain

Three stages system central to the system is the brain-represented by the neural network, which continually receives information, perceives it and makes appropriate decisions. Arrows pointing from left to right indicate the forward transmission of information – bearing signals through the system. On the other hand, arrows points from right to left signify the presence of feedback in the system. The receptors convert stimuli from the human body or the external environment into electrical impulses, which in turn, aid in conveying electrical impulses generated by neural net into discernible responses known as system outputs [4].

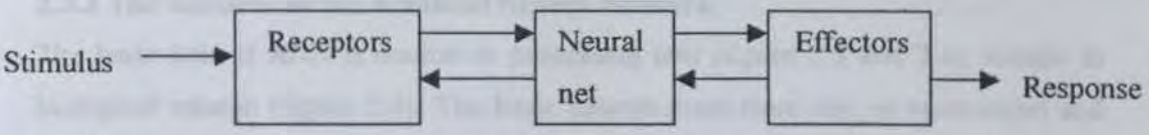


Figure 2.5 Block diagram representation of nervous system (Arbit, 1987)

2.3.2.1 Models of a Neuron

A neuron is an information-processing unit that is fundamental to the operation of a neural network. Here we identify three basic elements of the neuronal model.

1. A set of *synapses* or *connecting links*, each of which is characterized by individual *weight* or *strength*
2. An *adder* for summing the input signals, weighted by the respective synapses of the neuron; the operations described here constitute a *linear combiner*.
3. An *activation function* for limiting the amplitude of the output of a neuron.

In the human brain, the processing unit is the soma. It takes input coming from the dendrites, and outputs to the axon [2]. A typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long (an axon), which splits into thousands of braches. These processes are all handled with chemicals in real neurons.

At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

Figure 2.6 Show a biological neuron in a human brain.

2.3.3 The features of the Artificial Neural Network

The basic unit of ANN is neuron or processing unit (figure 2.3 and 2.4), similar to biological neuron (figure 2.6). The basic neuron must have one or more input and output.

The basic features of Ann are:

- Set of simple processing units in a pattern of connectivity.
- Collection of units that connected in some pattern allows the communication between the units.
- Rule for propagating signals through the network
- Rule for combining input signals.
- A rule for sum all the inputs entering the neuron, process and provide one output.
- Rule for calculating an output signal (activation function)
- A learning rule to adapt the weights
- Every input have their relative weight, the learning rule allows the neuron to change the weight (the ability of neuron modify its behavior in responses to the inputs).

2.3.4 Neural Network Capabilities

Artificial neural network are parallel computational models comprised of densely interconnected adaptive processing units. These networks are fine-grained parallel implementations of nonlinear static or dynamic systems. A very important feature of these networks is their adaptive nature, where "learning by example" replaces "programming" in solving problem [2]. Following are capabilities of a neural network.

1) Classification – Determine crop types from satellite photographs
   - Distinguish a submarine from a boulder given its sonar return

- Identify disease of the heart from electrocardiograms.

2) Noise Reduction – neural network can be trained to recognize a number of patterns. These patterns may be parts of time-series, images, and ET cerate.

3) Prediction – predicting the value of a variable given historic value of it.

Neural network are most likely to be superior to other method under following condition.

1) The data on which conclusions are to be based is "fuzzy". For example, human opinion, ill-defined categories.
2) The patterns imports to the required decision are subtle or deeply hidden.
3) The data exhibits significant unpredictable non-linearity.
4) The data is chaotic (in the mathematic sense).

Many artificial neural networks possess both substantial theoretical foundations and practical utility. Any problem that can be solved with traditional modeling or statistical method can most likely be solved more effectively by employing a neural network.

2.3.5 Benefits of Neural Networks

Neural network has massive parallel-distributed structure and its ability to learn therefore can produce reasonable outputs for inputs that are not encountered during training. It is possible for neural network to solve complex problems. In practice, however, neural network is unable to solve problem individually; they need to be integrated into a consistent system engineering approach [4].

Neural network offers the following useful properties and capabilities:

1) *Nonlinearity*. An artificial neuron can be linear or nonlinear. A neural network, made up of an interconnection on nonlinear neurons, is itself non-linear.

2) *Input-Output Mapping.* A popular paradigm of learning called *learning with a teacher or supervised learning* involves modification of the synaptic weights of a neural network by applying a ser of labeled *training samples* or *task examples.*
3) *Adaptivity.* Neural networks have built-in capability to adapt their synaptic weights to changes in the surrounding environment. In particular, a neural network trained to operate in a specific environment can be easily *retrained* to deal with minor changes in operating environment conditions.
4) *Evidential Responese.* In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to *select*, but also about the *confidence* in the decision made.
5) *Contextual Information.* Knowledge is represented by the very structure and activation state of a neural network.
6) *Fault Tolerance.* A neural network, implemented in hardware form, has the potential to be inherently *fault tolerant*, or capable of robust computation, in the sense that its performance degrades gracefully under adverse, recall of a stored pattern is impaired in quality.
7) VLSI I*mplementability.* The massively parallel nature of a neural network makes it potentially first for the computation of certain tasks.
8) *Uniformity of Analysis and Design.* Basically, neural networks enjoy universality as information processors.
9) *Neuralbiological Analogy.* The design of a neural network is motivated by analogy with the brain, which is a living proof that fault tolerant parallel processing is not only physically possible but also fast and powerful.

2.3.6 Network Architectures

Learning algorithms is used to structure the neurons of a neural network. There are three fundamentally distinct classes of network architecture.

1.Single-Layer Feedforward Network

In the form of layers, neural network are organized in a feedforward layer, which it is known by an input layer of source nodes that projects onto an output

layer of neurons, but is not the other way round. In this single-layer structure there is only one layer of output layer and the source node is not counted. Shown in Figure 2.7.

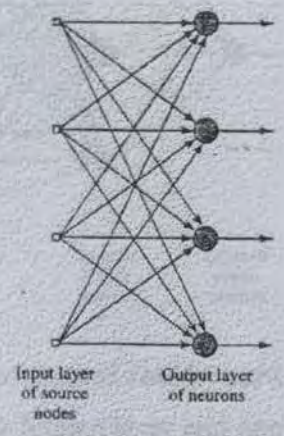


Figure 2.7 Feedforward or acyclic network with a single layer of neurons

2.Mutilayer FeedForward Network

With the presence of one or more hidden layer, the multiplayer feedforward network is formed. Hidden layers are computational nodes, whose is allocated in between the source node and output layer. The extra layers are making network enabled to extract higher-order statistics. This is valuable when the size of the input layer is large. As the single-layer network, the source nodes in the input layer supply the input signals to neurons in the second layer (hidden layer), and the output signal of the second layer become inputs signal for the third layer and so on for the rest of the network but not vice versa. Normally, the naming of this type of network used a series of numbers according to the number of nodes in each layer from the input layer to the final layer. Example in Figure 2.8 is 10-4-2 networks.

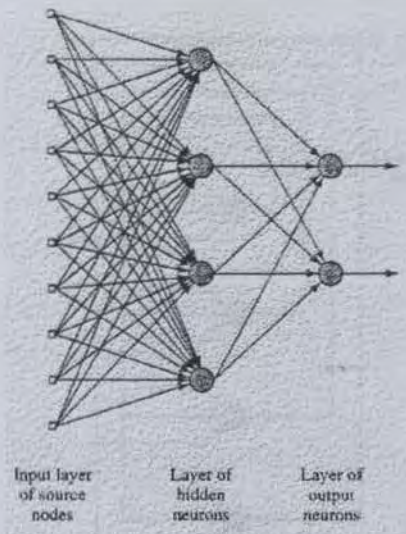


Figure 2.8 Fully connected feedforward or acyclic network with one hidden layer and one output layer.

3.Recurrent Networks

A recurrent network has at least one feedback loop for example a recurrent network may consist of a single layer or multiplayer of neurons with each neuron in the final layer feeding its output signal back to all the other neurons, but there are not self-feedback loop in the network. The presence of the feedback loops has a profound impact on the learning capability of the network and on its performance. The feedback loops involves the use of particular branches composed of unit delay element (z^{-1}), which result in a nonlinear dynamical behavior.

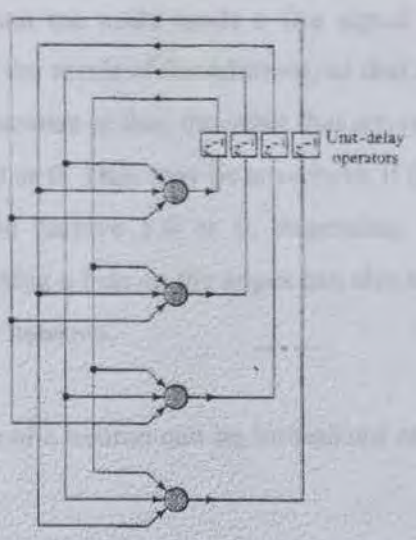


Figure 2.9 Recurrent network with no self-feedback loops and no hidden neurons.

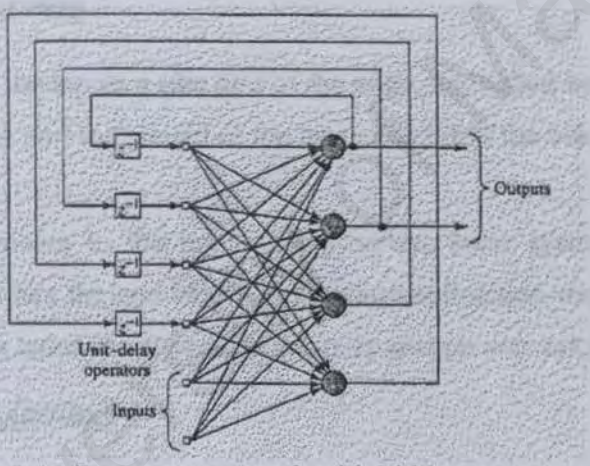


Figure 2.10 Recurrent network with hidden neurons.

2.3.7 How Artificial neural network work

We have a network of nodes connected by edges. When a node is processed, it takes all of the signals on the incoming edges and adds them together. One of these edges is a special bias or memory edge, which is just an edge and it is always on. This value can change to modify the behavior of the network (the higher the bias value, the more likely the neuron is to fire). If the summation of the inputting nodes is above the

threshold (usually 1.00), then the node sends a fire signal to each of its outgoing edges. The fire signal is not the result of the addition, as that may be much more than passed it higher or lower. Because of this, the input that arrives at a neuron can be just about any value, not just 1.0 or 0. They may be anywhere; if the edge bias was 5.0, for example, the neuron would receive 5.0 or 0, depending on whether the neuron attached to it fired or not. Using a bias on the edges can also make a fired neuron have a dampened effect on other neurons.

The equation for the output of a neuron can be formalized as follows:

$$x=b(B)+\sum_{n} bias_n(node_n)$$

$$out=\begin{cases} 1.0 \text{ if } x>1.0 \\ 0.0 \text{ otherwise} \end{cases}$$

Where we sum over the inputs n (the bias of the edge, multiplied by the output of the neuron attached to it) plus the weight of the bias node times the bias edge weight.

Other types of responses to the inputs are possible; some system use a sigmoid exponential function like the one below. A continuous function such as this makes it easier to train certain types of networks (back propagation networks, for example).

$$x=b(B)+\sum_{n} bias_n(node_n)$$

$$out=\frac{1.0}{1.0+e^{-x}}$$

2.3.7.1 Simple Neural Networks

Neural network are Turing-complete; they can be used to perform any calculation that computers can do, if given enough nodes and enough edges. Using NAND gates, you can construct any processor, this doesn't seem like too ridiculous a conjecture. Below is some example of basic logic circuits.

AND

Binary logic seems like a good place to start. As a first stab at a neural net, let's try design a neural net that can perform a binary AND.

AND	0	1
0	0	0
1	0	1

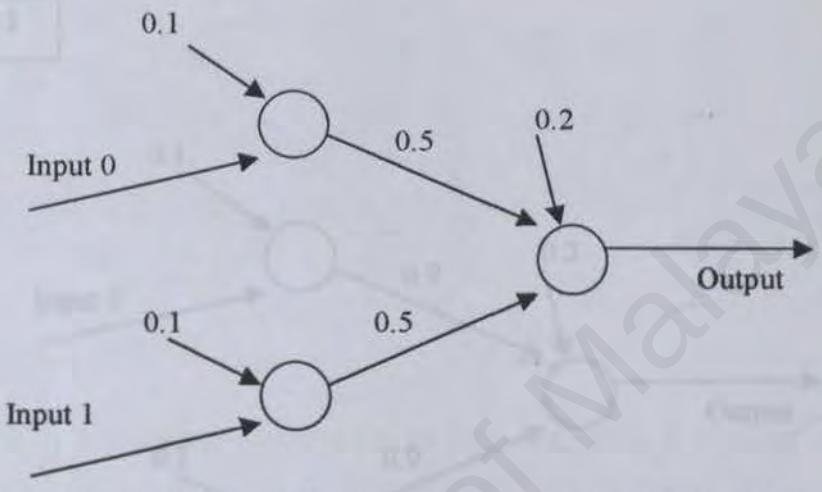


Figure 2.11 A neural network that can perform a binary AND function

Note that the input nodes have a bias of 0.1. This is to help fuzzily the numbers a bit. We could make the network strict if we'd like (setting the bias to 0.0), but for many applications 0.9 is close enough to 1.0 to count as being 1.0.

OR

Binary OR is similar to AND; our middle edges just have a higher weight so that either one of them can activate the output node.

OR	0	1
0	0	1
1	1	1

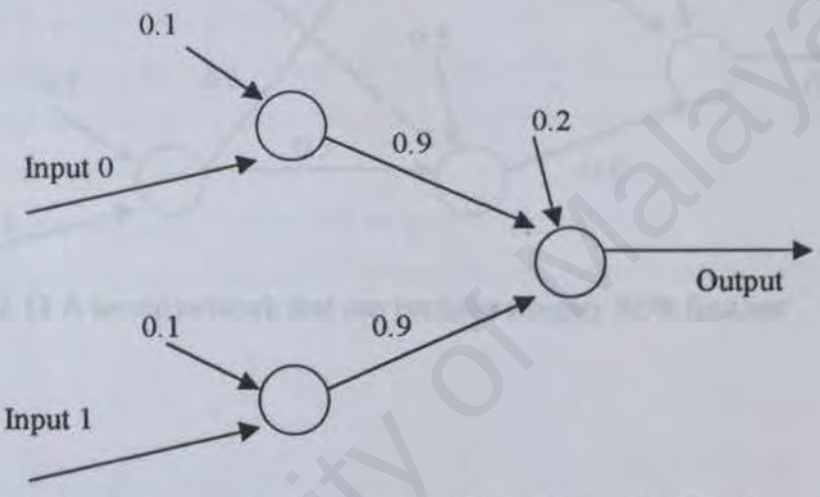


Figure 2.12 A neural network that can perform a binary OR function

XOR

Handling XOR requires a bit more thought. Three nodes alone can't possibly handle XOR; we need to make another layer to the network. A semi-intuitive reasoning behind the workings of figure 2.13: the top internal node will only be activated if both input nodes fire. The bottom one will fire if either of the input nodes fires. If both internal nodes fire, that means that both input nodes fired (a case we should not accept). Which is correctly handled by having a large negative weight for the edge leading form the top internal node to the output node.

XOR	0	1
0	0	1
1	1	0

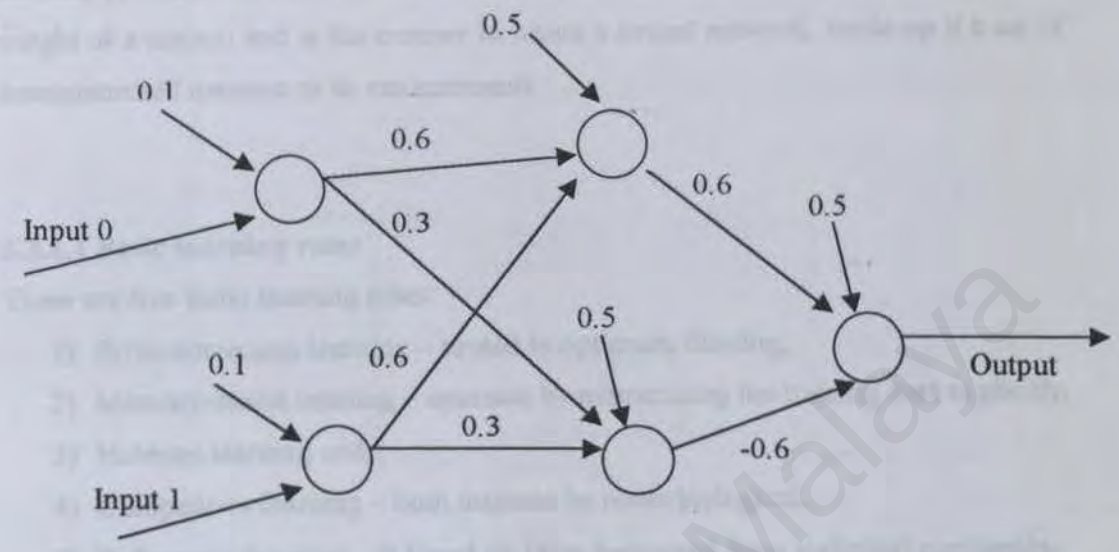


Figure 2.13 A neural network that can perform a binary XOR function

2.3.8 Learning Process

> "Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment on which the network is embedded. The type of learning is determined by the manner in which the parameter changes take places." (Mendel and McClaren, 1970) [3].

This definition of the learning process make neural network implies the following events:

1. An environment stimulates the neural network.
2. The neural network undergoes change in its free parameters as a result this stimulation.

3. The neural network responds in a new way to the environment because of the changes that have occurred in its internal structure.

Learning algorithm – a prescribed set of well-defined rules for the solution of a learning problem. In neural network, learning algorithms used to adjust the synaptic weight of a neuron and is the manner in which a neural network, made up if a set of interconnected neurons to its environments.

2.3.8.1 Basic learning rules

There are five basic learning rules:

1) Error-correction learning – rooted in optimum filtering,
2) Memory-based learning – operates by memorizing the training data explicitly,
3) Hebbian learning and,
4) Competitive learning – both inspired by neurobiological,
5) Boltzmann learning – is based on ideas borrowed from statistical mechanics.

2.3.8.2 Fundamental learning paradigms [3]

The two fundamental learning paradigms are supervised learning and learning without a teacher.

Supervised learning

In conceptual term, teacher has knowledge about the environment and in this manner; knowledge is being represented by a set of input-output examples. Teacher provides the neural network with a desired response for that training vector. Indeed, the desired response represents the optimum action to be performed by the neural network. The network parameters are adjusted under the combined influence of the training vector and the error signal.

Error signal is defined as the different between the desired response and the actual response of the network. In this way knowledge of the environment available to the

teacher transferred to the neural network through training as fully as possible. When this condition is reached, we may then dispense with the teacher and let the neural network deal with the environment completely by itself. The description above and the figure 2.14 is the example of supervised learning with error-correction learning rule.

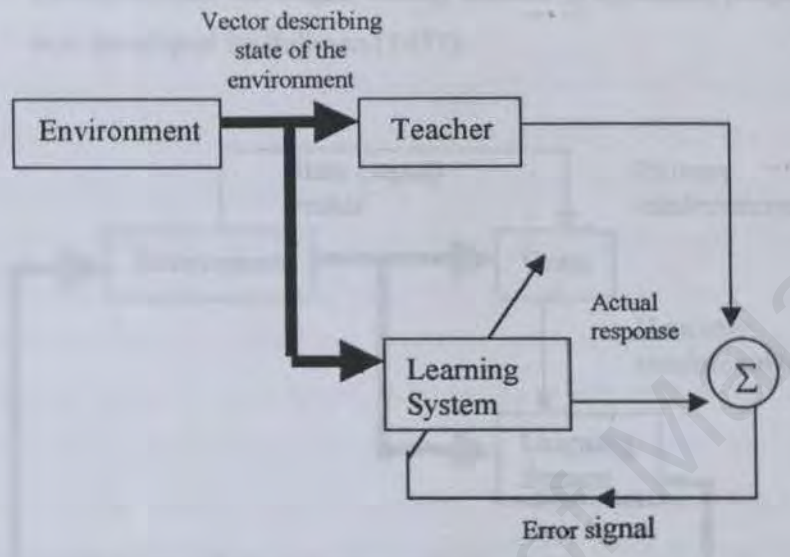


Figure 2.14 Block Diagram of learning with a teacher

Learning without a teacher

As the name implied, there is no teacher to oversee the learning process, no labeled examples of the function to be learned by the network. Two subdivision are identified in this paradigm, Reinforcement learning and unsupervised learning.

1) **Reinforcement learning/ Neurodynamic programming**

   The learning of an input-output mapping is performed through continued interaction with the environment in order to minimize a scalar index of performance. As shown in diagram, a critic is to converts a primary reinforcement signal received from the environment into a higher quality reinforcement signal called the heuristic reinforcement signal, both signal are scalar inputs.

The system is designed to learn under delayed reinforcement, which means that the system observes a temporal sequence of stimuli. Also received from the environment, which eventually result in the generation of the heuristic reinforcement signal.

Reinforcement learning is closely related to dynamics programming, which was developed by Bellman (1957).

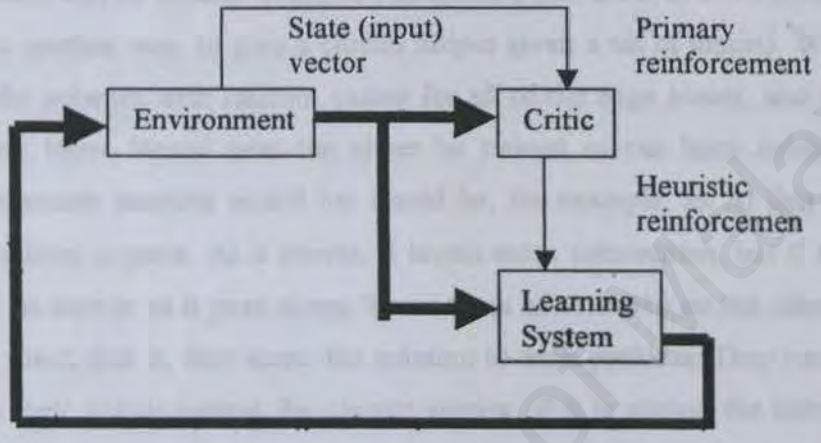


Figure 2.15 Block diagram of reinforcement learning

2) Unsupervised learning

Also call self-organized learning, has no external teacher or critic to oversee the learning process. Network is requires to learn with provision is made for a task-independent measure of the quality of representation/ the free parameter of the network are optimized with respect to that measure. When the network tuned to the statistical regularities of the input data, it develops the ability to form internal representation for encoding feature of the input to create new classes automatically. The competitive learning rule is used to perform unsupervised learning.

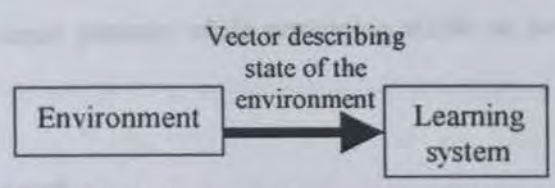


Figure 2.16 Block diagram of unsupervised learning

2.3.9 Training Neural Networks

It's important to know one if the most important and interesting features about neural nets: They can be trained. Suppose you create a neural net to solve a certain problem (or put another way, to give a certain output given a set of inputs). We can initially seed the network with random values for all of the edge biases, and then have the network learn. Neural nets can either be trained or can learn autonomously. An autonomously learning neural net would be, for example, an AI that was trying to escape from a maze. As it moves, it learns more information, but it has no way to check its answer as it goes along. These types of networks on the other hand have a cheat sheet; that is, they know the solution to each problem. They run an input and check their output against the correct answer. If it is wrong, the network modifies some of the weights so that it gets the correct answer the next time.

2.3.10 Examples of Network Model

In this section, few Network model is discussed.

Adeline Network

The Adeline can consider a single-layer backpropagation network. It used to trained on a pattern recognition task, like to classify a bitmap representation of the digits 0-9 into the corresponding classes. But this network has limitation and only trained for reconignizes the exact training pattern. When the application is ported into the multi-layer backpropagation network, a remarkable degree of fault-tolerance can be achieved.

Adaptive Resonance Theory

Adaptive resonance theory network, namely the ability to plastically adapt when presented with new input patterns while remaining stable at previously seen input patterns.

Backpropagation Network

This model has two phases. The first phases called forward phase, is when the input is presented and propagated forward through the network to compute the value for each processing element. In the second phase, called backward phase, use to recurring difference computation in backward direction [6].

Bidirectional Associative Memory

The bidirectional associative memory can be viewed as a generalization of the Hopfield model, to allow for a heteroassociative memory to be implemented.

Boltzman Machine

Boltzman Machine use stochastic learning procedures. Stochastic means involving change, probability, or a random variable. A processing unit can assume new state even if the result is an overall increase in energy.

Counterpropagation Network

The counterpropagation network is a competitive network, designed to function as a self-programming lookup table with the additional ability to interpolate between entries. The application is to determine the angular rotation of a rocket-shaped object, images of which are presented to the network as a bitmap pattern. The performance of the network is a little limited due to the low resolution of the bitmap.

Hopfield Model

The Hopfield model is used as an autoassociative memory to store and recall a set of bitmap images. Images are stored by calculating a corresponding weight matrix. Thereafter, starting from an arbitrary configuration, the memory will settle on exactly that stored image, which is nearest to the starting configuration in terms of Hamming distance. Thus given an incomplete or corrupted version of a stored image, the network is able to recall the corresponding original image.

Perceptron

Perceptron is the earliest of the neural network paradigms; propose by Frank Rosenblatt, a neurophysicist who had previously been psychologist. The application had been done using perceptron is classification of shapes, character recognition and robot vision system.

Self-Organizing Map

The self-organizing map is a competitive network with the ability to form topology-preserving mappings between its input and output spaces.

2.4 Intrusion Detection

Definitions: [Websters75]

- *Intrusion: The act of wrongfully entering upon, seizing, or taking possession of the property of another*
- *Detection: The act of discovering or determining the existence, presence, or fact of*
- *Intrusion Detection: The act of discovering or determining the existence, presence, or fact of the wrongfully entering upon, seizing, or taking possession of the property of another.*

2.4.1 Introduction

Intrusion detection is a security technology that attempts to identify and isolate ``intrusions" against computer systems. Different ID systems have differing classifications of ``intrusion"; a system attempting to detect attacks against web servers might consider only malicious HTTP requests, while a system intended to monitor dynamic routing protocols might only consider RIP spoofing. Regardless, all ID systems share a general definition of ``intrusion" as an unauthorized usage of or misuse of a computer system.

Intrusion detection is an important component of a security system, and it complements other security technologies. By providing information to site administration, ID allows not only for the detection of attacks explicitly addressed by other security components (such as firewalls and service wrappers), but also attempts to provide notification of new attacks unforeseen by other components. Intrusion detection systems also provide forensic information that potentially allows organizations to discover the origins of an attack. In this manner, ID systems attempt to make attackers more accountable for their actions, and, to some extent, act as a deterrent to future attacks.

2.4.2 Intrusion Detection Technology

Intrusion detection technology (IDS) pertains to techniques that attempt to detect intrusion into a computer or network by observation of actions, security logs or audit data.

- Seeks to reactively detect malicious break-ins
- Attempts to identify incongruities either manually or via software expert systems that operate on logs or other information available on the network

2.4.3 Type of IDS

IDS as named vulnerability assessment scanners (VA), also know as "risk assessment products" have three types of forms. There are Host IDS (HIDS), Network IDS (NIDS), and Network Node IDS (NNIDS).

2.4.3.1 Host IDS (HIDS)

HIDS allows the network administrator to define a security policy for the machine on his network but perhaps, a different policy for each operating system or type of server. The Scanner then audits every machine automatically, producing a report that shows the details. These need an agent that resides on each host to be monitored.

The agent scrutinizes event logs, critical system files and other auditable resources looking for unauthorized changes or suspicious patterns of activity. For instance, they will monitor login activities and take note of when an attempt is made to access an account with an incorrect password. In a short time, if there are too many failed attempts to span the system, it may be concluded that someone is trying to gain access illegally and an alarm can then be raised.

They can also monitor the state of system and application files. They do this by making an initial pass of a clean system and storing a condensed "snapshot" of how a certain system should look like. If an intruder or some sort of Trojan Horse does manages to gain access to the system and make changes, the IDS will recognize this and triggered off the alerts.

2.4.3.2 Network IDS

NIDS takes quite a proactive stance. With a sort of "hacker in a box" (database) providing a number of known attacks (web server exploits, Denial of Services), these monitor traffic on the wire in real time, examine packet in detail in order to spot

denial of service attacks or dangerous payload before the packets reach their destination and do the damage.

They match one or more packets against a database of known "attack signatures". The vendors update this database regularly as new attacks are discovered. They will both raise alerts and terminating the offending connecting immediately or in some cases, some will integrate with firewall, which will automatically define new rules to shut down the attacker in future.

Most of the network-based IDS available to date, work in what is known as a "promiscuous mode". It will examine every packet on the local segment. They usually require a dedicated host to run their heavy use of system resources. For instance, most attacks are not based on the contents of a single packer, but also made up of several packers. These are sometimes sent over a lengthy period of time. This means that the IDS need a buffer to store a number of packets in order to compare with database.

2.4.3.3 Network Node IDS (NNIDS)

To overcome the limitation of Network IDS, a hybrid IDS agent appear. This new agent works in a similar manner to the network-based IDS in that it takes packets from the wire and compares them against database entries but they will only concerned with packet targeted at the network node on which it resides.

Host IDS is to do with monitoring of log files and behavioral analysis, where as both Network and Network Node IDS are concerned with TCP analysis. The only different is NIDS is running in promiscuous mode whilst NNIDS is not.

NNIDS system is no longer expected to examine every single packet on the wine, so they can be much faster and take less system resources, and this allow it to be installed on existing servers without imposing too much overhead. For every server you need to protect, you will need a NNIDS agents for each.

2.4.4 Classify

IDS has weakness in classifying the normal users and system activities. The attacker will has an "action" like a normal user's and so, it will be difficult for the conventional to detect the attacker without the predicting ability. In addition, conventional IDS will easily send a false alarm to the user due to this weakness.

Classifier system attempts and learns how to classify future examples from a set of training data [5]. An example of a system that can be used as a classifier is a neural network, which uses a model of biological of biological systems to perform classification. Neural network is characterized by highly connected networks, which are trained on a set of data in the hopes that the network will correctly classify future example.

2.5 Unified Modeling Language (UML)

This modeling language is used to model the software design when software designer are using object-oriented approach. With this modeling language, the software designer and the developer or architect can communicate much better so as to aid in identifying certain system or application. This will also save the time of the designer because he does not need to explain so much to the other members in the development team as the diagram from this modeling language had given them a very clear understanding about the designing of the system or application.

The above paragraph shows that there is a gap between visionary (designers and architect) and developers (programmer). We use UML because it is a tool for building the bridge between gaps. It assists us in capturing the vision for IDS, and then enables us to communicate the vision later to others. It does this via a set of symbols and diagrams.

This modeling language has eight kinds of different diagrams. There are **use case** diagram, **object** diagram, **collaboration** diagram, **sequence** diagram, **state chart** diagram, **activity** diagram, **component** diagram and lastly, **deployment** diagram. Each diagram plays a different role within the development process.

2.6 Conclusion

In this chapter, an understanding about the project and the research or survey result is reported. This project involved three major area, artificial intelligent, networking and security. Section 2.2 discusses the definition of artificial intelligent, and section 2.3 discusses an AI approach- artificial neural network. Section 2.4 is a brief introduction of the IDS.

CHAPTER III

SYSTEM ANALYSIS AND SYSTEM DESIGN

3.1 Introduction

Software development is not as simple as doing the assignment or group project in the class. It needs a high level of discipline as well as in the requirement of design standard, which desires to follow the standard method or model. Software engineering is the application of scientific principles to the orderly transformation of a problem into a working software solution and the subsequent maintenance of that software till the end of its useful life.

This chapter will discuss about how to implement the neural network in the project, for example, choosing of the neural model, number of layers and the training set. Additionally, problems that are being arose and assumption will be explained here.

3.2 Software Development Life Cycle

The first model for system development is waterfall model propose by Royse to characterize the series of software engineering stages. The U.S. Department of Defense prefers to document the software development process as a series of bands along a horizontal belt [9]. Both of these failed to show the symmetry that existed between the earlier and later stage of the development life cycle.

The concept of a V model is used and it is a variation of the waterfall model that demonstrates how the testing activities are related to analysis and design (German Ministry of Defense, 1992). The V model suggests that the program design is verified through the usage of unit and integration testing. These show that all the aspects of the program design have been implemented correctly in the code. System integration should also verify the system design to insure that all system design is properly and correctly implemented. Acceptance testing is conducted by the user rather than the developer, validates the requirements by associating a testing step with each element

of the specification.

The V model linkage on the left side with the right side of the 'V' can problems are found during verification and validation, then the left side of the 'V' can be re-executed to fix and improve requirements, design and coding before the testing steps on the right side are reenacted. It focused more on the activity and accuracy.

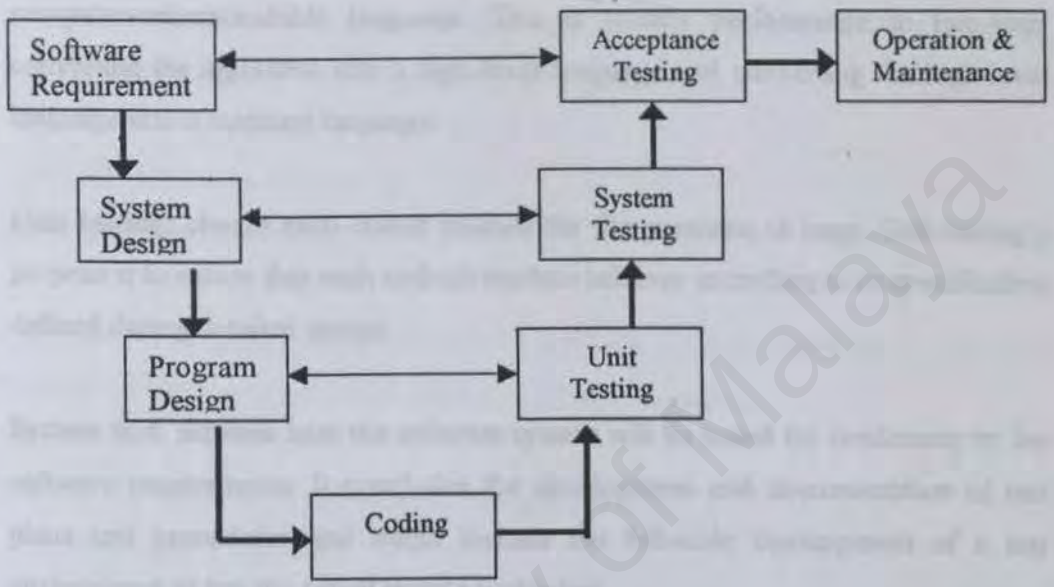


Figure 3.1 The V Model

The V model approach that will be adapted in the proposed project encompasses the activities at system analysis, system design, coding, testing and implementation. Each part of the V-model is discussed below:

Software requirements: includes analyzing the software problem at hand and conclude with a complete specification of the desired external behavior of the software system to be built; also called functional description, functional requirements.

System design: decomposes the software system into its actual constituent (architecture) component and then iteratively decomposes those components located into smaller and smaller subcomponents until the subcomponents located at the

leaves of the resulting design is small enough for people to able to manage it (coding).

Program design: defines and documents algorithms for each module in the design tree that will be realized as code.

Coding: transforms algorithms defined during the detailed design stage into a computer-understandable language. This is usually performance in two-step: converting the algorithm into a high-level language and converting the high-level language into a machine language.

Unit testing: checks each coded module for the presence of bugs. Unit testing's purpose is to ensure that each as-built module behaves according to its specification defined during detailed design.

System test: assesses how the software system will be tested for conformity to the software requirements. It concludes the development and documentation of test plans and procedures and might include the full-scale development of a test environment to test the actual system under test.

Operation & maintenance: is a stage that the completed program used in the real environment and run the operation as expected. Any failure will be fix after the user feedback.

3.3 Development Environment

Is the requirement hardware and software that are used in developing the system.

3.3.1 Hardware Requirement

The minimum hardware requirement

- Pentium II 266 MHz
- 128 MB of RAM (to ensure that JBuilder 4.0 is running smoothly and the operation of the software gain a good performance.)
- 4 GB of Hard Disk
- 17" SVGA color monitor
- Keyboard and mouse.

This minimum requirement is followed the software development tools requirement, JBuilder 4.0 need a minimum of 128 MB of RAM to operate.

This project will develop using this hardware specification

- Pentium III 733 MHz
- 320 MB of RAM
- 15 GB of Hard Disk
- 17" SVGA color monitor (best view with 1024x168)
- Keyboard and mouse.

The high-end machine is choice for development is to avoid any limitation of the hardware resource interrupt the progress of the development. The actual system is expected to require less hardware spec for operation.

3.3.2 Software Requirement

A few software will be used for develop this system. There can be categories in two group, operating system and development tool.

3.3.2.1 Operating System

Red Hat Linux 7.0

Linux is an operating system based on Unix system. It is a very stable for networking and server. It is free to download from the Internet. Linux has provided a lot of networking tools such as, telnet, ssh, scp, ftp, ngrep and others.

3.3.2.2 Development Tool

JBuilder 4.0

JBuider is a powerful Java Integrated Development Environment (IDE). It provides a lot of feature like UI development tools, jdk1.3, testing environment, servlet, tomcat and others. Increase productivity with visual development tools providing maximum flexibility for creating Pure Java applications on the development platform such as Windows, Linux and Solaris.

JDK 1.3

Java is an object oriented programming language. It developed on the year 1995 developed by Sun Microsystems, is a small, simple, safe, object-oriented, robust and powerful programming language. It also an interpreted language, JVM is a virtual machine to interpret the java code (byte code) into machine code, there for it can make the java a cross-platform language, only different JVM need to install in each different operating system. The standard version of JDK 1.3 is free to download form java.sun.com. In this version, in provide a lot of foundation class including awt, swing, applet, and networking.

Matlab

Matlab is a powerful tool in signal processing, image processing, neural network and etc. This software have provide a complete tools and libraries to develop a neural network. Some feature of Matlab will discuss in Chapter IV System Implementation.

3.4 System and User Requirement

System requirement is the needs of the system in a project and user requirement is a spec that system should provide to user. Therefore drawing of a guideline before developing a system is needed. All the description of requirement will be included in this guideline. There are two types of requirement, which are as follow:

- Functional requirement
- Non- functional requirement

3.4.1 Functional requirement

A functional requirement is about how the system should behave and interaction between the system and its environment. The functional requirement for this system are stated below:

User interface

Front-end designs for display the state of the network system and for interaction between user and the system. It is used for user, which sets the rule and training the neural network.

Agent

Agent is a module used to scan the wire and detecting the intruder.

Alarm

Is a warning message when system detects any attacking signal. The alarm is fired depending on how the system compares the information with the database. If the system does not have enough information, some alerts will check with their authorized users whether the particular packet is an intruder.

Database

Is a store to record the entire hacker signature and the learning module by the neural network.

Training Module

Is use to train the agent to detect the unusual signal and report a result.

3.4.2 Non-functional requirement

A non-functional requirement or constraint provides a methodical and meddlesome means to build a good quality of software system. Constraints are used to narrow the selection of language, platform or implementation technique and tool. Below are the non-functional requirements of the system:

i. Maintainability

Maintainability is the degree of the system to be easily maintained and takes note of cost-effectiveness. These ensure that the modification to the function will not decrease or affect the performance of the system. The system is easy to modify and test in updating process to meet the new request, correcting errors, or move to a different computer system.

ii. Reliability

Reliability is which the system performs in the proper manner and process in the way it was designed.

iii. Efficiency

Implementation of the system corresponds to the most cost-effective computing resource utilization, where process that can be called or accessed in an unlimited number of times to produce similar outcomes at a creditable pace or speed. The hardware and software should be used effectively to gain a good performance too.

iv. User friendliness

The system is required to have an easy to use interface. Is should designed for present the necessary information for user. Generally, the design of all the interfaces should confirm to the following criterions:

- Consistent, in terms of screen design and error messages displayed.

- High degree of understandability and avoid memorization of events and commands.

v. Usability

The system should be use without given any difficulty to the user. It shall help the user in using the software.

vi. Performance and scalability

The system should perform consistency even if it is used over thousand times. It should at least, perform a heavy load task when serving a thousand tasks at the same time.

3.5 System Development Method

To develop software, and out line of the purpose of the software must be draw. In this project, the basic architecture and the assumption during the development will be discussed in this section.

3.5.1 Host-IDS

Host base IDS normally work behind the machine. An agent of the Host-IDS must be built into a system like server. Therefore it can protect the server by scanning the logging file from the server, and determine the unfamiliar user-action. A set of data about the intruder signature will store in the database, like Trojan Horse and worm. Image of the system will also be captured frequently to match the machine current status with the status recorded in the image. If the differentiation is too wide, IDS will inform the user.

This kind of scanner is more passive as it works periodically following the schedule set by user. The advantage is that, it uses not much of the system resource. The limitation is, it can only know the intruder during break-in when the scanning is running. Maybe a lot data will be lost or be stealth during the inactive time.

Implement of the neural network in Host-IDS is not to fix the limitation but is to enhance the ability of the IDS in scanning and determining the unusual signature by classifying the normal user activity and the system activity.

3.5.2 Network-IDS

Network-IDS is more pro-active in scanning the information in the wire. It use "promiscuous mode" to copy a set of packet for analysis. It also has a database that stored the hacker record. It can alert the user immediately when the intruder is trying to attack the system, therefore the loss of data will be less.

However, it still has some limitations. Firstly, it consumes a lot of system resources because it needs a buffer for several packets of information for analyzing. Secondly, there is a possibility that it will not work well under the high-speed network like

Figure 3.2 show the basic architecture of the network protected by the IDS.

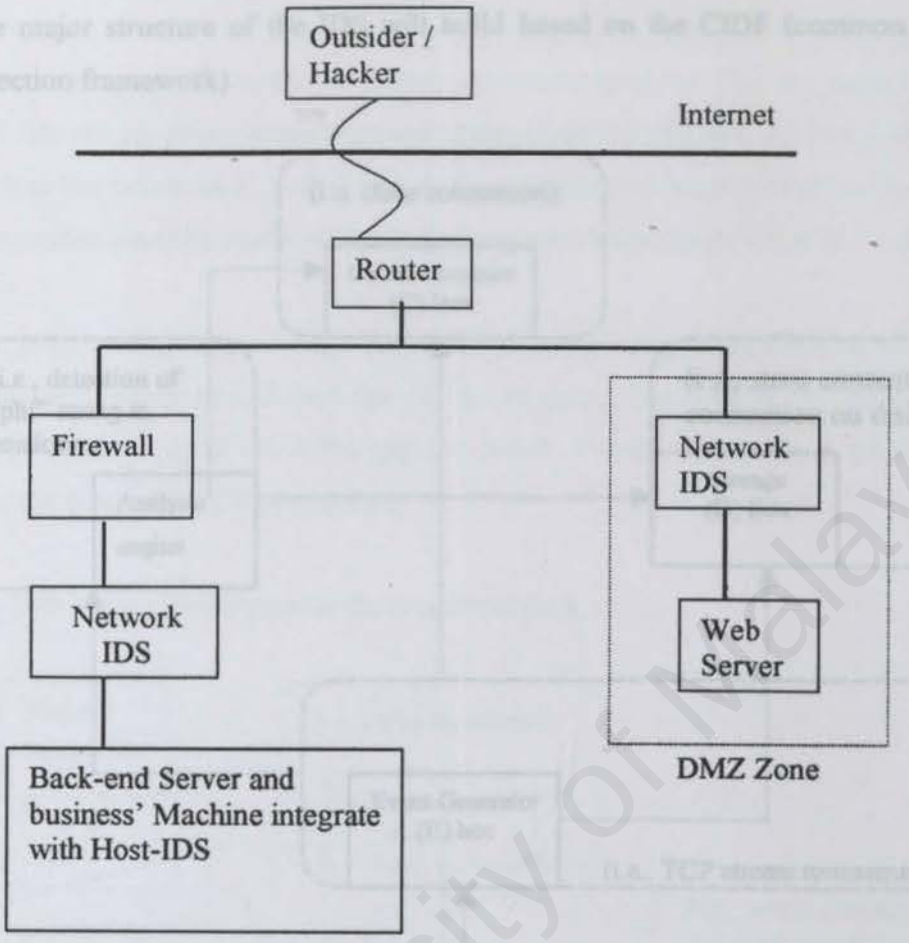


Figure 3.2 A basic architecture of network with IDS.

3.6 System Design

The major structure of the IDS will build based on the CIDF (common intrusion detection framework).

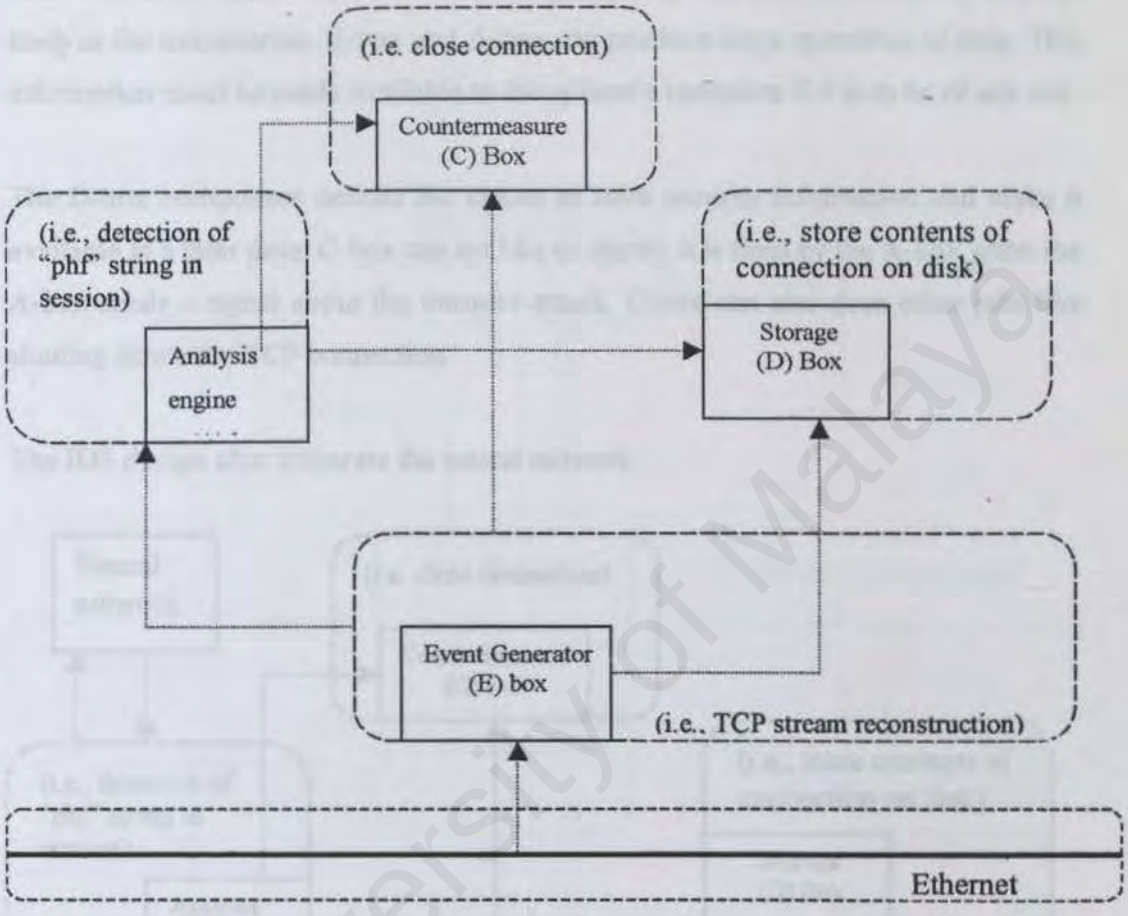


Figure 3.3 the CIDF model of network IDS.

The CIDF consists of four major components. The components include event generators ("E-boxes"), analysis engines ("A-boxes"), storage mechanisms ("D-boxes") and even countermeasures ("C-boxes"). This type of component can be built into software for example, a firewall or a standalone. Figure 3.3 show the mannerism of each component.

The E-box is used to provide the information about events to the rest of the system. It can analog to the sensory organ - human eyes to see the activity on the network.

Without the E-box, the IDS has no information about the outside for making conclusion about security events.

The A-box is to analyze the input from the even generator. This is a logic part of the IDS. However, many researches are being made in this part to find new ways to analyze the information. E-box and A-box can produce large quantities of data. This information must be made available to the system's operators if it is to be of any use.

The D-box component defines the means to store security information and make it available at a later time. C-box can act like an alarm. It is fired by the A-box when the A-box sends a signal about the intruder attack. C-box can also does other jobs like shutting down the TCP connection.

The IDS design after integrate the neural network

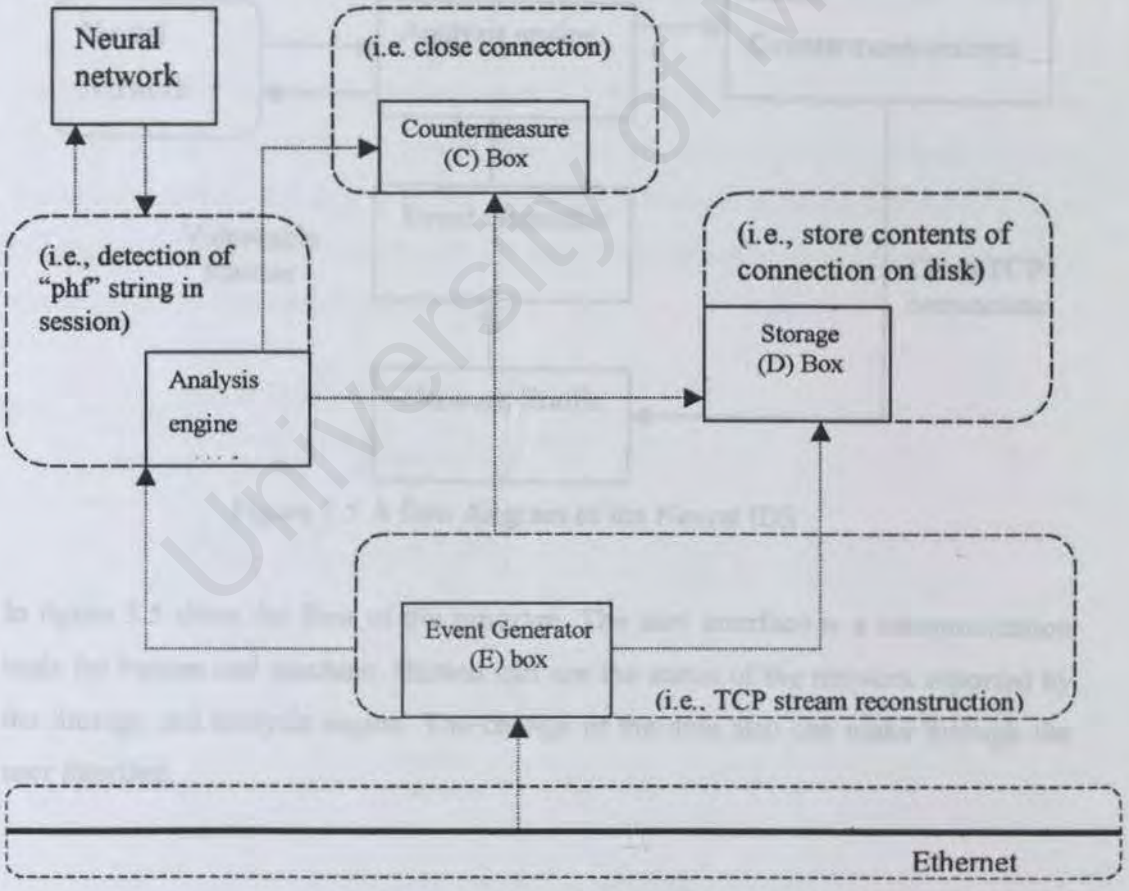


Figure 3.4 The CIDF model with neural network

A basic component of CIDF is not changed; only the A-box is enhanced with the neural network technology. The information from event generator will send to A-box. Inside the A-box have several module, like module to get the content of the packet, this content will send to neural network to do analysis. The analysis result will send back to A-box, another module will get result, is the result is means intruder then the module will fire the C-box. Other module in the A-box is used to record data like time, data, ip, and status of the packet.

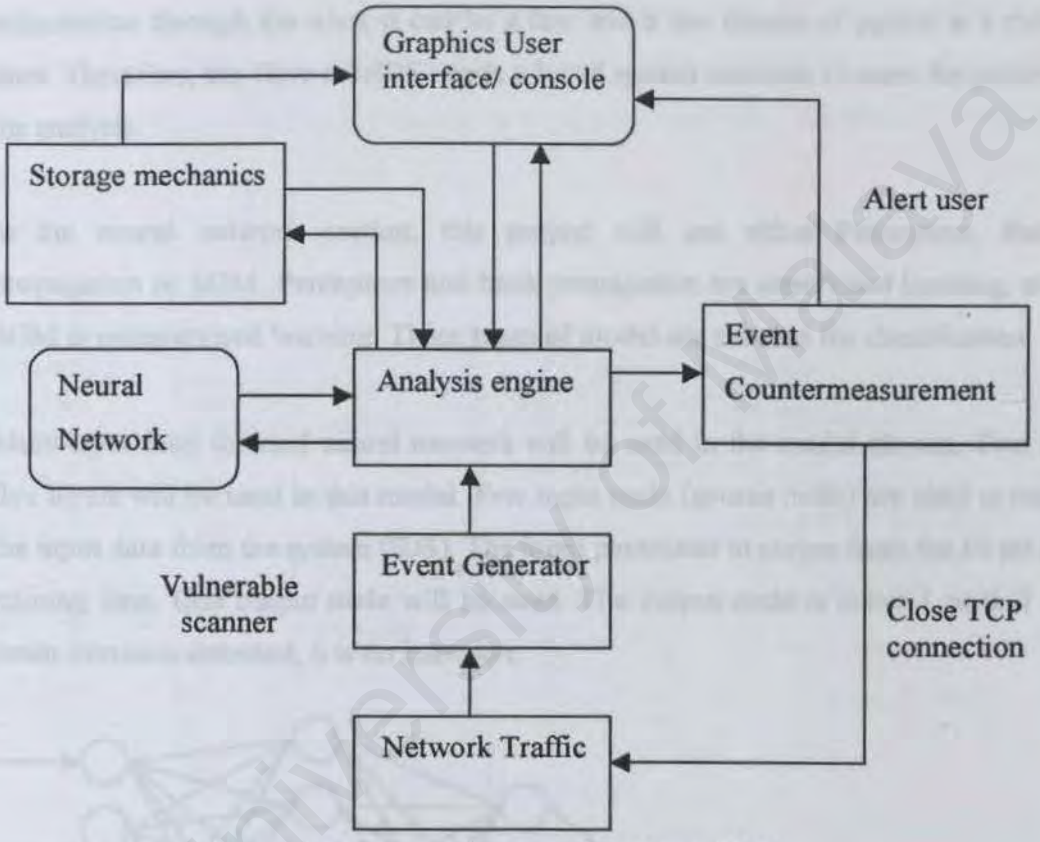


Figure 3.5 A flow diagram of the Neural IDS

In figure 3.5 show the flow of the program. The user interface is a communication tools for human and machine. Human can see the status of the network reported by the Storage and analysis engine. The change of the data also can make through the user interface.

3.6.1 Model Description

In this project the host-base IDS or the network IDS will be choice. Both IDS have its advantage during development. The Host-IDS is need less resource from the system, and it can build together with other software, but it is very depending on the Operating system operating system.

The Network-IDS is more independent; all the information it can get from the network traffic. But because the intruder wills not only a packet of hacking information through the wire, it can be a few and a few cluster of packet in a short time. Therefore, the Network-IDS needs a lot of system resource to store the packets for analysis.

In the neural network section, this project will use either Perceptron, Back propagation or SOM. Perceptron and back propagation are supervised learning, and SOM is unsupervised learning. These types of model are suitable for classification.

Multi-layer feed forward neural network will be used in the model chosen. Two to five layers will be used in this model. Few input node (source node) are used to read the input data from the system (IDS). The input parameter in comes from the 10 set of training data. One output node will be used. The output node is either 1 or 0. 1 is mean intrusion detected, 0 is no intrusion.

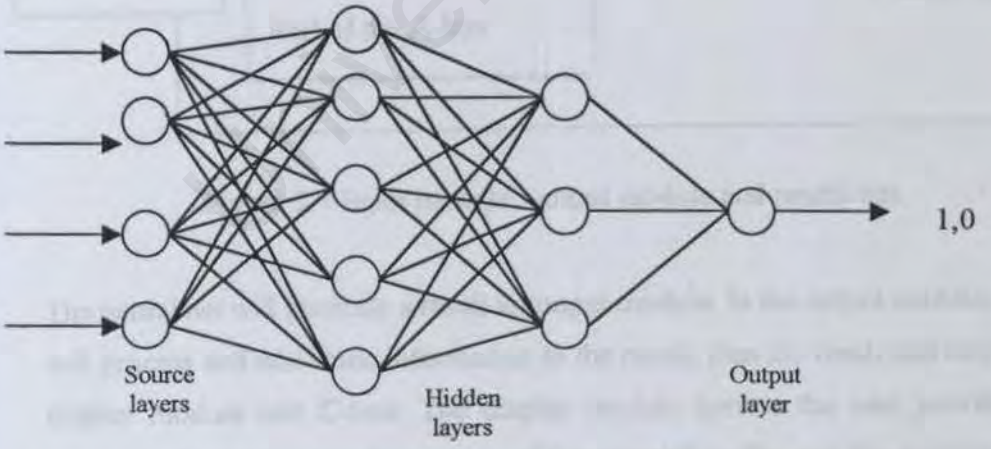


Figure 3.6 A 4-3-1 network

This neural network will be trained with 100 set of data, each set of data consist of network information, either normal signal or abnormal signal. The input parameter will be the statistic of the network traffic and the contents of each packet each time.

The A-box has three module, processing module, input module and output module. The processing module will process the data from the E-box (like arrange the data and filter the unnecessary data). The input module is to receive the input form the processing module and represent the input into the form that the neural network can understand. The neural network will receive the input from the input module and analyzing the information follow the learning algorithm. Within the knowledge the neural gained form the training, neural network can classify the information using this knowledge.

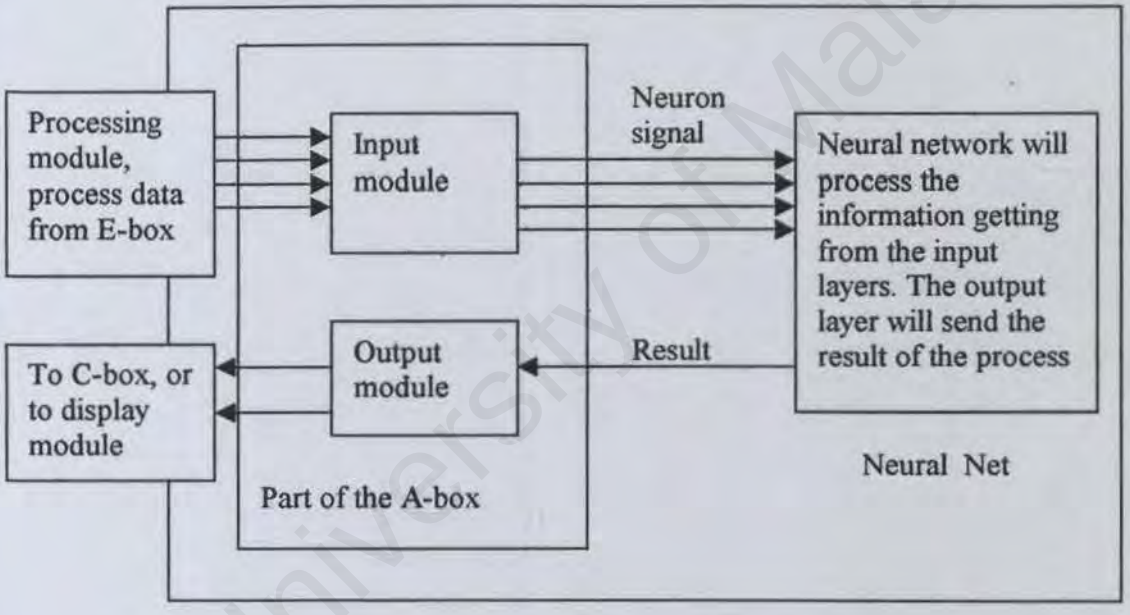


Figure 3.7 Input module, output module and neural net.

The neural net will generate a result to output module. In the output module, the result will process and add some information to the result, then the result will display by the display module and C-box. The display module here is the user interface of the program. User can view the statistic of the network traffic, and the analyze result.

The E-box is not only capture the packet form the wire, it will also record the detail of

the packet in the form (e.g. time, size) and generate the statistic of this packets. All of the information will pass to the A-box for further analyzing.

The C-box will receive the information from the output module in the A-box. Form the result; C-box can decide what kind of action will take, such as alert user. Different type of attack will need a different action to strike back.

The D-box is use to store the record of the pass data, rule and other important data. This D-box can consider the multipurpose database but is only store the data need for the system.

3.7 Conclusion

In this chapter, the methodology that used for this project is explained (section 3.2-3.4), including development model, development requirement, users and software requirement. The software development life cycle is use V-model.

In section 3.5 – 3.7 is about the system analysis and design. The overview of the system analysis in this project is discussed about the type of the IDS and the model of the neural network to be used. The next is the flow diagram of the data structure of the software and the use of modules.

CHAPTER IV
SYSTEM IMPLEMENTATION

4.1 Introduction

Implementation is the process of translating the detailed design into code. The whole project's process of this thesis is doing by an individual, (a developer working on through all the phases of system development).

The requirements analysis, system design and implementations phases do not have a clear boundary in a software development. Each phase tend to overlap each other. Implementing the system is base on the result of the system design. As the terms of implementation is to coding all the modules that designed and compile all the module to make a complete system. Most of the system's architecture is captured during the design phase and are adding in or modified during the coding and evaluation. The implementation needs to fulfill the requirement below [5]:

- Plan the system integration.
- Implement the design classes and subsystems found during system design. In particular, design classes are implemented as file components that contain source code.
- Unit test the components, and then integrate them by compiling and linking them together into one or more executables, before they are sent to integration and system tests.

The following figure depicts the role of implementation in the software life cycle.

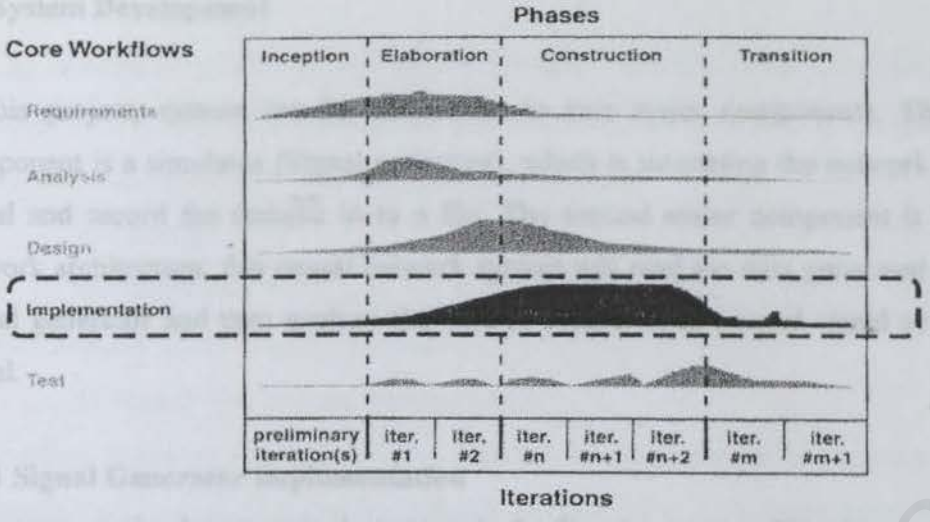


Figure 4.1 The focus of implementation

From figure 4.1, implementation phase is the most time consuming phase in the software life cycle.

4.2 System Development

In this project, system can be divided in to two major components. The first component is a simulator (Signal generator), which is simulating the network traffic signal and record the statistic in to a file. The second major component is neural network architecture; this neural network system will read the data generated by the Signal generator and then analyze the data to classify it as normal signal or attack signal.

4.2.1 Signal Generator implementation

Java programming language is chose to code the Signal generator. The object oriented features in Java like inheritance, polymorphism, abstraction and encapsulation has brought robustness and scalability to this project.

Good programming practice is very importance in implementation. Every specific programming language has its own set of recommendation on good coding style, included Java. A good programmer must awareness the cases below:

1. Use of consistent and meaningful variable names

   A clear and simple naming in variable names will increase the readability of the source code. This will also make the debugging of the code easily by other people. Using of native language is no encouraged, if the source code is used to distributed to other programmer form other country. For example, a Malaysian programmer will use Bahasa Melayu to naming the variable name. It will be meaningful to who is also know Bahasa Melayu, but is not for the programmer who is educated in English.

2. Code Layout for Increased Readability

   It is relatively simple to make a class easy to read. Such as

   - No more than one statement should appear on the same line.
   - Blank lines should separate methods.
   - Break up large block of code with blank lines. This extra space makes the code easier to read and comprehend.

3. Nested if Statements

   Any condition branch should have a properly formatted. For example, when faced with nested code containing the **if-if** construct, one way to simplify it is to make use of the fact that the **if-if** combination

   if <condition 1>

   if <condition 2>

   is equivalent to the single condition

   if <condition 1> **and** <condition 2>

   Another problem with the **if-if** construct is that nesting **if** statements too deeply lead to code that can be difficult to read. So it is recommended to avoid **if** statements nested to a depth not greater than three.

4.2.1.1 Design and development of signal generator

This generator is designed to stimulate a network structure with one server and five networks node or host. Each network node can have only one terminal or is another set of LAN, as shown in figure 4.2. This kind of network structure can be found in University Computer Labs, students will access database server to retrieve data from terminal.

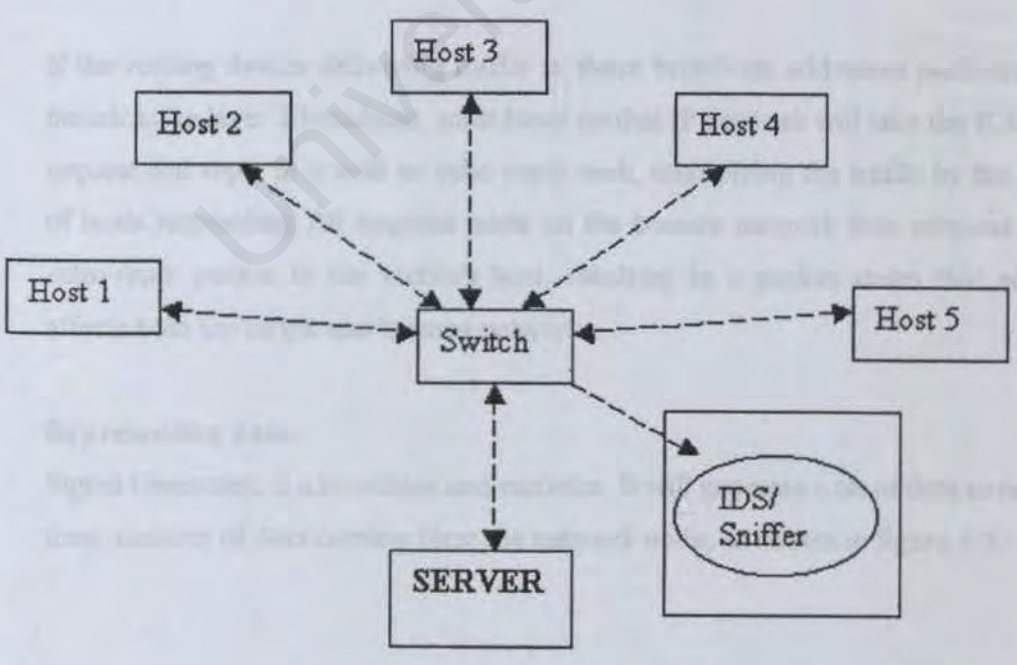


Figure 4.2 An simple network

Follow the example above, a Signal Generator can generate a set of data to represent the number of packets are received by server in certain time from each network node. Therefore this signal generator can generate a set of data to represent hacking pattern and normal pattern. (The real world intruder detection, a system need to collect not only the statistic of network traffic, but also the pattern of each packet need to be considering too).

Denial of Services

There are varies types of hacking pattern, in this project will focus on Denial of Services. Denial of Services as the name presented, is one of the tool used by intruder to congested the network and try to stop the server's service therefore they can do the more further attacks.

SMURF attack is one of the Denial of Services attacks. A smurf attack exploits vulnerability in IP based echo-requests to perform a denial of service attack against a selected host or network. An attacker will forge the source address of an echo-request packet to be that of a selected victim's host. The attacker then sends the packet to the broadcast address of a third party, commonly referred to as the bounce or amplifier network.

If the routing device delivering traffic to those broadcast addresses performs the IP broadcast to layer 2 broadcast, most hosts on that IP network will take the ICMP echo request and reply to it with an echo reply each, multiplying the traffic by the number of hosts responding All targeted hosts on the bounce network then respond with an echo-reply packet to the victim's host, resulting in a packet storm that adversely affects both the target and bounce networks.

Representing data

Signal Generator, is a simulator and recorder. It will generate a set of data to represent time, number of data coming form the network node, as shown in figure 4.3.

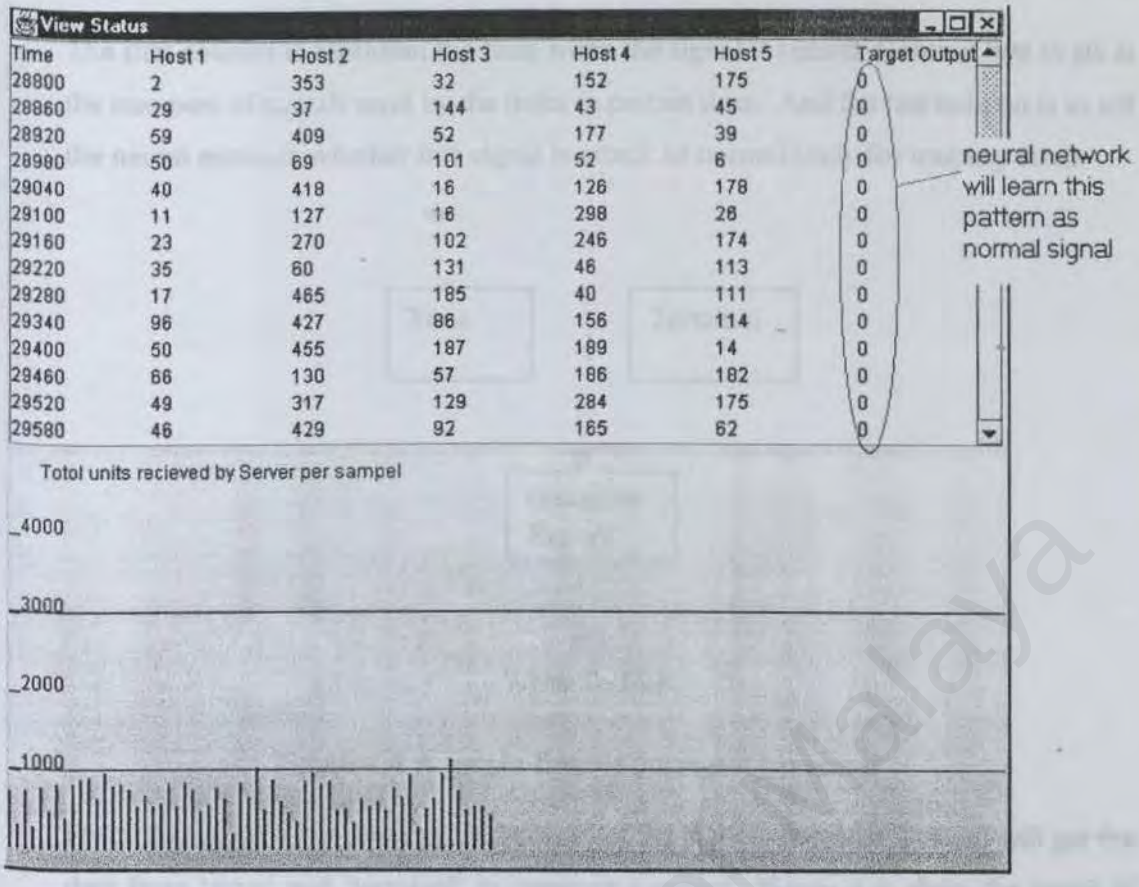


Figure 4.3a A sample of normal signal

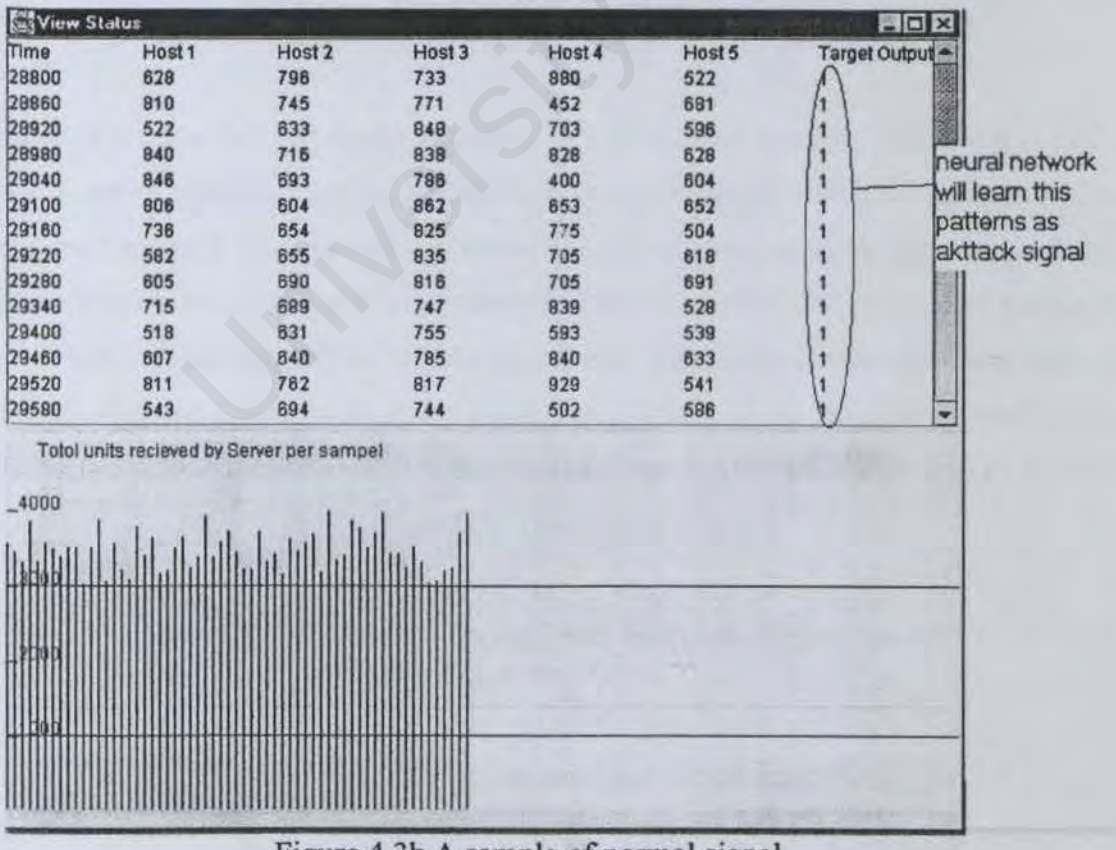


Figure 4.3b A sample of normal signal

The first column is represent the time when the signal is record. Column two to six is the numbers of signals send by the hosts in certain time. And the last column is to tell the neural network whether this signal is attack or normal (only for training data).

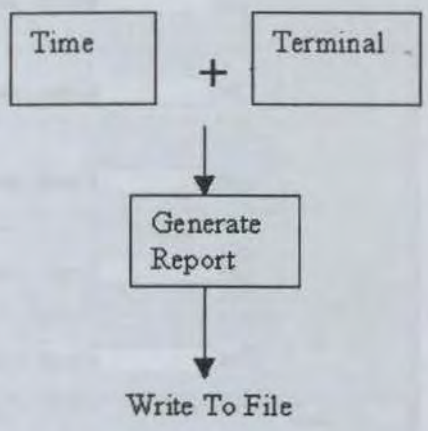


Figure 4.4 A simple flow of the signal generator

From the figure 4.3 is descript the process of the signal generator. System will get the data from 'time' and 'terminal' to generate a report. Figure 4.4; show the detail of this module.

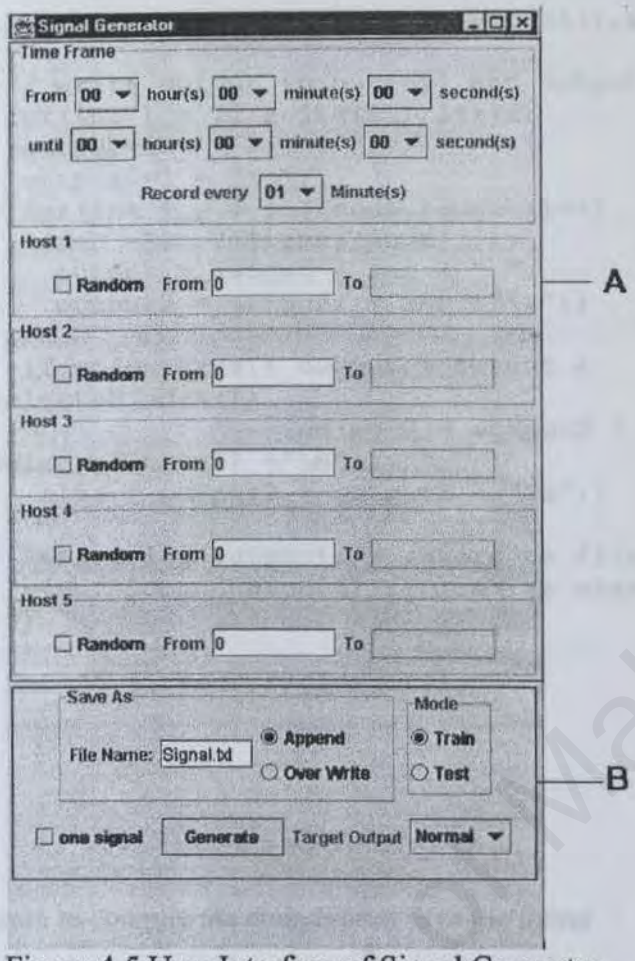


Figure 4.5 User Interface of Signal Generator

Figure 4.5 show that the signal generator can divide into two part, the first part (A) is setting panel which is used to setting the data to be generate such as times and range of signal generate by each network node. The signal generate by the network node can be constant number or randomly generate a set of number within a range setting by user. The second part (B) is a control panel, it let user to set the filename and status of the file (append or over write), set the mode of the is signal (testing or training) and the target output is for neural network to learn the signal (normal or attack), and last user can generate the result by pressing the Generate button.

Below is an example of code how the Generate button to perform the task.

```
Generate.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e)
         { init();
           //open a file or create a file for recording
```

```
            Record.setWriter(FileName.getText(),append);

            // write output to output1 and output2
            for(int i = T1 ; i<=T2 ; i=i+d)
            {output=i + " ";
             output2=i + "\t";
             for(int j = 0 ; j<node.length;j++)
              {rec =RecordSignal(node[j]);
               output = output + rec + " ";
               output2 = output2 + rec +"\t";}

             if(mode==true){ output = output +
Target.getSelectedIndex();
                             output2 = output2 +
Target.getSelectedIndex() +"\n";}
              else { output2 = output2 + "\n";}

             Record.log(output);// record to file
             view.Append(output2);//print to status
             }

            Record.close();//close file
        }
      }
    );
```

The main program to compile the components in to the frame

```
public SignalGenerator() {
    super.setTitle("Signal Generator");

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(Host.length,1));

    //set an object for host
    for(int i =0 ; i<Host.length;i++)
      { Host[i]=new Terminal("Host " + (i+1));
        panel.add(Host[i]);}

   //create a control panel and pass the teminal, time and
status to constructer
   control = new ControlPanel(Host,t);
```

```
    // add all component into JFrame
    Container c =getContentPane();
    c.setLayout(new BorderLayout());
    c.add(t, BorderLayout.NORTH);
    c.add(panel, BorderLayout.CENTER);
    c.add(control,BorderLayout.SOUTH);
    //c.add(status, BorderLayout.EAST);
    pack();
    show();
  }
```

The function of writing the report in to files

```
static void setWriter(String filename,boolean append) {
    if(writer!=null) writer.close();
    try {
      writer=new PrintWriter(new BufferedWriter(new
FileWriter(filename,append)));
    } catch(Exception e) {
      System.out.println(e.toString());
    }
  }

static void close() {
    if(writer!=null) writer.close();
  }

static void log(String msg) {
    if(writer!=null) writer.println(msg);
```

More example on class hierarchy, refer to Appendix A and B.

4.2.2 Build A Neural Network with Matlab

Matlab provide a neural network toolbox. This toolbox contains many types of neural networks including supervised and unsupervised neural network. This project uses Backpropagation neural network model to classify the network signal.

There have many variations of the backpropagation algorithm; the simplest implementation of backpropagatioon learning updates the network weights and biases in the direction in which the performance function decreases most rapidly. There are many types of Backprop network training function - the negative of the gradient. One iteration of this algorithm can be written

$$x_{k+1} = x_k - \alpha_k g_k$$

Where x_k is a vector of current weight and biases, g_k is the current gradient, and α_k is the learning rate.

The command "newff" is use to build a basic backpropagation neural network in Matlab. The syntax for "newff" is

net = newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF), where

PR - Rx2 matrix of min and max values for R input elements.
Si - Size of ith layer, for Nl layers.
TFi - Transfer function of ith layer, default = 'tansig'.
BTF - Backprop network training function, default = 'trainlm'.
BLF - Backprop weight/bias learning function, default = 'learngdm'.
PF - Performance function, default = 'mse'.

and returns an N layer feed-forward backprop network. For example

net = newff([0 86399;0 1000;0 1000;0 1000;0 1000;0 1000],[5 4 1],{'logsig' 'logsig' 'logsig'}, 'trainlm', 'learngd', 'mse');

This command creates a three layers network. There is one input vector with six elements. The values for the first element of the input vector range between 0 and 86399 is use to represent the time in second from 0 second to 86399 seconds (11:59:59pm), the value from the second element until the sixth element with the input range between 0 to 1000 is use to represent the number of signal from each

network node. There are five neurons in first layer, two neurons in hidden layer and one neuron in output layer. The reason of choosing this neural net structure is base on experience from training and testing various type of neural net. From the beginning, a 2-1 neural network (2-1 mean one hidden layer with 2 neuron and one output neuron). And found that it is not stable and cannot perform task well. Therefore one more layer of hidden layer is added. A neural net with 4-1-1, 6-5-1, 5-4-1, 5-3-1 and 5-2-1 was tested, and found that more neuron involved will given better result but will slow down the training process. Finally, an optimum network with 5-2-1 is chosen.

The transfer function used in these three layers is log-sigmoid. The log-sigmoid is using to generate the output value in between 0 and 1. If the output signal is closer the 0 is mean normal signal. If the output signal is closer to 1 is mean attack signal. This output is signal will be further using by the countermeasurement module to take action (this module is not in the scope of the project).

The three-network transfer function can be used in backpropagation are "logsig", "tansig" and "purelin".

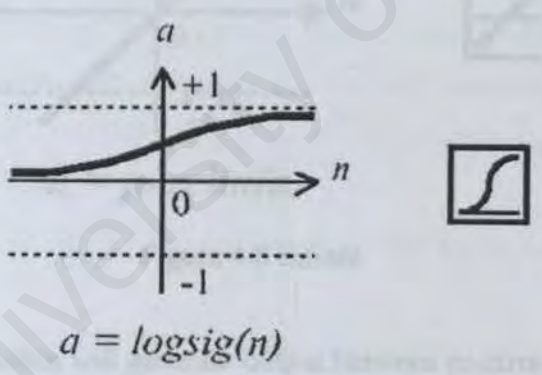


Figure 4.6 Log sigmol

Logsig transfer function will generate outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity.

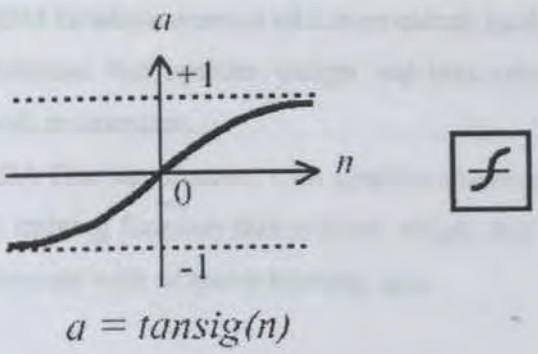


Figure 4.7 Tangent sigmol

Tansig transfer function will generate output between –1 and 1 as the neuron's net input goes from negative to positive infinity.

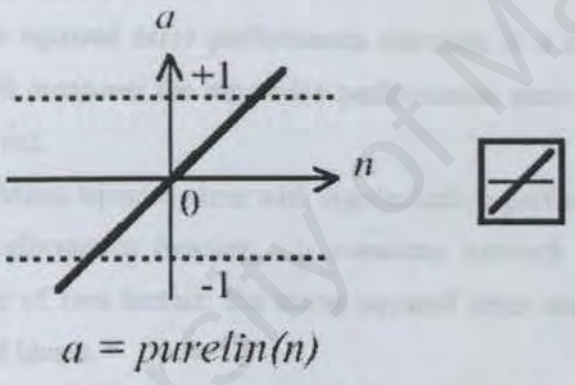


Figure 4.8 Linear

Pureline transfer function will generate output between positive and negative infinity as the neuron's net input goes from negative to positive infinity.

There are many variation of Backprop network training function. Below is some type of training function:

1. TRAINLM Levenberg-Marquardt backpropagation, is the default training function because it is very fast, but it requires a lot of memory to run.
2. TRAINGD Gradient descent backpropagation, is a network training function that updates weight and bias values according to gradient descent.

3. TRAINGDM Gradient descent with momentum backpropagation, is a network training function that updates weight and bias values according to gradient descent with momentum.
4. TRAINGDA Gradient descent with adaptive learning rate backpropagation, is a network training function that updates weight and bias values according to gradient descent with adaptive learning rate.

Two types of Backprop weight/bias learning function

1. LEARNGDM Gradient descent w/momentum weight/bias learning function is the gradient descent with momentum weight/bias learning function.
2. LEARNGD Gradient descent weight/bias learning function is the gradient descent weight/bias learning function.

Four type of performance function

1. MSE Mean squared error performance function, is a network performance function. It measures the network's performance according to the mean of squared errors.
2. MSEREG Mean squared error with regularization performance function, is a network performance function. It measures network performance as the weight sum of two factors: the mean squared error and the mean squared weights and biases.
3. MAE Mean absolute error performance function is a network performance function.
4. DMAE Mean absolute error performance derivative function. Is the derivative function for MAE.

After build the backpropagation neural network, user can configure the network like set the epoch, learning rate, and goal.

4.2.2.1 How to build and train the neural network

net = newff([0 86399;0 1000;0 1000;0 1000;0 1000;0 1000],[5 2 1],{'logsig' 'logsig' 'logsig'}, 'trainlm', 'learngdm', 'mse');

With the command above a simple neural network is build. During the training and test the neural network, the number of layer and neuron and training function is

changed frequently to a stable and well train neural net. Other parameter like learning function and performance function is follow the default setting because default setting get the stable performance to running the task for classification.

During the first time of testing, a small training data is used. A small training data is used because it can be handle easily and using less training times. But this training data is not given enough information to the neural network. During the training with this training data, a small neural net with 2-1 structures is able to get a good result.

After that, a complete training data is used to train the same neural network, and found that, two layer neural network is not stable in performance. Therefore a three layer neural network is used. During the testing, all the training function is used in training, and found that trainlm is given a more accurate result. Graf of training data were shown in figure 4.9.

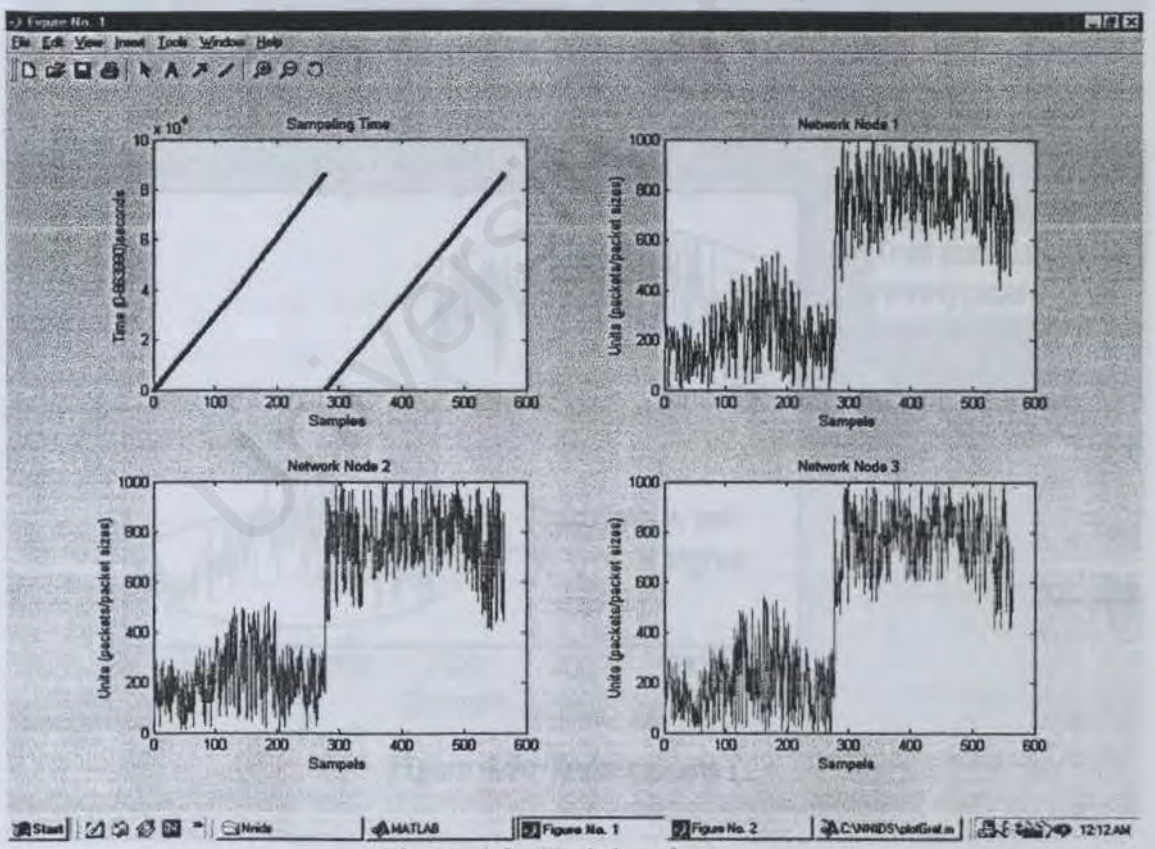


Figure 4.9a Training data

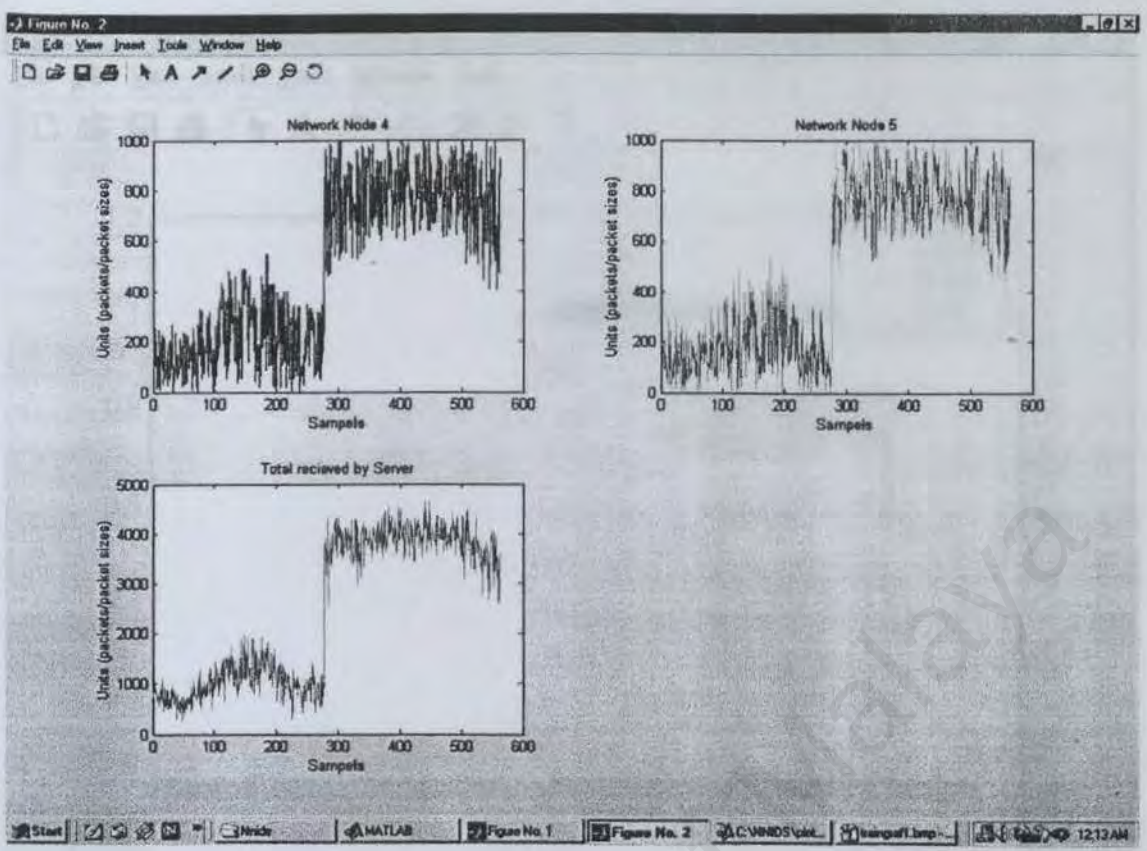


Figure 4.9b Training data

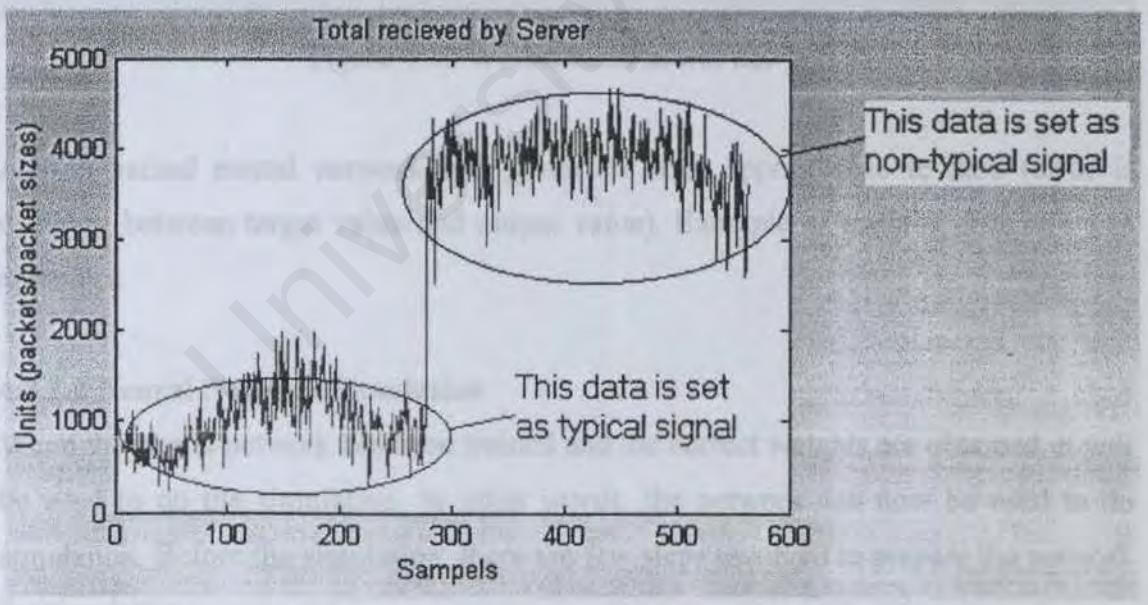


Figure 4.9c Training data

As shown in figure 4.9a and figure 4.9b, the training data is a set of number store in matrix, which is capture as sample per time. Figure 4.9c depict two sets of data, which represent normal and attack signal.

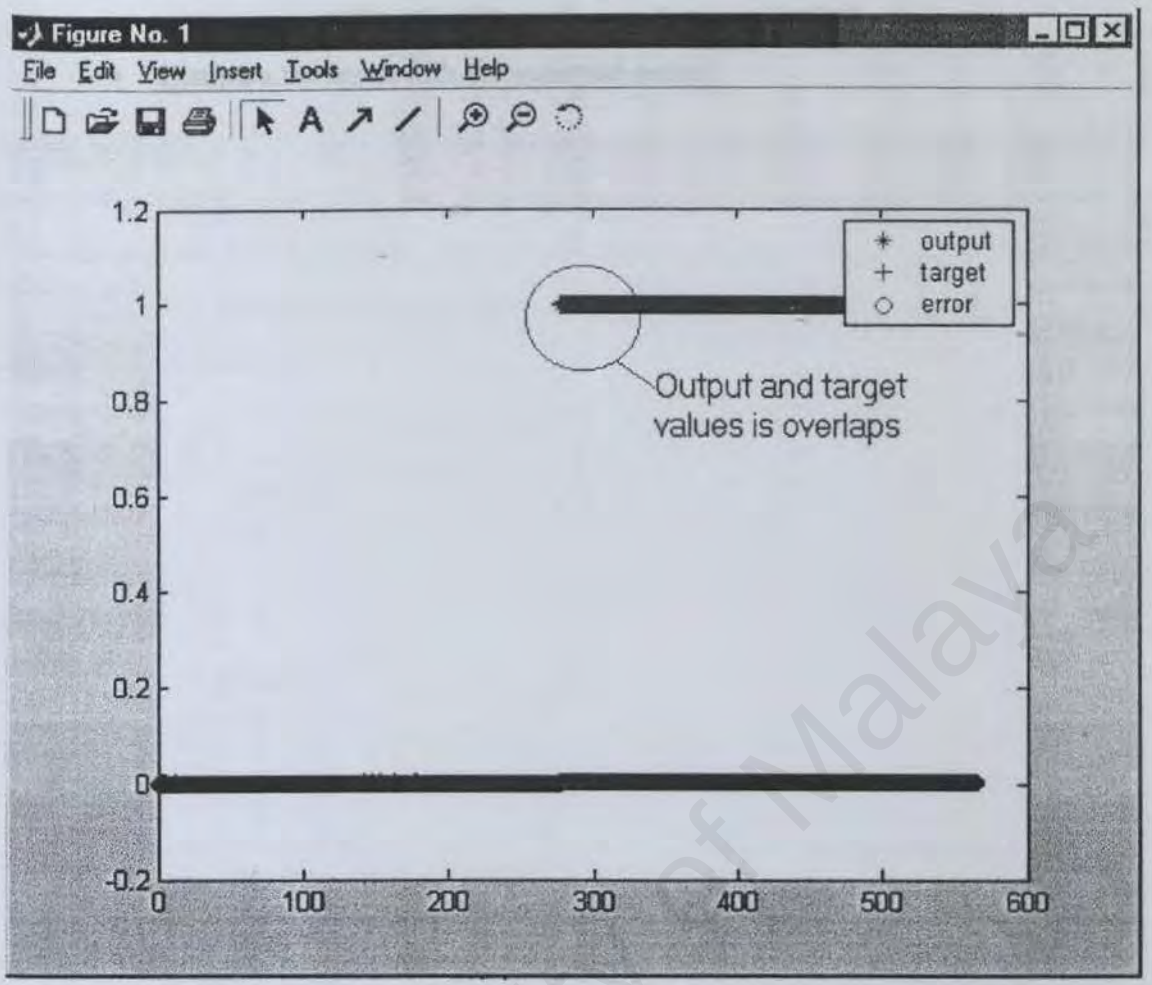


Figure 4.10 Well-trained neural net

A well-trained neural network will given an error approximate to zero (error is different between target value and output value). Example of training data refers to appendix.

4.2.2.2 Neural Network Simulation

When the neural network has been trained and the correct weights are obtained, it will be used to do the simulation. In other words, the network can now be used to do simulation. Before the simulation, there are few steps involved to prepare the network for simulation.

- Load the pre-trained neural network for example "load nnids.mat"
- Load the testing data from file to the array.
- Simulate the testing with this testing data, for example

"[Y1,Pf,Af,E1,perf] = sim(net,test1,[],[]);"

- Compare the result with the expected output.

4.3 Conclusion

This system is implemented to make a neural network to analyze the statistic data from network traffic. Analyze statistic of network signal is one of the technique to detecting intruder. To make the detection to be more accurate, analyze the pattern of the network packets is needed. Analyze the pattern of the network packets is no cover in this project, because analyzing pattern need a very complicated architecture (no in degree thesis syllabus).

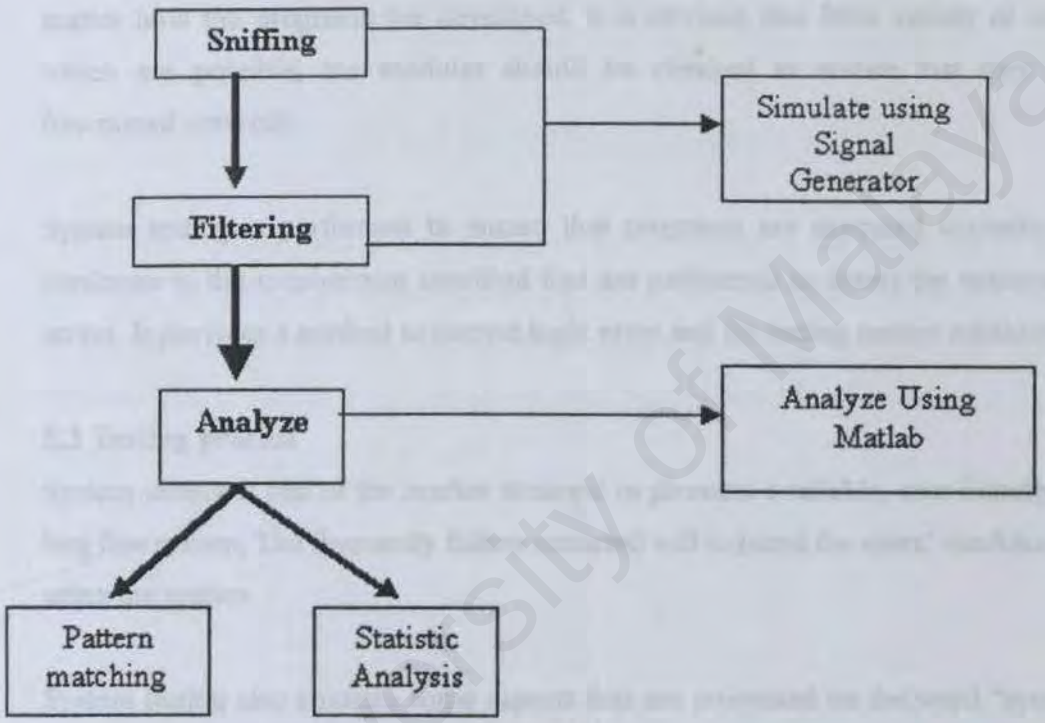


Figure 4.11 A simple IDS system develop in this project

The C-Box and D-Box discuss in Chapter III is a complete model of IDS, which is not develop in this project (not include in the scope of this project). Neural network is act as A-Box to analysis the data.

CHAPTER V
SYSTEM TESTING

5.1 Introduction

Testing is critical for newly developed systems, because a new system that is never distributed out is less or no user given a feedback to the system. Therefore system testing is a very important to establish the popularity in the market or user view. No matter how the programs are developed, it is obvious that from variety of errors, which are possible, the modules should be checked to ensure that they have functioned correctly.

System testing is performed to ensure that programs are executed correctly and conforms to the requirement specified that are performed to detect the existence of errors. It provides a method to correct logic error and for testing system reliability.

5.2 Testing process

System testing is one of the market strategic to promote a reliable, user friendly and bug free system. The frequently failure occurred will reduced the users' confidence in using the system.

System testing also contains some aspects that are orientated on the word "system". This means that those tests should be done in the environment for which the programs was designed, like a multi-user network or whatever. Different form function testing system testing will focus on the whole application and its environment.

The following figure depicts the role of testing in the software life cycle. Testing process is running along the whole project.

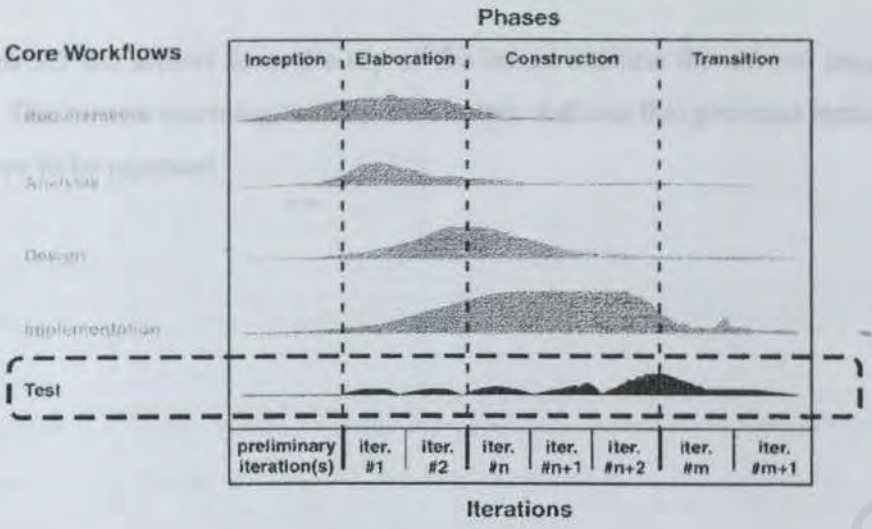


Figure 5.1 – The focus of testing.

Component testing, integration testing and user testing is the sequence of testing activities. As defects are discovered at any stage, program modifications are required to correct them and this may require other stages in the testing process to be repeated. The process is therefore an interactive with information being feedback from later stages to earlier parts of the process.

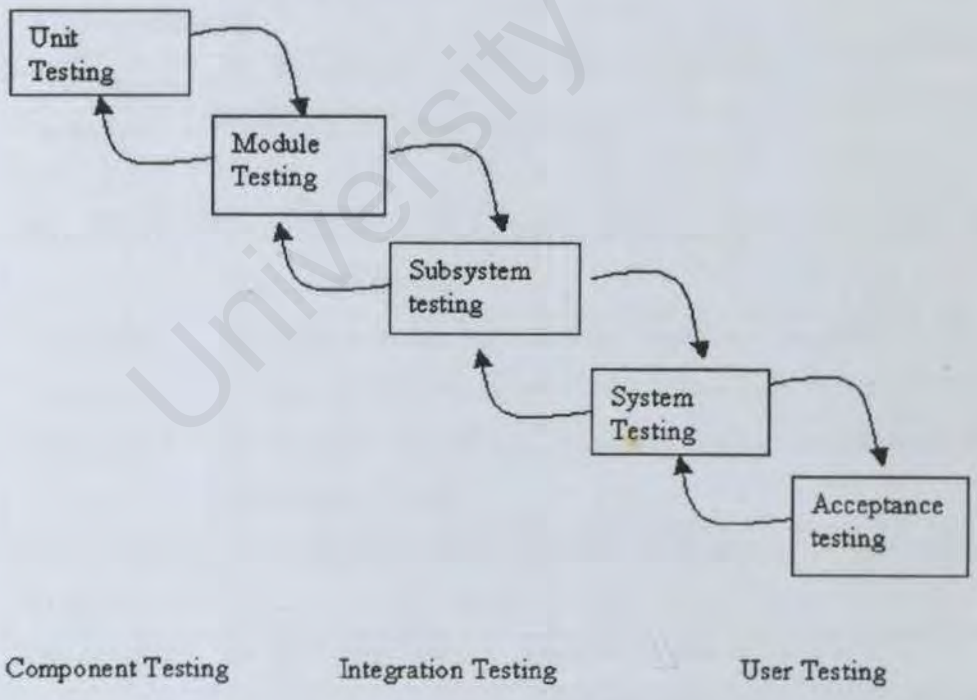


Figure 5.2 The most widely used testing process

In figure 5.2 the arrows form the top of the boxes indicate the normal sequence of testing. The arrows returning to the previous box indicate that previous testing stages may have to be repeated.

5.3 Test Plan

A test plan is developed to detect and identify potential problems before the software is distributed. A test plan also offers road map and testing strategy for testing activities. Below is the example of the test plan.

5.3.1 User Interface

Objective 1	To check the layout of user interface in Signal Generator is arranged properly.
Evaluation Criteria	Every button, combo box and etc, is locate in a proper place and easy to understand and manage by user.
Objective 2	To test that every button can perform well.
Evaluation Criteria	Every button can be activated or run certain tasks.
Objective 3	The Signal Generator can exit and close all related object before quitting.
Evaluation Criteria	Signal Generator is not throwing error signal or cannot close the program.

5.3.2 Functionality

Objective 1	To Test that the time frame can set the time.
Evaluation Criteria	The start, end timer can be set.
Objective 2	In the 'Host' section, the 'random' check box can enable or disable the second text box.
Evaluation Criteria	To make sure that, the 'random' check box is disable by default and the second text box is enable when the check box is selected.
Objective 3	In the 'Save As' section, check the text box is enable and the radio button can be used.
Evaluation Criteria	User can set the filename and to set it append or overwrite.
Objective 4	In the 'Mode' section, the radio button is enabled.
Evaluation	The 'Train' mode is selected and can deselected by selecting 'Test'

Criteria	mode and the 'Target output' combo box is disabled.
Objective 5	To test the 'one signal' mode can be selected.
Evaluation Criteria	By default the 'one signal' mode is disable, when it was selected, in the 'Time Frame' section, only the 'from' time can be set.
Objective 6	Test the "Target output" can change the status
Evaluation Criteria	User can set the signal to be normal or attack.
Objective 7	To test the 'Generate' button.
Evaluation Criteria	When user click the 'Generate' button, system will follow the setting of Time frame, Host, Save As and mode to generate the signal in to a file, and then pop out a frame to report the signals.

5.3.2 Testing the neural network

After the neural network had been trained, it will be tested with a set of testing data with the expected output. The output value form the neural network with the testing data as input will be compare with the expected output. The differential between the output value and expected output will be calculated to satisfy whether the neural network can pass the testing. If the neural network is failed then a new neural network will be train with different parameter, until a well-trained neural network is obtained.

Example of testing script

```
%graf 1
plottest1

[a,b,c,d,e,f]=textread('test1.txt','%d %d %d %d %d %d ');
test1= [a,b,c,d,e,f];
test1=transpose(test1);
[Y1,Pf,Af,E1,perf] = sim(net,test1,[],[]);

aread=[1:1:length(test1)];
target1=zeros([1,length(test1)]);
figure(3),plot(aread,Y1,'*',aread,target1,'+'),legend('output','targe
t');
pause
```

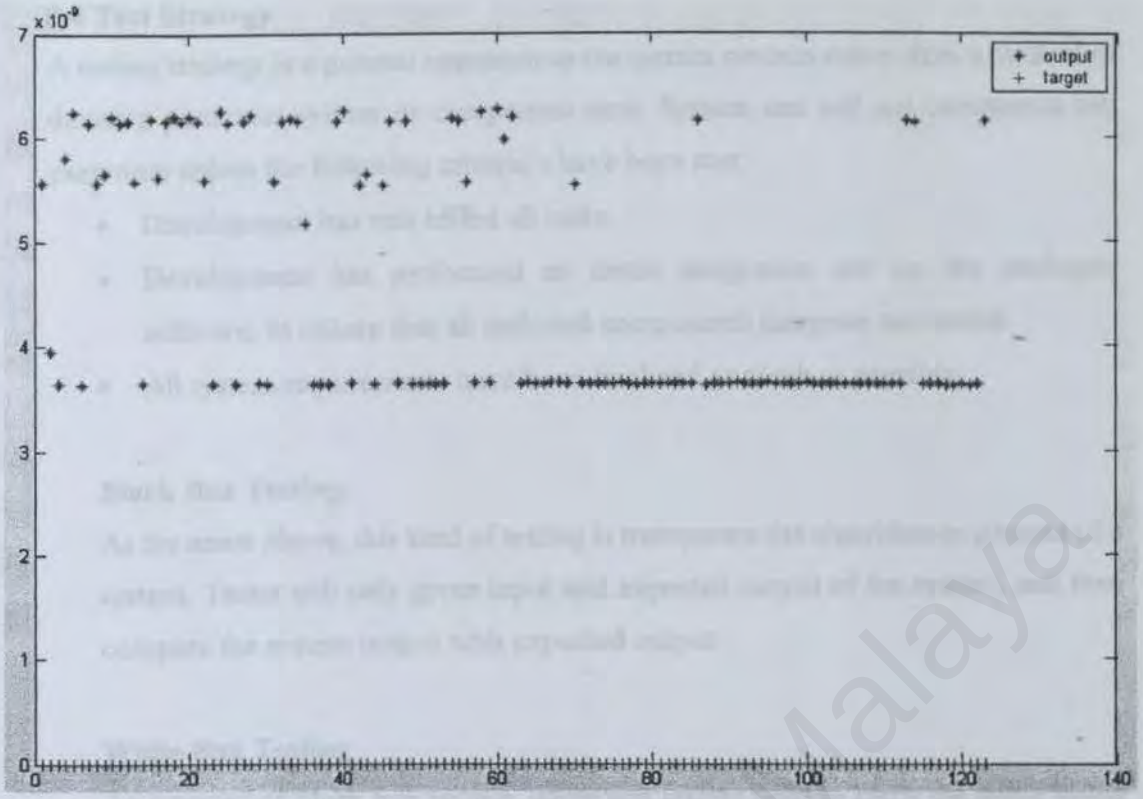


Figure 5.3 Example of the result form neural network (blue) with expected output (green).

5.3.3 Example of the error report form java

```
An unexpected exception has been detected in native code outside the VM.
Unexpected Signal : EXCEPTION_ACCESS_VIOLATION occurred at PC=0x6d4378eb
Function name=(N/A)
Library=(N/A)

NOTE: We are unable to locate the function name symbol for the error
      just occurred. Please refer to release documentation for possible
      reason and solutions.


Current Java thread:
      at
sun.awt.windows.WInputMethod.sendInputMethodEvent(WInputMethod.java:384)
      at sun.awt.windows.WToolkit.eventLoop(Native Method)
      at sun.awt.windows.WToolkit.run(WToolkit.java:188)
      at java.lang.Thread.run(Thread.java:484)

Dynamic libraries:
0x7CC00000 - 0x7CC1D000    C:\WINDOWS\SYSTEM\IMAGEHLP.DLL

Local Time = Mon Jan 14 02:51:55 2002
Elapsed Time = 31
#
# The exception above was detected in native code outside the VM
#
# Java VM: Java HotSpot(TM) Client VM (1.3.1-b24 mixed mode)
#
```

5.4 Test Strategy

A testing strategy is a general approach to the system process rather than a method of devising particular system or component tests. System test will not commence test execution unless the following criteria's have been met:

- Development has unit tested all code.
- Development has performed an initial integration test on the packaged software, to ensure that all included components integrate successful.
- All system requirements have been finalized as much as possible.

Black Box Testing

As the name above, this kind of testing is transparent the algorithm or process of a system. Tester will only given input and expected output of the system, and then compare the system output with expected output.

White Box Testing

An inverted of black box testing, white box testing need to know the detail inside the system. It will guarantee the system is running in the proper way. But it need more time then black box testing and the source code is easy to be exposed.

Thread testing

Is used for benchmarking system with multiple processes where the processing of transaction threads its way through these processes

Top-Down Testing

Top-Down testing assumes that the main logic or object interactions and systems messages of the application need more testing than an individual object's methods or supporting logic. A top-down strategy can detect the serious design flaws early in the implementation. This approach is suitable for testing the user interface and event-driven systems.

Bottom-Up Testing

Bottom-Up testing starts with the details of the system and proceeds to higher level by a progressive aggregation of details until they collectively fit the

requirements for the system. This approach is more appropriate for testing the individual objects in a system. Using this approach, we test each object, then combine them and test their interaction and the messages passed among objects by utilizing the top-down approach.

5.5 Conclusion

The primary result of testing is the test model, which describes how the system is tested. The test model includes:

Test Plan, which specify:

- Test Objectives, which specify the objectives of the system testing and how to achieve them.
- Test Strategy, which specify the approached use to finding defects in this project.
- Test Components, which automate some of the test procedures.

Besides that, testing also results the defects that can be fed back to other workflows, such as system design and implementation.

CHAPTER VI

SYSTEM EVALUATION AND CONCLUSION

6.1 Introduction

After finish the project implementation, a report of evaluation of this project will discuss in this chapter. It will cover the system strengths and limitations. A few suggestions will made as enhance of the system in the future.

6.2 System Strength

1. **Simple Interface**

   Signal Generator is design to have a simple interface. User who is interest in using this signal generator can easily handle and generate some signal (in numeric form) for neural network.

2. **Open source**

   The Signal Generator and matlab m-files source code can be retrieve freely from author for their projects. User is allows to changing and adding new module in this project, with the hope that this project can help user and the new module can be sharing out to public.

3. **Classify signal by analysis statistical data**

   This system can analyze data record by user, and interpret to classify this data as normal signal or attack signal

6.3 System Limitations

1. **Detecting signal**

   This system can only detecting a general behavior of DoS pattern, which is analysis number of packets flow on the line to avoid the DoS congesting the network traffic. To do an advanced detection, a huge database about each pattern of attacking is need. To gain this database consumes plenty of time and need a well knowledge in networking and hacking.

2. **Simulation**

   All the result in this project in done in simulation and it is only a guideline to the real world. It is not promise to be 100% reliable to implement to a real time system.

6.4 Future Enhancement

Because of the limitation in development time and lack of experience in networking skill, most advanced features cannot implement in this project. Therefore, some suggestion for the enhancement is discuss below:

1. **Use other AI methods in advanced the analysis process**
   In the future developments of this project, a recurrent neural network or self-organized neural network will be implement with the hope to advanced the intelligent of the system. Other method like genetic algorithm and fuzzy logic will be implement in this project too.

2. **Fully Java based programming to make it platform independent**
   Java programming is a very popular programming with the feature in object-oriented and cross-platform. To write a java based neural network, developer must very understanding the neural network algorithm and programming skill.

3. **Detecting more hacking pattern.**
   There are more than a thousands type of intruder signals flows in the network. Every signal have its own propose and behavior. To collect all of this data is no an easy work.

4. **Developer guide.**
   A complete developer guide for this project can help other developer to joining the IDS development project.

6.5 Problems Encounter And Solution Taken

During the design and development of this project, there are a lot of problems encountered. Below are some major problem and its solution.

1. **Problem: Lack of knowledge in artificial neural network**

   During choosing this title, the neural network subject is new subject that will only be learned in third year second semester. With the lack of knowledge of artificial neural network, choosing this topic as research paper is very challenging.

   Solution:

   With the help and advice form supervisor and doing a lot of research in neural network can gain knowledge and experience in artificial neural network.

2. **Problem: Lack of knowledge in networking**

   Because of not the student originally form networking department, the difficulty in develop a network based program is cannot be avoided. They're much kind of network architecture and scope like TCP/IP, ATM, LAN, WAN and etc.

   Solution:

   Joining the networking discussion group and read a lot of book about the fundamental networking. Lecturers and members in discussion group are kindly sharing their knowledge and advice.

3. **Problem: Limit of development time.**

   Lack of both knowledge in artificial neural network and networking and less experience in developing system were consume a lot of time to catch up all of this fundamental knowledge. Therefore, the development of this project is quite rush, and many advance feature is not implement in this project.

   Solution:

   Doing some time management and list out all suggested features. Only the

important part will be implement in this project. Using matlab was making the implementation of neural network faster.

4. **Problem: Difficulties in training the neural network**

   Backpropagation neural network is a very powerful feedforward neural network, but it is not easy to be trained because of its behavior and many local minimums occur around the goal.

   Solution:

   To solve the problem, many experiment have been done to find a good parameters for the neural network. The parameters are number of neuron, number of layer, training algorithm and etc. Discuss with lecture also can gain the experience in develop a neural network.

5. **Problem: System failure**

   During coding the signal generator, computer become not stable and hang frequently.

   Solution:

   After checking the system, found that the Java native code have conflict with Windows 98. This problem is not occurs when using Linux or Windows 2000. The report of the error can refer to appendix.

6.6 Knowledge Gained

1. **Knowledge of artificial neural network and networking**

   However, learning neural network and networking is very difficult and challenging, a lot of experience and knowledge was gained form this project.

   The advantages of neural network is it can perform that more advanced that conventional program, for example neural network can do prediction, pattern recognition, perception, motor control and other non-linear tasks.

   A basic knowledge about the networking was gained like how the data flow in packet, how router forward the packet, and basic networking security like how firewall work.

2. **Technique in design and programming skill**

   To develop a project, time management is very important. A development phrase and milestone must define very clear. Some knowledge of programming also can be learned form this project, like Java, an object-oriented concept, design pattern and unit testing; Matlab is a powerful programming language for technical computing. It is used in this system to implement the neural network, and how to read the text file as inputs data for the neural network, and also plotting the result.

6.7 Conclusions

After conducting some analysis and survey on the project previously, it can be concluded that a neural network based intrusion detection system can solve the problem that occur in conventional intrusion detection. For example, reduce the wrong warning signal, using pattern recognition technique, learning form experience, and auto detection.

The system was developed according the scope and limitation that had proposed. With a little modification to fixing the problem that occur during the development. The modification like using matlab as a neural network implement tool to substitute java programming. This modification is not affect propose of this project. This system has two major part, signal generator and neural network. Signal generator is used to generate the data to train and test the neural network. Neural network is used to classify the signal from signal generator.

This project covered three major areas, which are artificial intelligent, networking and security. Therefore this is a challenging project. Many problems occurred during this research. Firstly is the amount of time being consumed. A lot of books need to be referred because the project area is too wide. Secondly is the mastering of other related majors – my major is AI but I also need to master the networking and security. Third is the software designing problem – before master all this area, is very difficult to decide with module is choice for development.

To overcome this problem, an attempt was made to divide the eventual development into four stages. The first stage is in designing very basic IDS and built this model. Second stage is to develop and integrate the neural network in this IDS. Third stage is collecting training pattern to train the neural network. Lastly, the fourth stage is testing the software.

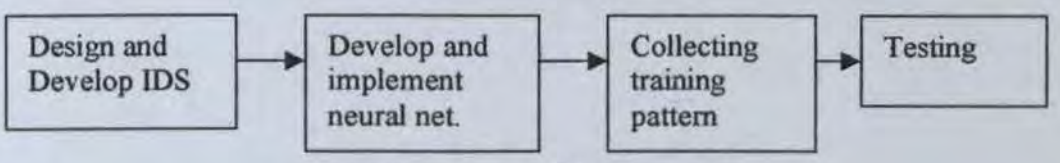


Figure 6.1 Stages of development

During the development of this project, many new things have learned like fundamental of neural network and networking. Development skill was gained form the experience in research this project. The most difficulty part in this project is catching up the fundamental skill and defines the project scope.

This project was implement and can give a brief idea in how the neural network can assist in network security. And also hope that this project can attract many people to advance the project.

APPENDICES

Appendix A:Hierarchy For Package signalgenerator

Class Hierarchy

- class java.lang.Object
  - class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - class java.awt.Container
      - class javax.swing.JComponent (implements java.io.Serializable)
        - class javax.swing.JPanel (implements javax.accessibility.Accessible)
          - class signalgenerator.**ControlPanel**
          - class signalgenerator.**Terminal**
          - class signalgenerator.**Time**
      - class java.awt.Window (implements javax.accessibility.Accessible)
        - class java.awt.Frame (implements java.awt.MenuContainer)
          - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
            - class signalgenerator.**SignalGenerator**
            - class signalgenerator.**Status**
  - class signalgenerator.**SignalLog**

Class ControlPanel

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--signalgenerator.ControlPanel
```

Class SignalGenerator

```
java.lang.Object
  |
  +--java.awt.Component
        |
```

```
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Frame
                          |
                          +--javax.swing.JFrame
                                |
                                +--signalgenerator.SignalGenerator
```

Class SignalLog

```
java.lang.Object
  |
  +--signalgenerator.SignalLog
```

Class Status

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Frame
                          |
                          +--javax.swing.JFrame
                                |
                                +--signalgenerator.Status
```

Class Terminal

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--signalgenerator.Terminal
```

Class Time

```
java.lang.Object
  |
```

```
+--java.awt.Component
      |
      +--java.awt.Container
            |
            +--javax.swing.JComponent
                  |
                  +--javax.swing.JPanel
                        |
                        +--signalgenerator.Time
```

Appendix B: Example of Signal Generator source code

```
package signalgenerator;

/**
 * Title:        Signal Generator for Neural Network
 * Description:  This project is used to generate a 'chuck' of signal
(Smurf attack).
 *               And the generated signal will used to train the
backpropagation neural
 *               network to detect the attack pattern.
 * Copyright:    Copyright (c) 2001
 * Company:      UM
 * @author Gan Sze Kai
 * @version 0.2
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class ControlPanel extends JPanel {

   private      JButton Generate        = new JButton("Generate");
   private    JCheckBox OneSignal       = new JCheckBox("one
signal");
   private   JTextField FileName        = new
JTextField("Signal.txt",6);
   private      JLabel FileLabel        = new JLabel("File Name:");
   private    JComboBox Target;
   private      JLabel TargetLabel      = new JLabel("Target
Output");
   private JRadioButton appendButton    = new JRadioButton("Append");
   private JRadioButton overWriteButton = new JRadioButton("Over
Write");
   private JRadioButton TrainMode       = new JRadioButton("Train");
   private JRadioButton TestMode        = new JRadioButton("Test");

      Terminal [] node;
   private  Time T;
```

```
    Status view=new Status();
   SignalLog Record=new SignalLog();

      String output,output2,statusString;
  private int a1,a2,b1,b2,c1,c2,d,rec;
  private int T1,T2;
  private String [] isAttack={"Normal", "Attack"};


      //status for radio button
      boolean mode = true;
      boolean append=true;


public ControlPanel(Terminal [] host, Time time) {
  node=host;
  T=time;
  Target = new JComboBox(isAttack);
  //make an output to statusString for status object
  statusString = "Time" + "\t";
  for(int i = 0; i<node.length; i++){
     statusString = statusString + node[i].getName() + "\t";}
  statusString = statusString + "Target Output\n";

  //view.set_Size(20,statusString.length());


  // Grouping the component for "Save As" status
  //button group
  appendButton.setSelected(true);
  ButtonGroup group = new ButtonGroup();
  group.add(appendButton);
  group.add(overWriteButton);

  JPanel pane4=new JPanel(new GridLayout(2,1));
  pane4.add(appendButton);
  pane4.add(overWriteButton);
```

```
    RadioListener myListener = new RadioListener();
    appendButton.addItemListener(myListener);
    overWriteButton.addItemListener(myListener);

    JPanel pane5=new JPanel();
    pane5.add(FileLabel);
    pane5.add(FileName);
    pane5.add(pane4);

pane5.setBorder(BorderFactory.createCompoundBorder(BorderFactory.crea
teTitledBorder("Save As"),
            BorderFactory.createEmptyBorder(0,0,0,0)));

    //Grouping for "Mode" section
    TrainMode.setSelected(true);
    ButtonGroup group2 = new ButtonGroup();
    group2.add(TrainMode);
    group2.add(TestMode);

    JPanel pane3 = new JPanel(new GridLayout(2,1));
    pane3.add(TrainMode);
    pane3.add(TestMode);

pane3.setBorder(BorderFactory.createCompoundBorder(BorderFactory.crea
teTitledBorder("Mode"),
            BorderFactory.createEmptyBorder(0,0,0,0)));


    //button listener
    RadioListener2 myListener2 = new RadioListener2();
    TrainMode.addItemListener(myListener2);
    TestMode.addItemListener(myListener2);

    FileLabel.setLabelFor(FileName);
    //panel add "Save As" and "Mode" in one group
    JPanel panel = new JPanel();
    panel.add(pane5);
    panel.add(pane3);

    //pane2 add "OneSignal, Generate, Target in one group
```

```
    JPanel pane2 = new JPanel();
    pane2.add(OneSignal);
    pane2.add(Generate);
    pane2.add(TargetLabel);
    pane2.add(Target);

    //add panel and pane2 to main panel
    setLayout(new GridLayout(2,1));
    add(panel);
    add(pane2);

    Generate.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e)
         { int total=0;
           init();
           view.imagePanel.reset();
           //open a file or create a file for recording
           Record.setWriter(FileName.getText(),append);

           // write output to output1 and output2
           for(int i = T1 ; i<=T2 ; i=i+d)
           {output=i + " ";
            output2=i + "\t";
            total=0;
            for(int j = 0 ; j<node.length;j++)
             {rec =RecordSignal(node[j]);
              output = output + rec + " ";
              output2 = output2 + rec +"\t";
              total=total+rec;}

            if(mode==true){ output = output +
Target.getSelectedIndex();
                            output2 = output2 +
Target.getSelectedIndex() +"\n";}
             else { output2 = output2 + "\n";}

              Record.log(output);// record to file
              view.imagePanel.setNextPoint(total/20);
              view.Append(output2);//print to status
           }
```

```
            Record.close();//close file
        }
    }
    );

    //action to enable and disable the time frame
    OneSignal.addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent e){
         if(OneSignal.isSelected()) T.setClose();
         if(!OneSignal.isSelected()) T.setOpen();}
        }
      );

  }

   void init(){
//T.getTime1(a1,b1,c1);

          a1=T.Time1.getHour();
          b1=T.Time1.getMinute();
          c1=T.Time1.getSecond();
          if(OneSignal.isSelected()){
             a2=a1;
             b2=b1;
             c2=c1;}
           else{
                a2=T.Time2.getHour();
                b2=T.Time2.getMinute();
                c2=T.Time2.getSecond();}

          d=T.getDuration()*60;
          T1=a1*3600+b1*60+c1;
          T2=a2*3600+b2*60+c2;

          view.Name(statusString);
          view.show();
   }
```

```
 int RecordSignal(Terminal T){
   if(T.isCBSelected())
    return T.getStartNo()+(int)(Math.random()*(T.getEndNo()-
T.getStartNo()));
   else return T.getStartNo();}

// Radio action listener for "Save As"
 class RadioListener implements ItemListener{
   public void itemStateChanged(ItemEvent e){
    if(e.getItem()==appendButton) append=true;
    if(e.getItem()==overWriteButton) append=false;
    }
  }

// Radio action listener for "Mode"
  class RadioListener2 implements ItemListener{
   public void itemStateChanged(ItemEvent e){
       if(e.getItem()==TrainMode)
           {mode=true;
            Target.setEnabled(true);}

       if(e.getItem()==TestMode)
          { mode=false;
            Target.setEnabled(false);}
    }
  }
}
```

Appendix C: Example of M-File

```
clear all
close all
net = newff([0 86399;0 1000;0 1000;0 1000;0 1000;0 1000],[5 2 1],{'logsig' 'logsig'
'logsig'}, 'trainlm', 'learngdm', 'mse');
net=init(net);
net.trainParam.epochs=300;

%input data
[a,b,c,d,e,f,Target]=textread('Train.txt','%d %d %d %d %d %d %d');
Data= [a,b,c,d,e,f];
Data=transpose(Data);
Target = transpose(Target);

%train
[net,tr,Y,E,Pf,Af] = train(net,Data,Target);

a=[1:1:length(Target)];
figure(1),plot(a,Y,'*',a,Target,'+',a,E,'o'),legend('output','target','error');
```

Appendix D: Example Training Data

29700 352 293 243 23 202 0
30000 206 377 253 31 81 0
30300 253 318 297 267 336 0
30600 252 381 124 62 374 0
30900 52 218 275 185 264 0
31200 230 203 145 234 189 0
31500 254 65 188 337 381 0
32100 325 331 66 210 180 0
32400 165 111 209 360 49 0
32700 442 220 255 350 72 0
33000 132 310 34 329 214 0
39905 117 470 24 458 281 0
40205 127 101 315 146 165 0
40505 282 58 343 176 408 0
40805 216 218 189 38 63 0
41105 280 222 134 65 313 0
41405 452 71 193 360 277 0
41705 193 368 317 195 251 0
42005 396 222 130 293 144 0
45641 303 334 56 413 136 0
45941 487 456 223 422 314 0
46541 69 400 338 119 225 0
47741 527 45 42 310 117 0
48341 169 378 81 55 140 0
48941 430 482 546 353 99 0
49541 188 98 0 373 130 0
49841 405 333 500 43 85 0
50141 512 72 520 338 237 0
50741 307 409 490 327 82 0
30921 672 730 798 993 782 1
33621 893 883 703 706 826 1
52821 823 892 845 767 803 1
53121 679 899 861 667 827 1
53421 626 824 953 784 661 1
73221 962 711 572 559 567 1
73521 677 522 867 540 599 1
74121 843 978 731 721 749 1
74721 695 800 672 582 853 1
75921 507 716 812 880 615 1
77121 644 488 709 529 406 1

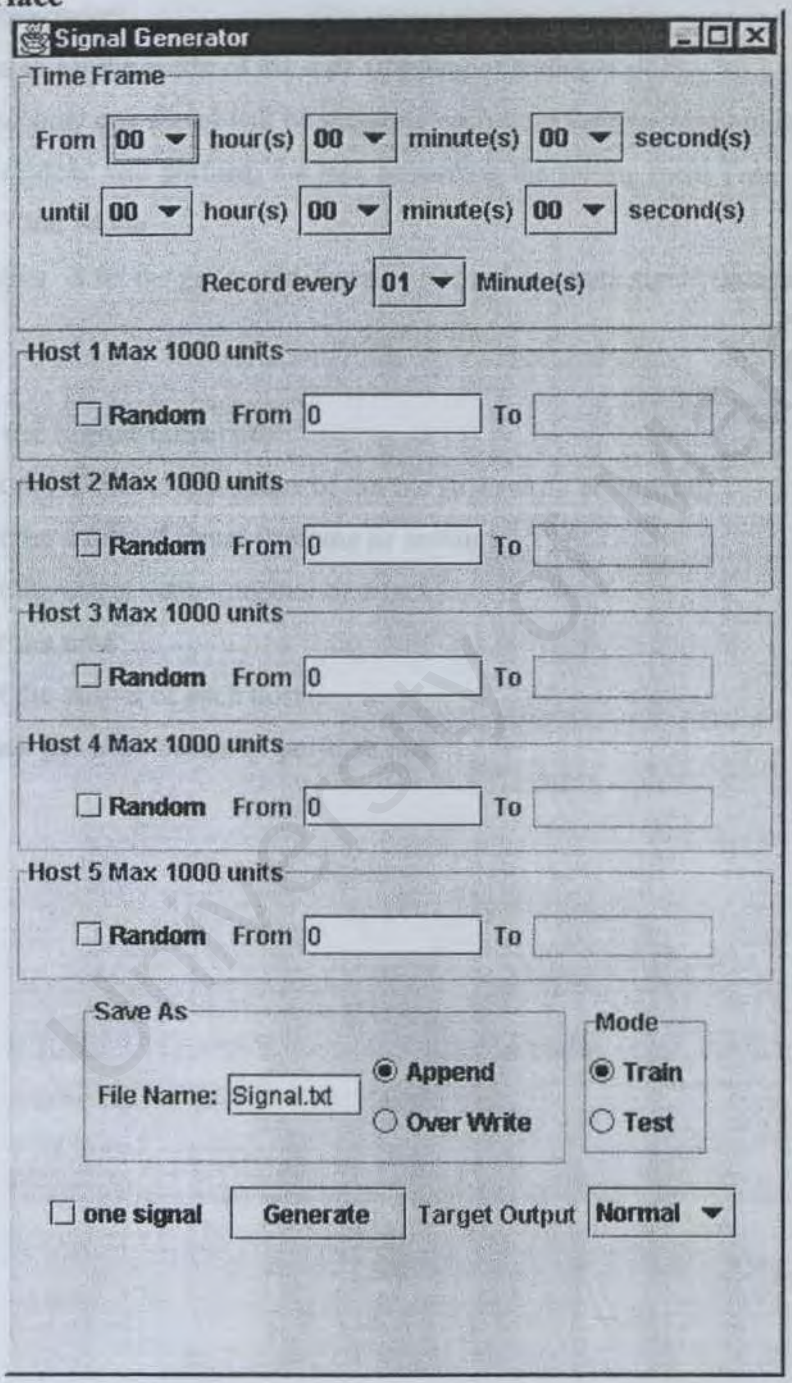
Note: Column 1 is represent time when data is capture. Column 2 until column 6 is representing the number of signal forms the hosts. Column 7 is representing the type of the signal (in testing mode, this column is empty).

Appendix E: User Manual

Intrusion Detection Using Artificial Neural Network

Signal Generator

1. User Interface



Description:

Time frame: Use to set the timer for capturing the data from Host 1 – Host 5.

Host 1-Host 5: Is use to set the number generated when every time the record sequence is fired by time frame. The Host can be set to generate a constant number or a random number within the pre-setting range.

Save As: This section allows user the set the filename and status of the file (append or overwrite).

Mode: Use to set the mode of the data (training or testing mode).

One signal: only one signal will be generate each time the generate button is pressed.

Generate button: will perform the task according the setting form Time frame, Host and Mode

Target output: It set the generated signal as normal or attack signal (in training mode).

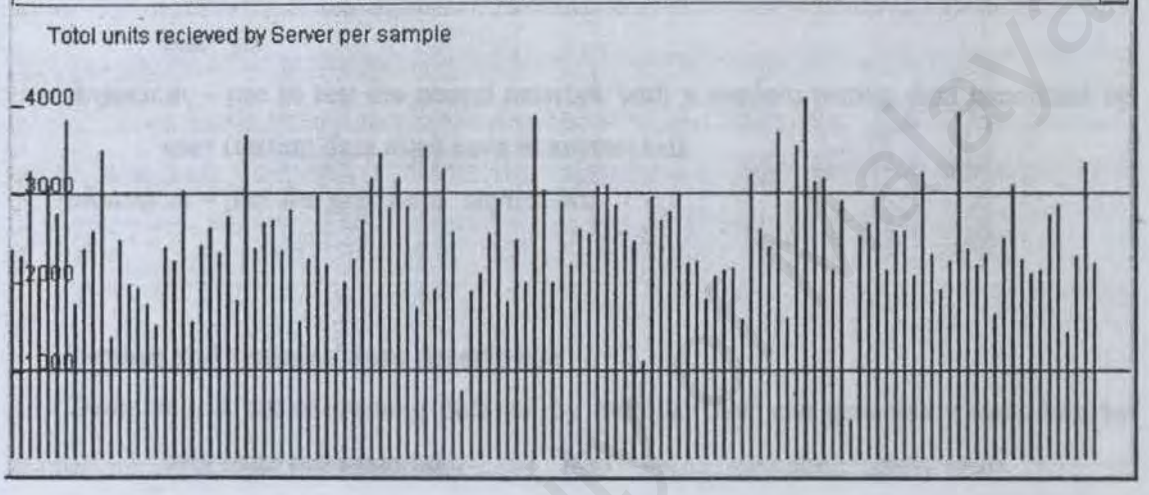
2.How to use Signal Generator

1. Set the filename and status of the file (overwrite or append).
2. Set the mode of signal (training or testing)
3. Set the target value (normal or attack)
4. Set the time
5. Set the output of each hosts.
6. Click generate button to perform task.

Example of the output after the generate button is pressed.

View Status

Time	Host 1	Host 2	Host 3	Host 4	Host 5	Target Output
0	947	944	153	187	111	0
60	73	839	373	269	738	0
120	265	648	971	656	716	0
180	28	649	412	704	283	0
240	923	182	400	883	693	0
300	507	913	691	289	377	0
360	889	948	754	609	630	0
420	290	183	516	145	617	0
480	943	163	351	428	493	0
540	92	935	814	487	248	0
600	220	720	805	987	748	0
660	193	182	469	203	321	0
720	133	434	719	238	945	0
780	368	573	123	610	298	0



Neural Network

Matlab : M-files description

trainIDS.m – use to train the neural network with the training data from Train.txt

plotTrainGraf.m – Plot the graf for training data from Train.txt.

plotTrainGraf2.m– Plot the graf for training data from Train.txt

plotTest1.m – Plot the testing data form test1.txt

plotTest2.m– Plot the testing data form test2.txt

plotTest3.m– Plot the testing data form test2.txt

plotTest4.m– Plot the testing data form test2.txt

test.m - Test the pre-trained neural network (nnids.mat) with the testing data form test*.txt.

Anytest.m – use to test the neural network with a random testing data generated by user (testing data must save in anytest.txt).

plotany.m – plot the graf form anytest.txt.

Testing and training data description

Train.txt - A set of training data set by default. User can generate a new data by overwrite the Train.txt.

Test1.txt - Test4.txt – A set of testing data

BIBLIOGRAPHY

[1] Elaine Rich and Kevin Knight, Artificial Intelligence, McGraw-Hill, Inc., 1991,pp.3

[2] Mohamed H. Hassoun, Fundamental of Artificial Neural Network, Bradford Book, 1995, pp. 1-9.

[3] Timothy Master, Practical Neural Network Recipes in C++, Morgan Kaugmann, 1993, pp2, 6.

[4] Simon Haykin, Neural Networks A Comprehensive Foundation, 2 ed, Prentice Hall, 1999, pp2, 6, 10, 21-28,34-36,51-67.

[5] Frank, Jeremy. Artificial Intelligence and Intrusion Detection: Current and Future
http://citeseer.nj.nec.com/frank94artificial.html
Date Referred: 1 August 2001.

[6] Marilyn McCord Nelson and W.T. Illingworth. A practical Guide to Neural Nets, Addison-Wesley Publishing Company, 1994, pp138

[7] Thomas H. Ptacek, Timothy N. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection.
http://www.robertgraham.com/mirror/Ptacek-Newsham-Evasion-98.html
Date Referred: 30 July 2001.

[8] Ming-Yuh Huang, Thomas M. Wicks. A Large-scale Distributed Intrusion Detection Framework Based in Attack Strategy Analysis.
http://citeseer.nj.nec.com/388567.html
Date Referred: 30 July 2001

[9] Alan M.Davis, Software Requirements Objects, Functions, and States, 1993, pp 7-11.

[10] Tan, Kymie. The Applicaton of Neural Network To Unix Computer Security. http://citeseer.nj.nec.com/tan95application.html
Date Referred: 5 August 2001

[11] Cannady,James. Artificial Neural Network for Misuse Detection. http://citeseer.nj.nec.com/cannady98artificial.html.
Date Referred: 7 August 2001

[12] Ryan,Jake, Meng-Jang Lin, and Risto Miikkulainen. Intrusion Detection with Neural.
http://citeseer.nj.nec.com/ryan98intrusion .html
Date Referred: 8 August 2001

[13] Mark Grand, Patterns in Java Volumn 1, Wiley Computer Publishing, 1998

[14] Terry Escamilla, Intrusion Detection- Network Security Beyond the Firewall, Wiley Computer Publishing, 1998.

[15] Joel Scambray, Stuart Mcclure, George Kurtx, Hacking Exposed: Network security Secrets & Solutions Second Edition, Osborne/McGraw-Hill, 2001, pp483-506.

[16] Hans-Erik Erikson, Magnus Penker, UML Toolkit, Wiley Computer Publishing, 1998.

[17] Shari Lawrence Pfleeger, Software Engineering :Theory And Practice,2ed, Prentice Hall International, Inc, 2001, pp321-327.

[18] Bruce Eckel, Thinking in Java, 2ed, Prentice Hall International, Inc, 2000, pp311-348, 689-824.

[19] Marina Bykova, Shawn Ostermann, Brett Tjaden. Detecting Network Intrusions via a Statistical Analysis of Network Packet Characteristics.
http://Zen.ece.ohio.edu/~inbounds/presentations.shtml
Date Referred: 8 December 2001

[20] Wenke Lee, Philip K. Chan. Real Time Data Mining-based Intrusion Detection.
http://Citeseer.nj.nec.com/452795.html
Date Referred: 8 December 2001

[21] Joseph Barrus, Neil C. Rowe. Distributed Autonomous-Agent Network-Intrusion Detection and Response System.
http://www.cs.nps.navy.mil/people/faculty/rowe/barrupspap.html
Date Referred: 12 December 2001