Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et de la Technologie

Université de Tunis

Institut Supérieur de Gestion de Tunis

LARODEC

Neural Network Application to Distributed Intrusion Detection System

Realized by:

KANZARI Dalel

Supervised by:

Pr. MELLOULI Khaled

Juillet : 2004

INTRODUCTION

Many researches point that Neural Network offer improved performance to handle noisy and missing data over conventional programming technologies whish reveal inapt to deal with incomplete and disturbed information.

Intrusion Detection is one of the most perilous problems that affect system integrity, confidentiality and availability. Many security solutions are attempted to resolve this problems but they still remain far from realizing promising results.

Neural Network with its capability to handle problem in an unknown environment, can handle intrusion attacks with a high degree of confidence.

The present memory studies how Neural Network can be applied to the Distributed Intrusion Detection in order to improve system security.

Our dissertation is organized in three parts:

Part I: State of the Arte

This part introduces the two current technologies whish are the Neural Network and the Intrusion Detection System. It is arranged into two chapters:

Chapter 1: Artificial Neural Network

This chapter holds out fundamental concepts about Neural network technology, presents a general framework and the different types of Neural Network and then defines the different aspects of the learning process.

Chapter 2: Intrusion Detection System

This chapter develops eminent notions in the Intrusion Detection, shows its general architecture and working mechanisms. It classifies the Intrusion Detection into Host-based IDS (HIDS) and Network-based IDS (NID). At last it treats an example of IDS implemented with the Petri Net technology.

Part II: Approach of the Application of Neural Network to a distributed Intrusion Detection

This approach presents a new way to improve security systems by applying neural network technology into intrusion detection attacks to assist the handling of right countermeasures.

This part consists of three chapters.

Chapter 3: Neural Network Application for Distributed Intrusion Detection

This chapter introduces the notions about Distributed Intrusion Detection System, the used protocol (IDIP protocol) that allows communication between network components and the Common Intrusion System language (CISL). It presents then our approach, its role in the system and its integrate modules.

Chapter 4: Design of Neural Network IDIP System

This chapter shows in details how the system can be build. It describes its functionalities (modules) and the phases to realize them, the intern architecture and the interaction of the system with the environment.

Chapter 5: Implementation of Neural Network IDIP System

This chapter describes the realization and the setting up of system's module. It presents data structure, algorithms and schemes that assist to construct an Intrusion Detection System.

Conclusion

Chapter 1: Artificial Neural Network

I. Introduction

Machine Learning is a set of technologies that attempt to solve new problems from past experience ,without programming, they are able to develop and process information to solve specific problems by means of the learning process. These technologies have also the ability to extract knowledge from complicated and missing data.
One of these technologies we will deal with the Artificial Neural Network.

ANN is an information processing paradigm that is inspired by the biological nervous systems to solve problems that people are good at.
ANN is configured for specific applications such as pattern recognition, data classification and data clustering, it is used to extract patterns and detect trends that are too complex to be determined by either humans or other computer techniques

In this chapter we will begin with the Historical Background of Artificial Neural Network and then we will discuss the fundamental concepts of Artificial Neural Network

II. Historic Background of Artificial Neural Network

Through a long time ago, neural network researches are attempting to engineer solutions to problems that have not been solved by traditional computing by means of experiences.
They are beginning since 1943 when McCulloch, a neurophysiologist, and a mathematician Walter Pitts modelled a simple neural network with electrical circuits.
In 1949, Donald Hebb wrote the Organisation of behavior. He noticed that neural connections are strengthened each time that are used.
In 1958, Frank Rosenblatt, a neurobiologist, began work on the Perceptron. A single layer perceptron was found to be useful in classifying continuous-valued set of inputs into one of two classes, subtract a threshold and pass one of two possible values out as a result.

Unfortunately, the perceptron is limited and was proven by Marvin Minsky and Seymour Papert in 1969 in their book Perceptron. They demonstrated that even the simple Exlusive_OR(XOR) logical function could not be implemented with a perceptron.

In1959, Bernard Widrow and Marcian Hoff developed models called ADALINE and MADALINE (Multiple ADAptive LINear Element). MADALINE is an adaptive filter which eliminates echoes on phone lines.

After a stagnant period, ANNs researches emerged again in the early 80's with John Hopfield, a renowned scientist, which founded an approach about using neural network for creating mechanics devices.

In 1986, Rumelhart and McClelland published a book that introduced the backpropagation neural network model to the scientific community. This model allowed the NN to learn a much larger class of functions (including the XOR logical function).

Since that time, the field of Neural Network continually progressed.

III. Artificial Neural Network

"A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use". [1]

An Artificial Neural Network is defined as an information processing paradigm so that they receive input data , propagate them into net and produce output data in response to outside stimulus.

The purpose of artificial neural network is to mimic certain processing capabilities of human brain (learning, recognition ,...etc) to solve complex and hard programming problems by the mean of his historic experience.

A. General Framework of Artificial Neural Network

There are many different ANN models but each model can be precisely specified by the following eight major aspects [RHM86] :

- A set of processing units (neurons) ;
- A state of activation for each unit ;

- An output function for each unit ;
- A pattern of connectivity among units or topology of the network ;
- A propagation rule, or combining function, to propagate the activities of the units through the network ;
- An activation rule to update the activities of each unit by using the current activation value and the input data received from other units ;
- An external environment that provides information to the network and/or interacts with it ;
- A learning rule to modify the pattern of connectivity by using information provided by the external environment.

There are too many kind of neural network, distinguished by their architecture, activation and learning rules , but they all match for this general framework.

B. Structure of a single unit:

ANN is composed of a collection of processing elements called neurons that are grouped into interconnected layer.

Basically, a neuron receives inputs from other sources, combines them in some way, performs a generally non linear operation on the result, and then outputs the final result. The figure below shows the inputs ,the core and the outputs of a single neuron.

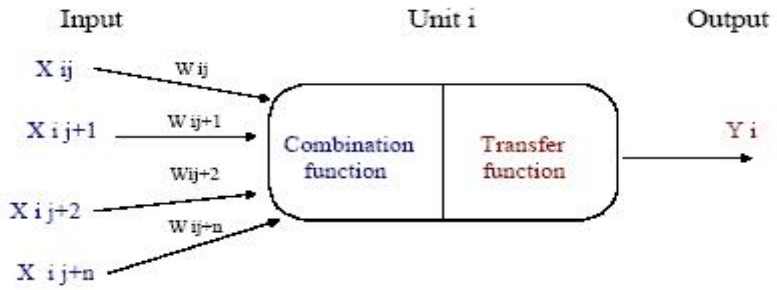


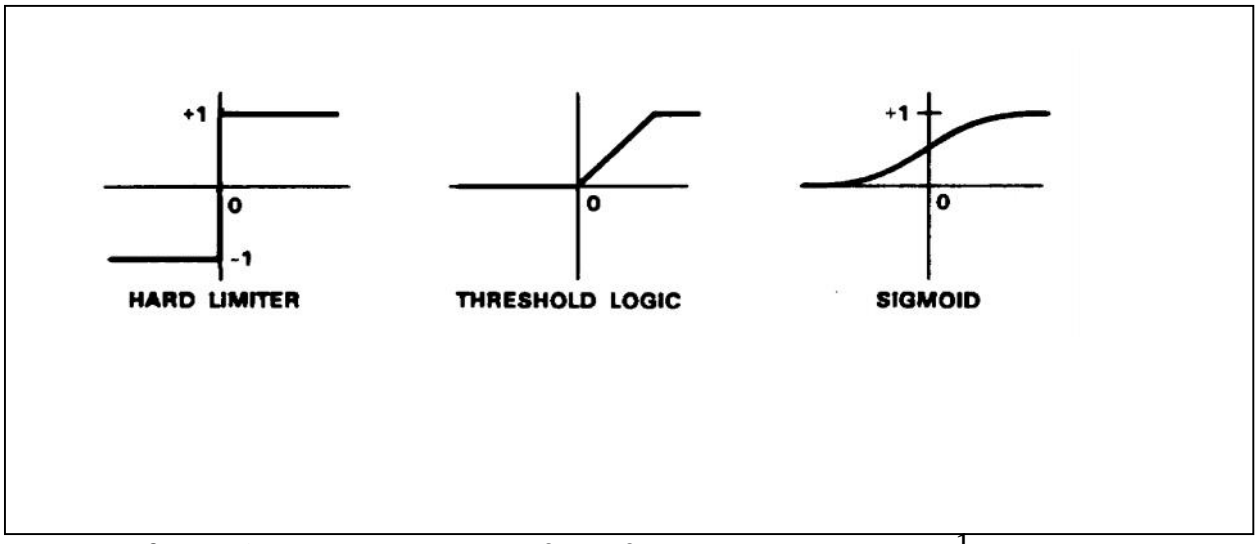
Fig 1 : General model of a processing Unit

- X_{ij} : Input value from unit j to unit i .
- W_{ij} : weight of the connection from units j to unit i . It express the relative importance of input value.

- Combination function: scalar function that add up the values of incoming weights.
- General formula:

$$Y_i = \sum_{j=1}^{j=n} Xi.Wij$$

- Transfer function : function that compute the output value of a neuron.



$$\begin{cases} x \prec 0,\ y = -1 \\ x \geq 0,\ y = 1 \end{cases} \qquad \begin{cases} x \leq 0,\ y = 0 \\ 0 \prec x \leq 1,\ y = x \\ x \prec 0,\ y = -1 \end{cases} \qquad y = \frac{1}{1 + e^{-x}}$$

Fig 2 : Examples of Transfer functions

IV. Architectures of Neural Network

The architecture of the networks defines the way in which neurons are interconnected and information is propagated in the net.

Neural networks consist of a highly interconnected network of comparatively simple processors (called nodes, units or artificial neurons) each one has a set of input and output data. Information is contained in the spatiotemporal relationship between neurones.

Connections between neurons are pondered by valued weights that measure the strength degree between nodes. These measures are updated once time during the leaning phase and then become constant during the use of the neural network.

The basic types of neural networks are :

A. Feedforward Networks

Input value travels network in only one way from input to output neurons. There is no feedback, the output of any layer does never go back. Feedforward networks tend to be straight forward networks that associate inputs with outputs.

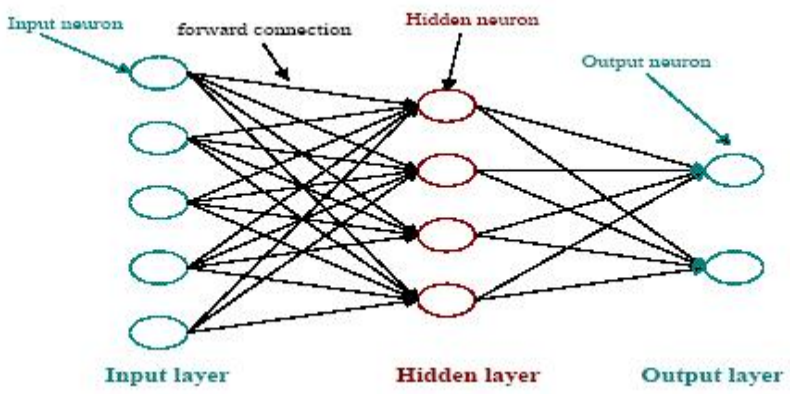


Fig 3 : Typical Feedforward Layered Networks

The input and output neurons are accessible to the training environment. In between these two there are hidden neurons which are not accessed by the environment. The task of these units is to develop an internal representation of the training set and to extract feature from environment.

B. Feedback networks (recurrent nets)

Input value travels network in both directions by introducing loops in the network.

They are dynamic and their state is changing continuously until reaching an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found.

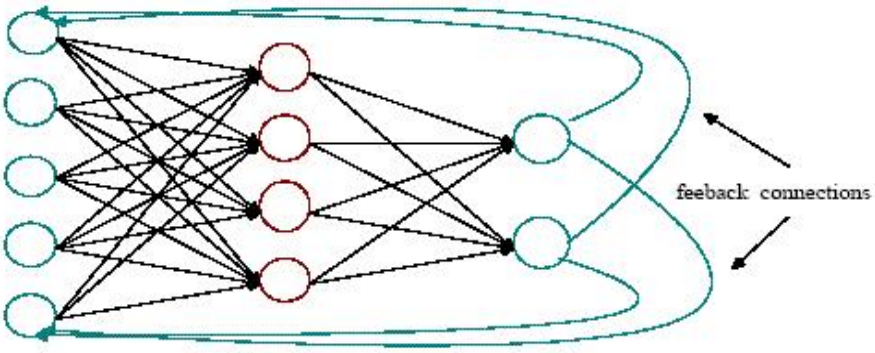


Fig 4 : Example of Feedback Networks

C. Competitive learning networks

These are a kind of hybrid, where a feedforward structure contains at least one layer with intra-layer recurrence. Each neuron is connected to a large neighbourhood of surrounding neurons by negative or inhibitive weighted inputs, and to itself via positive or excitatory inputs. These lateral connections are usually fixed and do not partake in the learning process. It is also connected to the previous layer by a set of weighted inputs.

The point of the lateral recurrent links is to enhance an initial pattern of activity over the layer, resulting in the firing of only one neuron , hence is the label of ‘competitive’ or ‘winner-take-all’ networks.

The object here is to encode groups of patterns in a 1-out-of- n code, where each class is associated with an active neuron in the winner-take-all layer.

V. Types of Neural Network

Several types of neural networks exist. They can be distinguished by their structure (feedforward or feedback) and the learning algorithm they use.

The type of a neural net indicates directives connections between neurons : Feedforward neural nets allow only connections between two different layers ,while nets of the feedback type can have connections in the same layer.

A. Perceptron

The Perceptron was first introduced by F.Rosenblatt 1958. It is a simple neural net type with two neuron layers (input and output layer) that process only 0/1 values .

The net can solve basic logical operation AND or OR (but not XOR) and classify patterns.

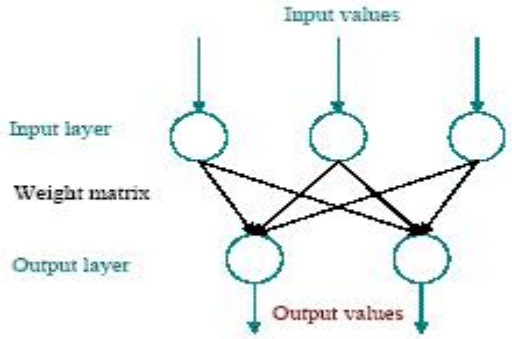


Fig 5 : Sample structure of the Perceptron

B. Multi-Layer-Perceptron

The Multi-layer-Perceptron was first introduced by M.Minsky and S.Papert in 1969.

It is an extended Perceptron and has one or more hidden neuron layers between its input and output layers. One neurons layer is fully connected to the previous and next layer.

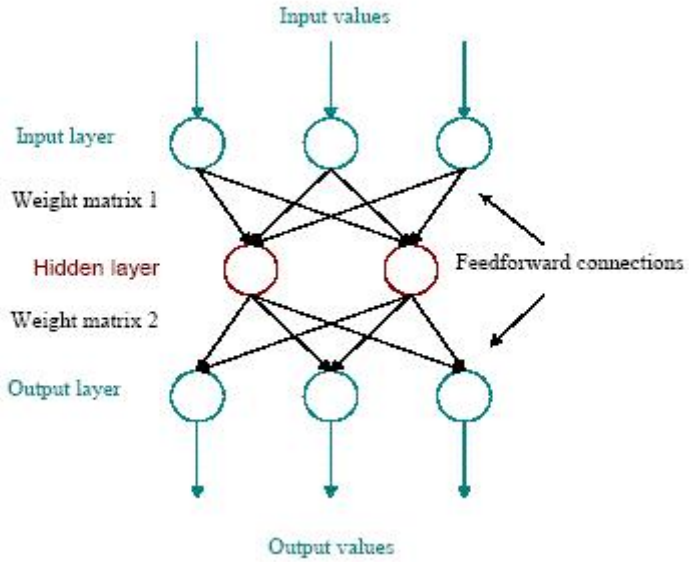


Fig 6 : Structure of Multi-Layer Perceptron

C. Backpropagation Networks

The Backpropagation Networks was first introduced by G.E. Hinton, E.Rumelhart and RJ.Williams in 1986. It has the same structure as the Multi-Layer- Perceptron and uses the backpropagation learning algorithm.

This type of Networks is mainly used to solve complex logical operations, pattern classification and speech analysis.

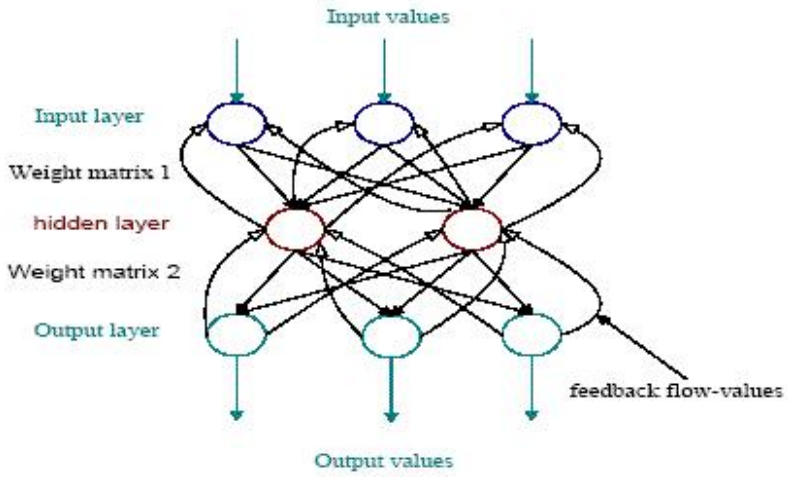


Fig 7 : Structure of Backpropagation Networks

D. Hopfield Net

The Hopfield was first introduced by the physicist J.J.Hopfield in 1982 and belongs to the group of ”thermodynamical models”. It is composed by a set of neurons ,where each neuron is connected to each other neuron. there is no differentiation between input and output neurons.

The main application of a Hopfield Net is the storage , recognition of patterns and the optimisation problems.

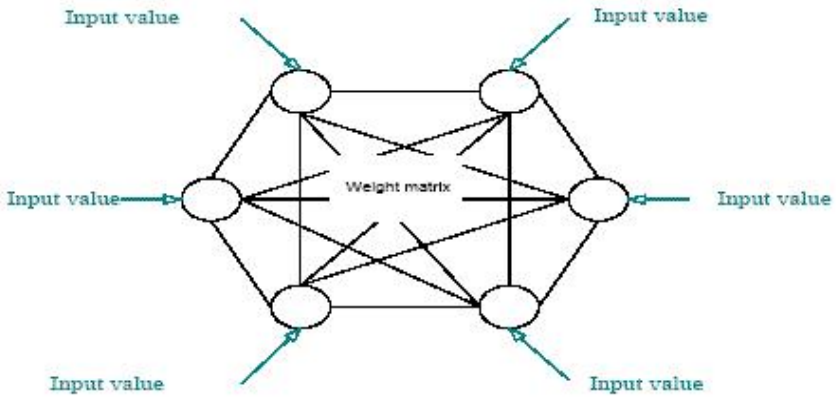


Fig 8 : Structure of Hopfield Net

Weight matrix hold the weight values between connected neurons.

E. Kohonen Feature Map

The kohonen Feature Map was first introduced by Finnish professor Teuvo Kohonen (University of Helsinki) in 1982 and has as aim to simulate the learning process of the human brain.

Networks in which location of output unit conveys information

It have one neuron layer where neurons are organizing themselves according to certain input values. The architecture of this neural net is both feedforward (input layer to feature map) and feedback (feature map).

It is mainly used in pattern classification, optimisation problems and simulation.

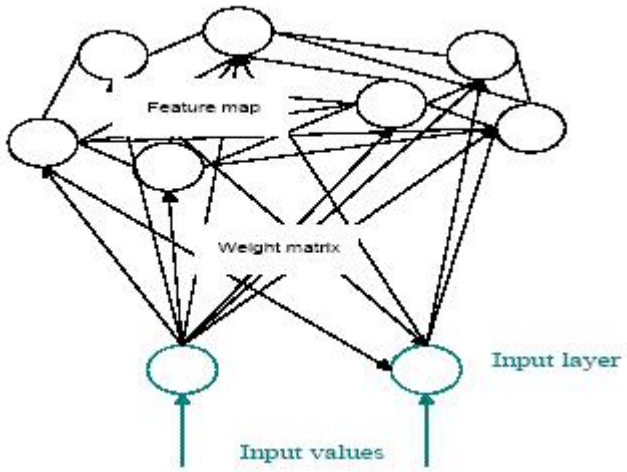


Fig 9 : Structure of Kohonen Feature Map

Feature map :feature space that represent the underlying structure of input date.

VI. Learning Process

Simon Haykin [2] define learning as : “Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The of learning is determined by the which the parameter changes take place”.

Learning process consist in adjusting the parameters of the network (weights, number of neurons,..) to improve its performance. After learning ,the net could react to new external stimulation basing on his memory acquired during this phase.

A. Learning Law

Many learning law are in common use. Most of these laws are derived from the oldest Hebb's Rule.

1. Hebb's Rule

It is the first learning rule introduced by Donald Hebb.

"when an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it. Some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased". [3].

So that if a neuron receives an input from another neuron , and if both are highly active(have the same sign),the weight between the neurons should be strengthened.

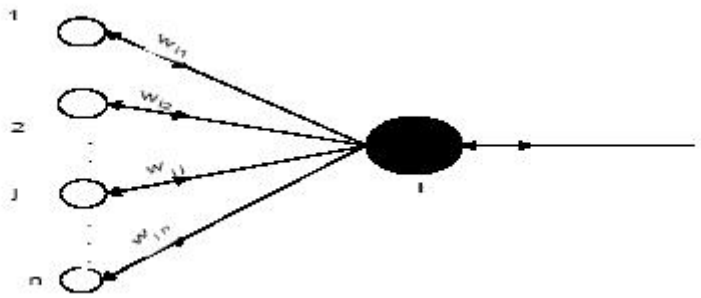


Fig 10 : Signal flow crossing a single neuron

Storing p memories in a Hebbian net :

$$w_{ij} = \sum_{p=1}^{Q} x_i^p \bullet x_j^p$$

Transfer function

$$x_{i(n+1)} = sgn\left(\sum_{j=1}^{N} w_{ij} \bullet x_j(t)\right)$$

2. Hopfield Law

It is a variant to Hebb's rule with the add of magnitudes measures of the strengthening or weakening of the connections. It states : " If the desired output and the input are both active or both inactive, increment the connection weight by the learning rate, otherwise decrement the weight by the learning rate".

3. The Delta Rule

According to S.Haykin Neural Network a comprehension foundation "the delta rule may be stated as: The adjustment made to a synaptic weight of a neuron is proportional to the product of the error signal and the input signal of the synapse in question".

This rule is based on the simple idea of continuously modifying the strengths of the input connection to reduce the difference between the desired output value and the actual output of a neuron.

It changes the synaptic weights in the way that minimizes the Error function (function that compute error) of the network .

Let $w_{kj}(n)$ synaptic weight from neurons j to neuron k at time step n,

the adjustment $\Delta w_{kj}(n)$ applied to the synaptic weight w_{kj} at time step n is defined by :

$$\Delta w_{kj}(n) = \eta\, e_k(n) x_j(n)$$

$x_j(n)$: input value to the neuron k.

η : learning rate determine the step of the variation of synaptic weight.

$e_k(n)$: error measure that compute the difference between the desired response and the output value of the neuron k

$$e_k(n) = d_k(n) - y_k(n)$$

$y_k(n)$: computed output of the neuron k at the n th iteratio

$d_k(n)$: target output of the neuron k at the n th iteration.

4. The gradient Descent Rule

This rule is similar to the delta rule ,in that the derivative of the transfer function is still used to modify the delta error before it is applied to the connection weights. Learning rate is also used in the updating weight.

It is commonly used ,even though it converges to a point of stability very slowly.

$$\Delta w_{kj}(n) = \eta\, \delta_k(n) \bullet x_j(n)$$

$\delta_k(n)$: measure the local gradient of neuron k at the n th iteration , depend on whether k is an output neuron or a hidden neuron:

- **k is an output neuron** :

$$\delta_k(n) = e_k(n) \bullet f'(v_k(n))$$

$v_k(n)$: induced local field produced at the input of the transfer function associated with neuron k (generally equal to the sum of input values of neuron k).

$f'(v_k(n))$: derivative of the associated transfer function.

- **k is a hidden neuron** :

$$\delta_k(n) = \left(\sum_j^m \delta_j w_{jk}(n) \right) \bullet f'(v_k(n))$$

$w_{jk}(n)$: synaptic weight from the hidden neuron k to an output neuron j.

m : total number of output applied to neuron k.

5. Kohenen's Learning Law

The principle that: as two input patterns get closer in input space, the winning output units get closer in output space.

A neuron compete for the opportunity to learn, or update its weight. The winner has the capability of inhibiting its competitors. Only the winner has an output, and only the winner and its neighbors are allowed to adjust their connection weights.

Weights into all units are changed on each update (there are no dead units).

Let i^* the winning output unit, i^* verify :

$$|w_{i*} - x| \leq |w_i - x| \quad (\text{for all unit i})$$

Weight adjustement :

$$\Delta w_{ij}(n) = \eta h_{i,i*}(n) \bullet (x_j(n) - w_{ij}(n))$$

$h_{i,i*}(n)$ neighborhood function that measure the excitation of a neuron by the winner

at the nth iteration.

$$h_{i,i*}(n) = 1 \quad (\text{for } i = i^*)$$

Typical neighborhood function (**Gaussian function**) :

$$h_{i,i*}(n) = \exp\left(-\frac{d_{i,i*}^2}{2\sigma^2(n)}\right)$$

$d_{i,i*}$: lateral distance between winning neuron i^* and excited neuron i, it measures the discrete output space.

$h_{i,i*}$ attains its maximum value at the winning neuron i for which the distance $d_{i,i*} = 0$

$$d_{i,i*}^2 = \| r_i - r_{i*} \|^2$$

The discrete vector r_i defines the position of excited neuron i and r_{i*} defines the discrete position of winning neuron r_{i*}

σ : “effective width” of the neighborhood function, it measures the degree to which excited neurons in the vicinity of the winning neuron participate in the learning process.

Both η and σ generally start large and are decreased during training, their role is to refine the learning of the networks.

These rules above are the basics supports of two principles learning paradigms : supervised learning, and unsupervised learning.

B. Supervised Learning

The learning process of the network is controlled by an outside value called target or desired response.

Network have to adjust its connection weights so that the output computed values come closer to the desired response.

A commonly used rule is the Delta rule.

1. Error Back-propagation Algorithm

Error Back-propagation algorithm follows two consecutive pass :

- **Forward pass** :

  An input vector is applied to the input layer neurons, and its effect propagates through the network until reach the output layer, the latter will deliver a set of output values.

  During the forward pass the synaptic weights of the networks are all fixed.

- **Backward pass** :

  The synaptic weights are all adjusted in accordance with an error-correction rule.

  a. the actual response of the network is substracted from the desired response to produce an error signal.

  b. The error signal is propagated backward through the network, against the direction of synaptic connections. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response.

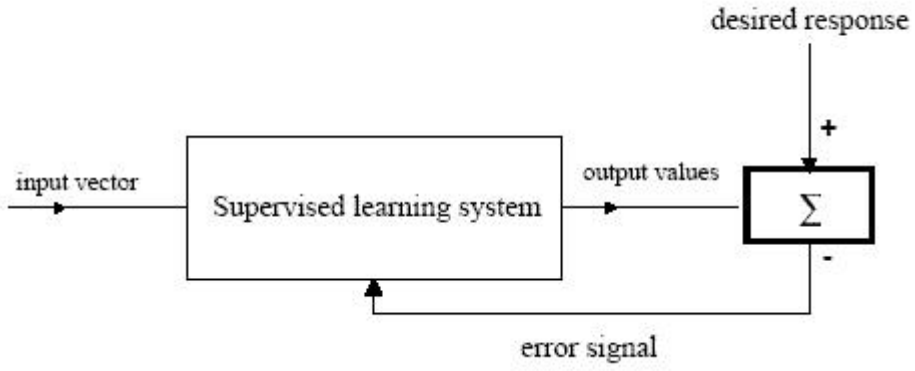


Fig 11 : Error Back-propagation algorithm

C. Unsupervised Learning

During unsupervised learning neural networks have no target outputs to respect, it only process local information.

The net is autonomous : it trend to extract information about input data identify their underlying structure and produce output properties hence the name of self-organisation network.

1. Kohonen's Self-Organisation Map (SOM)

Based on competitive learning, the output neurons of the network compete among themselves to be activated. The result that one neuron, or one neuron per group, can be fired, this latter is called a **winner-takes-all** neuron or simply a **winning neuron**.

There are three essential processes involved in the formation of the Sef-Organisation Map:

- **Competitive process** :

  For an input pattern, each output neuron compute its induced local field, the one which have the largest value among all the other win the competition.

  Let V_k the induced local field of a neuron k and Y_k its output response .

$$Y_k = \begin{cases} 1 \text{ if } v_k > v_j \text{ for all neuron j} \\ 0 \text{ otherwise} \end{cases}$$

$$V_k = W_j * X$$

  X : input pattern , $X = [x_1, x_2 .. x_m]^T$ (m is the number of input neurons).

  W_j : synaptic weight vector of neuron j , $W_j = [w_{j1}, w_{j2}, ...w_{jm}]^T$, $j = 1, 2, ...L$. .

  L : total number of neurons of the network.

- **Cooperative process** :

  The winning neuron determines the spatial location of a topological neighborhood of excited neurons.

  So the winner tends to excite the other neurons in its immediate neighborhood more than those farther away from it Given the different lateral distances between the winning neuron i^* and excited neurons we can

draw the topological neighborhood function by using the Kohenen's Learning Law.

- **Synaptic adaptative process** :

  This mechanism enables the excited neurons to increase their individual values of the discriminate function in relation to the input pattern through suitable adjustments applied to their synaptic weights. The adjustments made are such that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced.

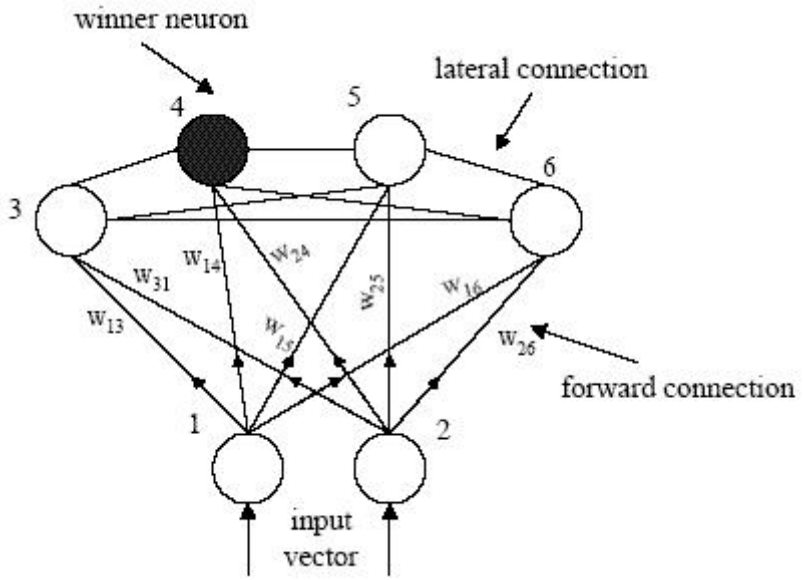


Fig 12 : Kohonen's Self-Organisation Map

VII. Applications of Neural Network

Neural networks have found many successful applications in computer vision, signal processing, character recognition, expert systems, medical image analysis, robotic processing, industrial inspection, and scientific exploration. The application domains of neural nets can be roughly divided into the following categories :

Association, Clustering, Classification, Pattern Completion, Regression and Generalisation, and optimisation.

A. Association, Clustering and Classification :

In this application, input static patterns or temporal signals are to be classified or recognized. Ideally, a classifier should be trained such that when a slightly distorted version of stimulus is presented it can still be correctly recognized.

Equivalently, the network should have a certain noise immunity feature, that is, it should be able to recover a "clean" signal from noisy environments or channels.

- **Association** : there are two association formulations : auto-association and hetero-association.

  The auto-association problem is to retrieve the complete pattern, given partial information of the desired pattern.

  The heter-association is to retrieve a corresponding pattern set B, given a pattern in a set A.

- **Clustering** : the synaptic weights of the network are trained by an unsupervised learning rule. The network adapts the weights and verifies the result based exclusively on the input patterns.

- **Classification** : Based on supervised learning, this application adopts some forms of approximation . Training data consist of pairs of input and output patterns. It is more suitable for the applications problems, which involve a large number of classes with more complex separating borders.

B. Pattern Completion

Information Completion consist in recovering the original pattern given only partial information.

C. Regression and Generalization

Linear or non-linear Regression provides a smooth and robust curve fitting to training patterns. It can be extended to an interpolation problem. The system is trained by a large set of training samples based on a supervised learning procedure. A network is considered successfully trained if it can closely approximate the teacher values for the

trained data space and can provide smooth interpolations for the untrained data space. The objective of Generalization is to yield a correct output response to an input stimulus to which it has not been trained before. The system must induce the salient feature of the input stimuli and detect the regularity. Such regularity discovery ability is critical to many applications. It enables the system to function competently throughout the entire space, even though it has been trained only by a limited body of exemplary patterns.

D. Optimization

Neural nets offer an appealing tool for Optimization applications, which usually involve finding a global minimum of an energy function. Once the energy function is defined, then the determination of the synaptic weights is relatively straightforward. A major difficulty associated with the optimization problem is the high possibility of a solution converging to a local optimum instead of the global optimum.

Chapter 2: Intrusion Detection System

I. Introduction:

The goal of computer security is to provide trust in the computing system functionalities by means of confidentiality, integrity, and availability on system's services.
But with the rapidly growing connectivity to the Internet networked systems become very vulnerable to malicious attacks that aim to destroy its internal structure.

Usual techniques of security, like mechanisms of authentication, authorization and access control, became unsuitable to protect system because they react only preventively before one attack is occurred. So, to provide sufficient security for a computer system, the Intrusion Detection System attempt to find malicious users of the system and to prevent them from destroying its resources.
This chapter is organized as follows : first, we will define Intrusion Detection, after that will present the global architecture of IDS, then we will define different classes of Intrusion Detection System, after that we will present the problematic and at last we will analyze new axes of research to improve IDSs performances and particularly the application of Neural Networks to Intrusion Detection and Automated response.

II. Intrusion Detection

Subversion attempts try to exploit flaws in the operating system as well as in application programs and have resulted in spectacular incidents like the Internet Worm incident of 1988.
According to Edward Amoroso : "**Intrusion detection** is the process of identifying and responding to malicious activity targeted at computing and networking resources." [4]
"An environment for anomaly and misuse detection and subsequent analysis of the behavior of systems and networks".
So Intrusion Detection is a mean of defense against malicious use of system's resources.

1. Intrusion

Intrusion is a set of actions that try to exploit system's vulnerability and to violate security policy. For the operating systems, intrusion refers to unauthorized attempt to access or damage information resources.

It can be defined as "Any set of actions that attempt to compromise the CIA (Integrity, Confidentiality or Availability) of a resource" [19]

Intruders is classified into two types, the external intruders who are unauthorized users of the machines they attack, and internal intruders, who have permission to access the system, but not some portions of it. He further divided internal intruders into intruders who masquerade as another user, those with legitimate access to sensitive data, and the most dangerous type, the clandestine intruders who have the power to alter the core of the system and turn off audit control for themselves.

There are too many types of Intrusion, we will quote below the 6 main types of Intrusion [5] :

- **Attempted break-ins :** which are detected by atypical behavior profiles or violations of security constraints.
- **Masquerade attacks :** which are detected by atypical behavior profiles or violations of security constraints.
- **Penetration of the security control system :** which are detected by monitoring for specific patterns of activity.
- **Leakage :** which is detected by a typical use of system resources.
- **Denial of service :** which is detected by a typical use of system resources.
- **Malicious use :** which is detected by a typical behavior profiles, violations of security constraints, or use of special privileges.

2. Intrusion Stages :

Ruiu's analysis of intrusions [6] divides intrusions into seven stages: Reconnaissance, Vulnerability Identification, Penetration, Control, Embedding, Data Extraction & Modification, and Attack Relay:

a. Reconnaissance :

The first part of an attack is to get to know the network topology in the region surrounding the target and the parties it communicates to.

As example of well known tool for net topology identification is "nmap" program and its derivatives.

The objective of the reconnaissance is to understand as much about the target and to identify attack destinations defences and potential attack relay bases.

Il prepare the path to a reliable penetration step.

b. Vulnerability Identification :

The objective of the mapping phase is to find externally accessible traffic paths into the target's net systems. Attackers attempt to download pre-compiled program that exploit and run against targets. The standard script-kiddy technique is to set up a broad address sweep broadcast of probe traffic, to the whole section of the Internet that seeks some sort of response from the target, that would indicate that software is installed with the vulnerability the exploit is using.

The classic vulnerabilities that we frequently see sweeps for are :

- FTP Server Exploits. Especially vulnerable are servers with anonymous write access.
- NFS and SMB share vulnerabilities.
- Holes in POP and IMAP mail delivery servers.
- Vulnerabilities in the "bind" name daemon software.
- Web server CGI exploits (Apache, MS IIS).
- Installed control daemons such as BackOrifice.

The scans for these holes are so common these days that it is difficult to even catalog origins of such scans. These kinds of scans are so commonplace that, as long as traffic volume and frequency is controlled, it is possible to conduct them with relative impunity.

c. Penetration :

In this step the attacker try to enter inside the system. The most successful hack is the one where the target doesn't even know it has been penetrated. The next best thing is that when the intrusion is detected, they won't know where it's coming from. Since the source may be detected, it's better to use attack relays so the attacker's anonymity can be maintained. The general technique is to quickly find some clueless newbie who has put his home system or office server on the net with major vulnerabilities, and use that as a relay.

There are a variety of virtual private network (VPN) and proxy programs, which can be used to relay traffic to inside a protected network and not make the traffic appear on an intrusion detection system .

d. Control :

Once an attacker has access to the network, the focus is on gaining control and removing signs of entry . These steps can consist of installing a small script that contacts an already compromised machine from which a larger program (e.g. Back Orifice) is downloaded. After performing these operations, a cleanup of log files (e.g., system, event files, file integrity checker files, and ID systems files) is performed by rewinding these files to their pre-attack positions. Ruiu says, "you can completely remotely control a machine and run programs on it, upload and download data, without any indication to the user."

e. Embedding :

In this phase, the attacker ensures that he can still retain control, even in the event that he is discovered. Since gaining control as described above puts the attacker in an exposed position, the aim here is to assure that access from this point forward does not generate obvious symptoms. This can be accomplished, for example, by overwriting little-used system files with malicious code that can survive a system reboot. Ruiu describes an attacker who inserted a piece of code that erased itself if not contacted within a specified period of time. Another insidious attack involved inserting code

into an EEPROM of a network card¾thus even when the operating system was reinstalled the exploit was not removed.

f. Data Extraction & Modification :

Surreptitious retrieval of information suggests that data be encrypted and trickled out at a low rate. An additional strategy is to hide outgoing data using a common protocol such as HTTP for cover, perhaps disguising the data as a JPEG or GIF file.

g. Attack Relay :

The final goal of an intruder may be to use the compromised network as a springboard for attacking other systems. Since leaving a history of attack activity from one site makes the intruder vulnerable, it is in his interest to have multiple attack relay sites. These provide the opportunity to distribute an attack, thus reducing the likelihood of being detected.

3. Intrusion Detection Systems IDS

Intrusion Detection Systems (**IDS**) are mainly employed to secure company networks .It has to detect unusual activity and illegitimate use of the system.

The main resource to detect intrusions is the audit data generated by the operating system. An audit trail is a record of activities on a system that are logged to a file in chronologically sorted order. Since almost all activities are logged on a system, it is possible that a manual inspection of these logs would allow intrusions to be detected. However, the incredibly large sizes of audit data generated (on the order of 100 Megabytes a day) make manual analysis impossible. IDSs automate the parsing of the voluminous audit data . Audit trails are particularly useful because they can be used to establish guilt of attackers, and they are often the only way to detect unauthorized but subversive user activity.

Ideally, an IDS has the capacity to detect in real-time all attempted intrusions, and to execute work to stop the attack

Many times, even after an attack has occurred, it is important to analyze the audit data so that the extent of damage can be determined, the tracking down of the attackers is facilitated, and preventives actions may be taken to prevent such attacks in future. An IDS can also be used to analyze audit data for such insights.

IDS can be implemented at various levels: End-User Devices, Servers or Network Devices (Firewall, Routers, etc)

III. Global Architecture of IDS

Intrusion Detection System is based on the Common Intrusion Detection Framework (CIDF) funded by the intrusion-detection working group (IDWG).

CIDF is an effort to develop protocols and application programming interfaces to share information and to be reused in other systems [7].

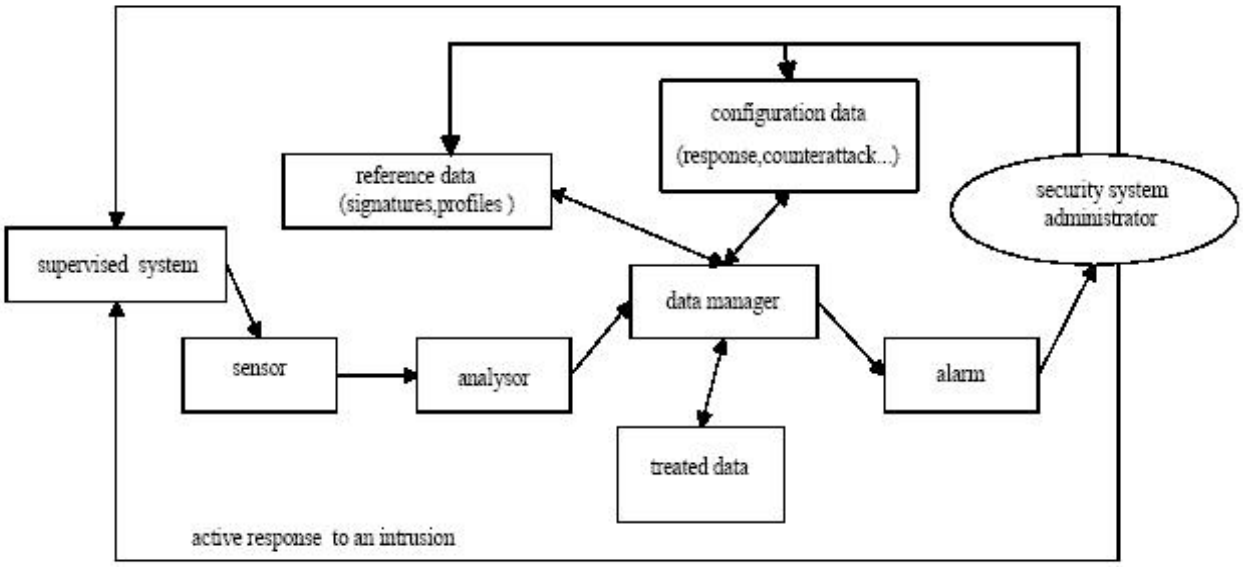


Fig 1 : General Architecture of an IDS

The main components of an IDS are :

- **sensor** : picks up useful data for intrusion detection. It collects information indicative of inappropriate activity that may have resulted from misuse, access, or malicious activity of system components and then sends them to the analysor.

- **Analysor** : analyses collected data and storages them in data base. It collaborates with the sensor to perform data acquisition, formatting and analysis.

- **data manager** : manages the different components of the IDS. It processes events to realize the actual detection process and provide alerts indicating that malicious activity has been discovered. The alerts are sent to the security system administrateur.
- security system administrator :supervise the output of the system and advises appropriate responses.
- **configuration data** : forms a set of responses and counterattack for an intrusion detection.
- **reference data** : contains signatures and user profiles. Signature can be defined as a sequence of events or conditions that indicate an intrusion attempt. User profiles present a set of normal activities of one user.

- **Response** : shows reaction face to an alert. It represent appropriate actions to be taken to protect system from attacks.

Intrusion Detection System Process :

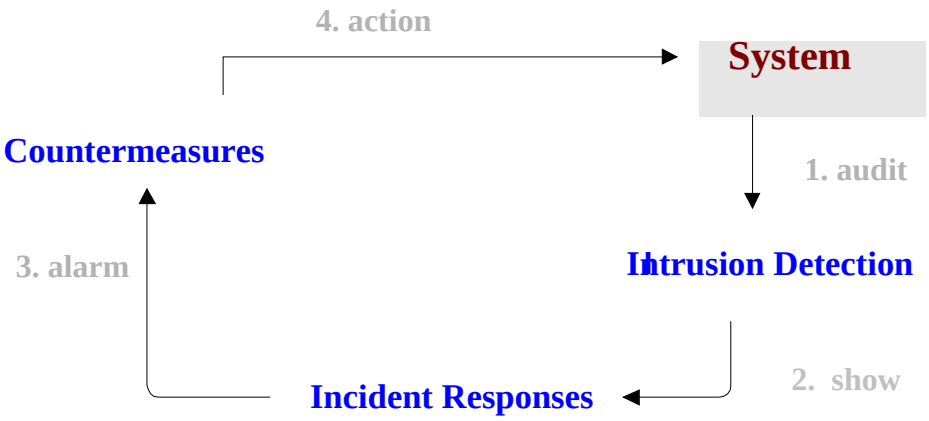


Fig 2 : Intrusion Detection process

We can compose Intrusion Detection process into three steps :

1. **Intrusion Detection System** : detects suspicious data, alert system and shows possible responses against attack. It makes aware about entering data and monitors the network.There are two main types if intrusion detection systems :

    - **Anomaly detection system :** which assumes that all intrusive activities are necessarily anomalous comparing to "normal activity profile" of the system.It detects intrusions by searching “abnormal” behavior compared with previous normal behavior, primarily using statistical means. [8]

      Example : sudden load increase, lost of strangers IP addresses.

    - **Miseuse detection system:** for which attacks are represented as a pattern or a signature to detect known attack. It detects intrusions by looking for activity that corresponds to known signatures of intrusions or vulnerabilities generally stored in a knowledge base.

      Example : scanning, Land attack (source and dest IP are the same).. etc.

      Most IDS based on misuse detection model must always be updated whereas IDS based on anomaly detection model have the ability to detect symptoms of attacks without specifying model of attacks, but the latter are very sensitive to false alarms.

2. **Response :** generally is taken by the security system administrator that choose the most correctly and optimised response to defend system and try to preserve its integrity after attack.We can distingh two types de response :

    - **Active response :** is a mechanism in intrusion detection systems (IDS) that provides the IDS with capability to respond to an attack when it has been detected.As examples of active response we cite the session disruption and the filter rule manipulation of boundary controllers (firewall,router).

- **Passive response :** is limited to genarate alerts, to log files and to notify system through e-mail and pagers.

3. System's countermeasures : present the security policy such that the access control of users, content filtering of input data, encryption, and a set of preventive techniques of security such that the firewall and the router.

Intrusion Countermeasures empower a system with the ability to take autonomous action to react to a perceived intrusion attempt.

Two primary intrusion countermeasure techniques are autonomously acting IDS's and alarmed system resources[9]. Although the former may be considered simply giving intrusion detection techniques teeth, the latter will react to suspicious actions on the system without ever processing audit data to perform "detection" :

- **Intrusion Countermeasure Equipment (ICE)** refer to mechanisms which not only detect but also autonomously react to intrusions in close to real-time. Such a tool would be entrusted with the ability to take increasingly severe autonomous action if damaging system activity is recognized, especially if no security operator is available.

  Intrusion cuontermeasure equipement provide protection to system without the continuous poursue of the system security operator. It assist also to make complexes decisions and response to even automated attacks at a critical time. As preliminary response the IDS can alter filtering rules of the firewall to limit authorizations to attackers.

- **Alarmed Files / Accounts** refer to mechanisms which aim to trap and to lure an intruder by revealing his activities. Accessing an alarmed file or account releases immediate action. Alarms can be silent (only notifying the SSO, even remotely) or can prompt immediate retaliatory action against the intruder. An ideal candidate for an alarmed account is a default administrator account with default password intact.

IV. Classification of Intrusion Detection Systems

Intrusion Detection Systems can be classified into three categories :

- **Host-based IDS (HIDS)** is installed locally on host machines making it a very versatile system that evaluates entering information, including contents of operating systems, system and application files. Attackers attempt to exploit design or implementation flaws of a computer system and attempt to take advantage of hidden features of bugs to gain access to system.

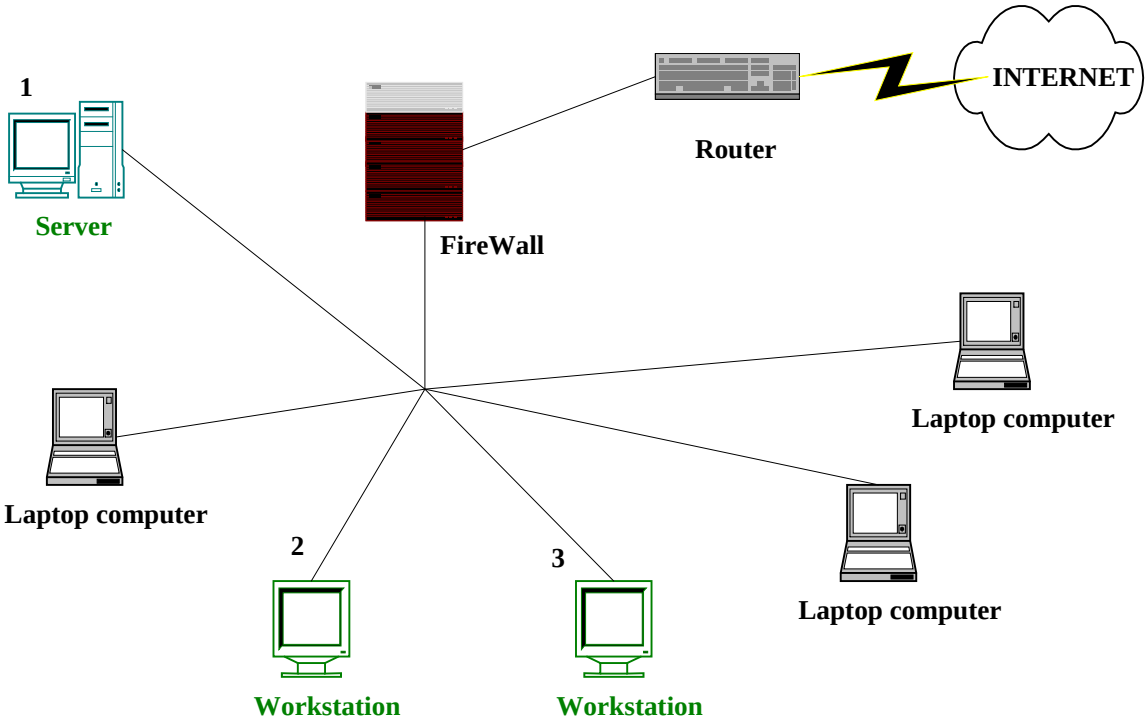


Fig 3 : Typical Example of a HIDS

The figure above shows a typical networks that support HIDS protocol. The numerated machines represent where the HIDS is installed. These devises are protected all time and anywhere.

Example:

- Tripwire[http://www.tripwire.com/products/index.cfml]
- SWATCH [http://freshmeat.net/redir/swatch/10125/url_homepage/swatch]

- GragonSquire[http://www.entersasys.com/ids/squire/]
- Tger[http://freshmeat.net/redir/tiger-audit/3058/url_homepage/tiger]
- Security Manager[http://www.netiq.com/products/sm/default.asp]

- **Network-based IDS (NIDS)** is integrated into network, evaluates information captured from network communications, analysing the stream of packets travelling across the network. Packets are captured through a set of sensors. Intruder try to capture valuable data or secrets by eavesdropping on networks passively or actively: try to catch Password or Credit Card number

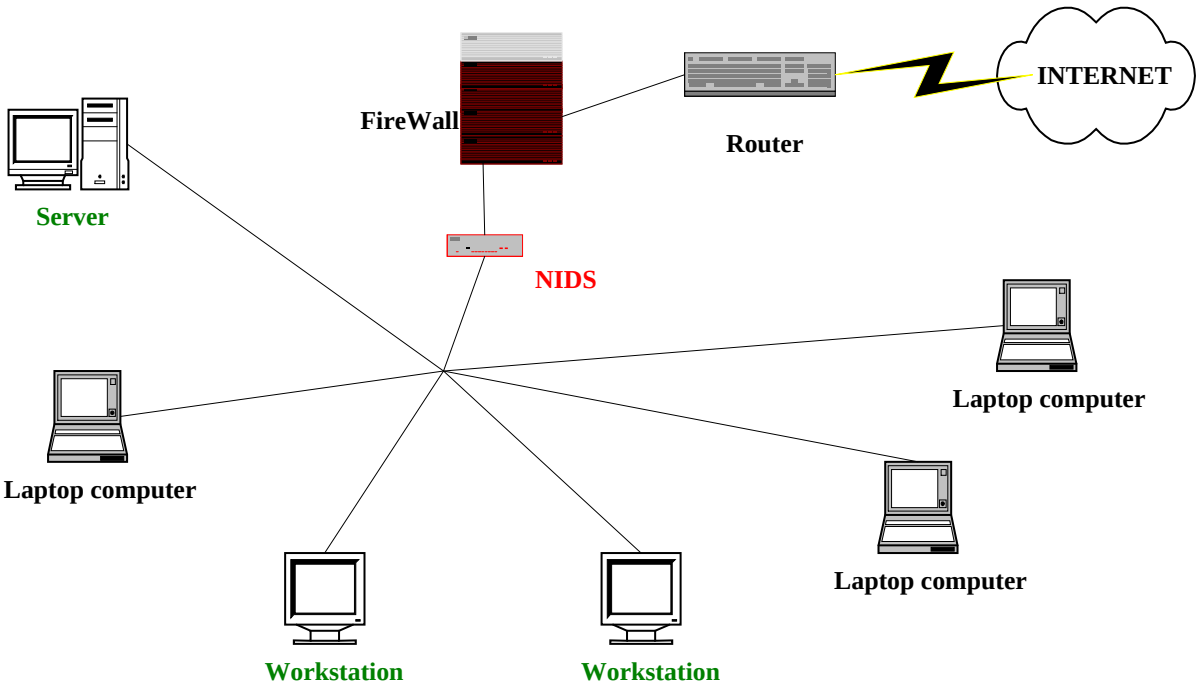


Fig 4 : Typical Example of a NIDS

Figure 4 represents the typical NIDS scenario where an attempt has been made to funnel the traffic through the NIDS device on the network. NIDS supervises input data and monitor the all network's devices.

Examples :

- NetRanger [http://www.cisco.com]
- Gragon [http://www.securitywizards.com]
- NFR [http://www.nfr.net]

- Snort[http://www.snort.org]
- DTK [http://all.net/dtk/dtk.html]
- ISS RealSecure
- [http://www.uh.edu/infotech/software/unix/realsecure/index.html]

- **Vulnerability-Assessment IDS :** a vulnerability (security flaw) can be defined as a weakness in a computing system (bufferOverlow) that allows a violation of the security policy.
  The purpose of a vulnerability-Assessment IDS is to detect vulnerabilities on internal networks and firewalls.

V. Approaches for misuse detection

Misuse detection model compare activity with signatures of known attacks.
Approaches for the misuse detection model are :

- **Expert systems**, contains a set of implication if...then rules that describe attacks. If one rule is verified then an attack is detected and the system is altered to react to the intrusion. It is also possible to add new rules.

- **Neural Network**, analyse missing and deformed data . It can treat audit flow on real-time. Neural Network is used to filter and select suspect information in order to give detailed analyse to an expert system.

- **Genetic algorithm**, defines each scenario as a set of event. It can combines sets to find new possible attacks.

- **Model-Based System** , where attack are modelled as a sequence of events that define an intrusion, it is useful to detect similar intrusions but can't find new attacks.

- **State-transition diagrams**, where a model is created with an initial state for which the system is not compromised .Once the intruder attains the system and

executes a series of actions then the model transits to a new state. It's vulnerable for low-level attack.

- **Signature verification**, where attack scenarios are translated into sequences of audit events.

The common approach for misuse detection concerns "signature verification", where a system detects previously seen, known attacks by looking for an invariant signature left by these attacks. This signature is found in audit files, in host-intrused machine, or in sniffers looking for packets inside or outside of the attacked machine.

VI. Limitation of this Approach

- frequent false-alarm detection (failed to alarm).
- the need to specify a signature of the attack, and then to update signature of attacks on.
  every IDS tool. A signature of an attack may not be easily discovered.
- new attack signatures are not automatically discovered without update of the IDS.
- can't cover all type of serious attacks.
- Not a replacement for a strong security policy.

VII. Approaches for anomaly detection

Anomaly detection model is based on the principle that intrusive behavior deviates from normal system use.

Anomaly Detection in Network-based or Host-based IDS includes :

- **threshold detection** detecting abnormal activity on the server or network, for example

abnormal consumption of the CPU for one server, or abnormal saturation of the network.

- **statistical measures**, learned from historical values, based on the assumption that
  many attackers behave differently from normal users ,or that a system or process behaves differently during attack. If a user is behaving abnormally it may indicate an attacker using that user's account.

- **rule-based systems**, automatically develop rules to explain the historical data they collect. Rules are modified over the lifetime of a system in order to keep the rule set accurate and manageable.

- **non-linear algorithms,** such as Neural Networks or Genetic algorithms. IDS learn profiles of normal behavior and report anomalies to either the operator to another part of the system for more detailed analysis.

The common approach for anomaly detection concerns the statistical analysis, where the user or the system behavior is measured by a number of variables over the time. These variables may be the login and the logout time of each session, the amount of resources consumed during the session, and the resource duration. The major limitation of this approach is to find a correct threshold without frequent false-alarm detection. Also it's difficult to define normal behavior since it can change over time.

VIII. Limitation of IDS

Although the IDS is able to defend and protect the operating system, but it has some limits :

- IDS cannot stop ongoing intrusions on real time.
- IDS cannot trace intrusion with poor authentication of the attackers.
- IDS can only trace intrusion to point of entry to system.
- IDS systems are generally vulnerable to attackers.
- Widely spread attacks may be ignored by the IDS systems.

- IDS systems responses depend upon seeing all traffic.
- Not a replacement for well managed firewall and regular security audit.

IX. Current Development in IDS

There are new axes of research that attempts to improve IDSs performances :

- Distributed and scalable IDS (GrIDS).
- Use of Artificial Intelligence and pattern matching.
- Embedded IDS in network devices.
- Automatic recognition of new attacks (adaptive AI).
- IDS which responds to attacks in progress.

There are a few different groups advocating various approaches to using neural networks for intrusion detection.

A couple of groups created keyword count based misuse detection systems with neural networks (Lippman and Cunningham, 1999 ; Ryan et al, 1998). The data that they presented to the neural network consisted of attack-specific keyword counts in network traffic. Such a system is close in spirit to a host-based detection system because it looks at the user actions.

In a different approach, researchers created a neural network to analyze program behavior profiles instead of user behavior profiles (Ghosh et al, 99). This method identifies the normal system behavoir of certain programs, and compares it to the current system behavior.

Cannady (1998), developed a network-based neural network detection system in which packet-level network data was retrieved from a database and then classified according to nine packet characteristics and presented to a neural network.

X. Example : The Graph Based Intrusion Detection System : GrIDS

The GrIDS is an intrusion detection system that builds dynamically graphs describing network activity by applying user-defined rules to audit data in order to detect attacks or abuse of explicit policies. It generates a graph that used to represent hosts and activity as nodes and edges.

Connections from host to host are reported to allow graph building time and graph shape analysed - eg rapid fanout pattern is detected as a worm, also it can detect network port scans and IP sweeps.

The model allows intuitive aggregation of nodes and edges into reduced graphs that represent the whole system as a single unit. Reduced graphs allow higher level of analysis and data sharing, resulting in a scalable design.

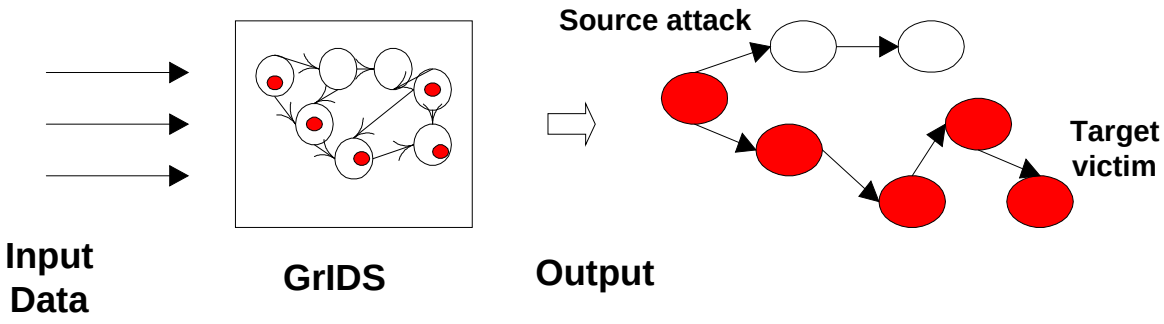


Fig 5 : Example of Graph representation of an attack generated by a GrIDS

The GrIDS can be developed using the Colored Petri Net model as a design specification to improve performance and allow flexibility in system implementation.

1. Petri Net :

A Petri net is a graphical and mathematical modeling tool used to describe and to study systems that are characterized to be concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. The graphic presentation is a design tool that model relationship, activities and states of the distributed components.

The activities are called transitions and are representaed by boxes and states are called places and are represented by circles.

As a graphical tool, Petri nets can be used as a visual-communication aid similar to flow charts, block diagrams, and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems [10,11,12].

Petri Nets have been under development since the early 1960's when Carl Adam Petri defined the language in his dissertation Kommunikation mit Automaten. It was the first general theory of discrete parallel systems that was formulated. The language is a

generalisation of automata theory such that the concept of concurrently occurring events.

2. Petri Net Architecture :

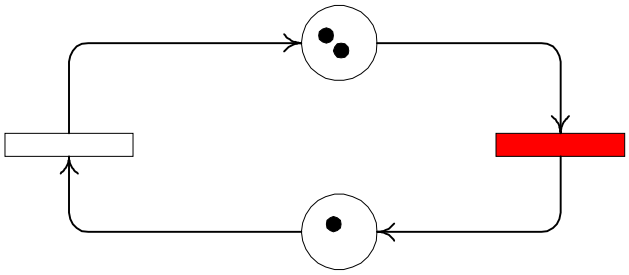


Fig 6 : General architecture of a Petri Net

 : a place that contains two tokens.

 : an enabled transition

 : a disabled transition

A Petri net consists of places and transitions linked by directed arcs, with input directed arcs from places(pre-conditions) to transitions, and output directed arcs from transitions to places (post-conditions). A Petri net can be described as a bipartite directed graph whose nodes are a set of places and a set of transitions.

Places represent passive system components which store tokens and take particular states. Graphically, places are represented by circles.

Transitions represent the active system components which may produce, transport or consume tokens. For each transition there is a set of input places and a set of output places. Graphically, transitions are represented by rectangles.

Arcs connect places with transitions and represent the relations between them. The arc's direction indicates the flow of information (token flow) through the net: from input places to transitions or from transitions to output places.

Arcs have a weight which indicates the number of tokens that are transferred from an input place to the connected transition or from a transition to a connected output place.

A transition is enabled if in each of its input places there is a number of tokens greater or equal to the weight of the corresponding arc. When enabled, a transition removes a number of tokens, equal to the weight of the corresponding arc, from each input place and adds a number of tokens, equal to the weight of the corresponding arc, to each output place.

Marking or state of a Petri net is the position of tokens in the net at any instant in time.
A given marking of a Petri net defines which transitions are able to fire at that time. The firing of a transition moves the net to a new marking.
Formally a Petri Net is defined by the triplet <P,T,W>, with P defines the number of places, T defines the number of transistions and W defines the weight of connexion arcs between places abd transistions.
The state of Petri Net is resolved by a Marking function M that computes the number of tokens to be deplaced throught network.

3. GrIDS Modeling with the Petri Net :

Petri net view of a system is based on two primitive concepts, events and conditions.
The occurrence of events is controlled by the state of the system which can be described as a set of conditions. Condition is a predicate or logical description which may be weather true or false.
For an event to occur some preconditions must hold and the occurrence of an event may cause preconditions to cease to hold and post conditions to become true.

Example :

As an example we consider the following signature which raises an alarm if the number of unsuccessful login attempts for the user xxx exceeds 3 within a minute. The example below only serves to illustrate the idea, and has no bearing on the signature representation format.
login(user = xxx, exception = YES, time = T1) #failed login due to exception
login(user = xxx, exception = YES)
login(user = xxxr, exception = YES)
login(user = xxx, exception = YES, time = T2) if T2 _ T1 < 180s

Its corresponding CP-net is

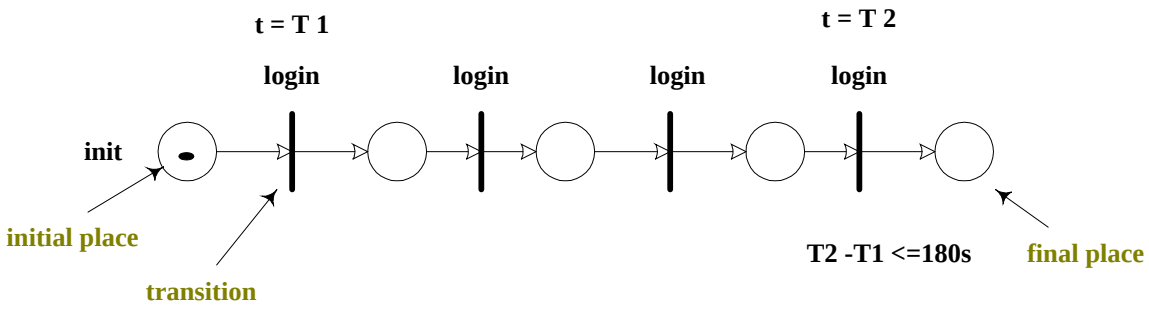


Fig 7 : Four failed logins attempts within a minute

The token encapsulate in the initial place move through the petri net each time when the login failed

XI. Conclusion:

We summarize that an intrusion detection system is defined as a process of identifying computing or network activity that is malicious or unauthorized , but this process still carries problems of *"false negatives"*, wherein an attacker is misclassified as a normal user and problems of *" false positive"*, which degrades the productivity in the system by invoking unnecessarily countermeasures. Also even if the attacks are detected the system can stay too many time out of service to repair damage.

To alleviate to these failures many research efforts are focused to develop distributed intrusion detection prototypes and Frameworks that permit to detect accurate intrusions and to automate response without the intervene of human resource.

Chapter 3: Neural Network Application for Distribution Intrusion Detection System

I. Introduction:

The increased number of heterogeneous interconnected computer gives greater access to outsider and makes it easier for intruders to avoid detection.

Also single Intrusion Detection System reveals inefficient to monitor multiple hosts connected via network and to protect operation systems from distributed hostile intrusions.

So we try to implement Distributed Intrusion Detection Systems (DIDS) that permit to aggregate and to correlate data stream from multiple hosts and the network.

Neural Network (NN) with its capability to treat missing and noisy data can be applied in this unsuitable environment to detect network nodes activities.

Through the present approach we will try to combine these two technologies (DIDS & NN) and to show what contribution can they provide to enhance system's security.

In this chapter we will deal with; at first the Distribution Intrusion Detection Systems, at second the involved protocol for communication between network components, then the motivation for the use of presented technologies and at last we will present our Approach which is "the Application of Neural Network in the Distribution Intrusion Detection System" .

II. The Distributed Intrusion Detection Systems (DIDS) :

A distributed Detection System (DIDS) is an effort to share hard-won knowledge. A DIDS is defined as :

"multiple Intrusion Detection System (IDS) [spread] over a large network, all of which communicate with each other, or with a central server that facilitates advanced network monitoring, incident analysis , and instant attack data." [13]

According to this approach reports are sent fluently from dispersed network hosts to the DIDS director through a communication infrastructure.

High level communication protocols between the components are based on the ISO Common Management Information Protocol. As an example of standard framework we will study the Intrusion Detection and Isolation Protocol (IDIP) and the Common Information Specification Language (CISL) that permit the data transfer between network nodes.

III. Motivation for DIDS & Isolation Protocol (IP)

Distribution Intrusion detection and isolation protocol is an extension of the classic intrusion detection system that aims to monitor the wholeness of distributed network components to improve the system's security face to the variety of hostile attacks.

Current IDS technology is increasingly unable to protect the global information infrastructure due to several problems :

- Keeping trace of distributed attacks and finding with accuracy the attack's source to take the right responses.
- Correlation between incoming events from heterogeneous network components
- Existence of single intruder attacks that cannot be detected based on the observations of only a single site.
- Coordinated attacks involving multiple attackers that require global scope for assessment.
- Normal variations in system behavior and variation in attack behavior causes false detection and identification by a single IDS.
- Detection of attack intention and trending is needed for prevention.
- Automated and autonomous attacks, such that rapidly worms spread, require rapid assessment and mitigation.

- Sheer volume of attack notifications received by host owners can become overwhelming and affect their performances.
- The early intrusion detection research efforts realized the inefficiency of a manual review for a system audit trail. So to identify attacks presented within the voluminous audit data, an automated system is demanded.

So, it is required to develop a generic framework in which different components can share their security knowledge to form a robust construction for the system protection.

It presents a consensus on which attacks are worth prevented and automated countermeasures are entrusted.

IV. Description of the Intruder Detection and Isolation Protocol (IDIP)

Funded by DARPA, the Intruder Detection and Isolation Protocol (IDIP) is an infrastructure for integrating IDSs, boundary controllers (firewalls, routers) and automated response components to work together in a consistent fashion.[14]

IDIP provides cooperation among intrusion detection systems, firewalls, routers, network management components, and hosts so that intrusions that cross multiple network boundaries can be automatically traced and blocked as close to the source as possible without killing the critical processes of the system.

IDIP provides a network management called the Discovery Coordinator to allow components access to services, data management, situation display, access to network management, response policy management and secure communications between diverse components.

IDIP uses CISL as an attack description language.

1. The Common Information Specification Language : CISL

CISL is a standard language that allows different network components to exchange information and to bring together against an attack. The purpose of this language is to share distributed information about attacks and possible responses between heterogeneous components . [15]

CISL is used in IDIP to communicate trace and report information. It provides terms for describing an attack in terms of :

- Time , show the time when an attack is occurred.
- Source, indicate the IP Address of the attacker.
- Outcome, show what did the attacker achieve. For instance, the end result of an attack might be the acquisition of root privileges on a given host.
- Attack class, signalize the conventional form of attack. Each attack has an unique identifying.
- Mechanism, indicate the specific actions by which the attacker attempted the attack.

But not all attacks will exhibit all these aspects. If an attack fails, for example, then there is no real "outcome" : no significant system change has been effected, and the fact that the attack failed will be noted in the expression of the attack class.

CISL can specify response actions that should be taken to trace or to stop the attack. Responsive actions are currently described using three terms : trace, audit, and block. The block response is locally interpreted as whatever action is needed to stop the attack. For example, on hosts, the block response could kill the connection, temporarily disable the service, kill the offending process tree, and disable the user account to which the process belongs. On firewalls, the block response might just kill the connection and install a filtering rule to temporarily disable the service.
CISL is able to specify a wide range of complex attack ,for example it can describe a “worm” traversing a network by the trace messages and attack reports.

2. Framework :

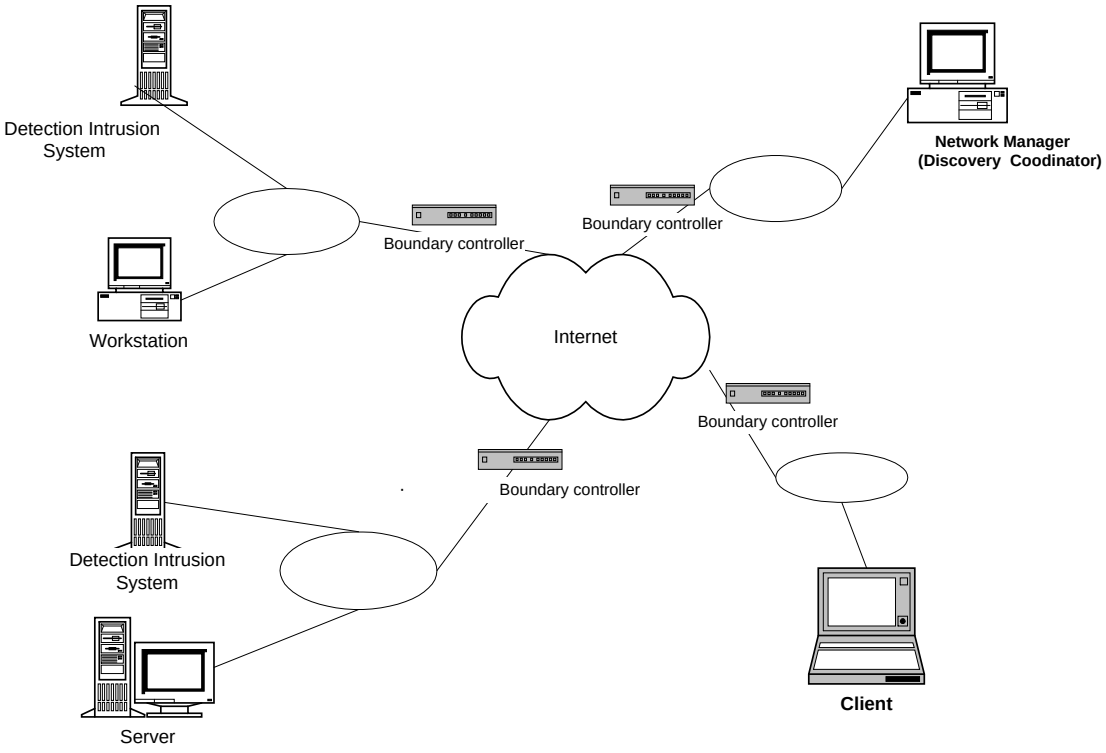


Fig 1 : Framework of the IDIP

This figure shows the various components that can participate in an IDIP-based response :

- Intrusion detection components initiate IDIP response messages, and can support damage assessment and recovery within the local environment.
- Boundary controllers (routers and firewalls) provide network-based response mechanisms by blocking the intruder's access to network resources.
- Hosts (client) provide finer-grained responses by killing processes and connections associated with intruders.
- A centralized network management component (the Discovery Coordinator) receives intrusion reports and audit data from other IDIP nodes and coordinates the overall system response to attacks.

IDIP systems are organized into communities, which represent an administrative domain, controlled by the Discovery Coordinator.

Each community is divided into neighborhoods, where a neighborhood is the collection of components directly connected without Boundary Controllers.

Each component must know its neighborhood to send trace back information.

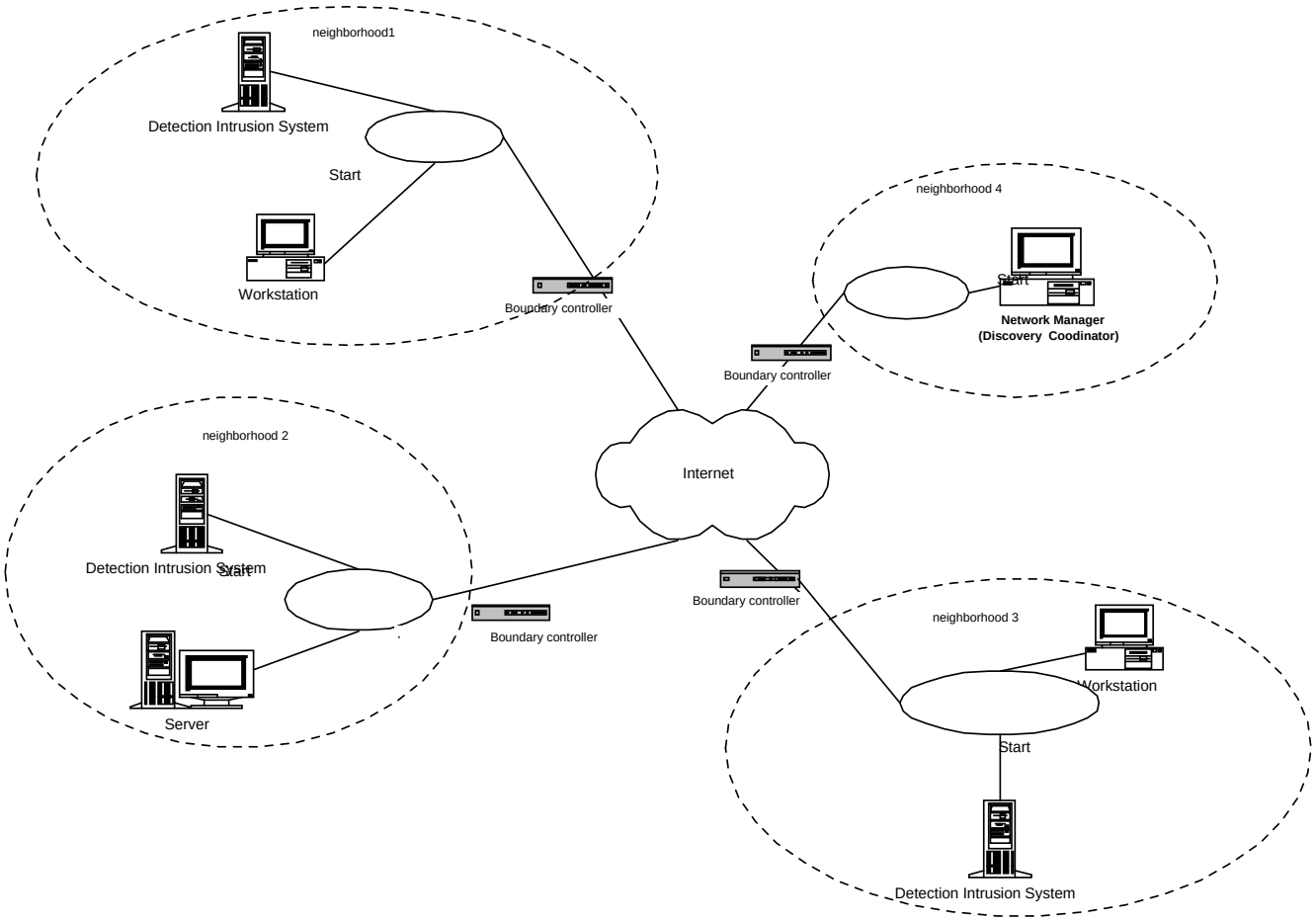


Fig 2 : Organization of the IDIP into communities

3. Protocol:

IDIP protocol engine monitors connection passing through members of IDIP domains :

- If intrusion observed, engine reports it to neighbors.
- Neighbors propagate information about attack and trace connection, datagrams to boundary controllers.

- Boundary controllers coordinate responses by blocking attack, notifying other controllers to block relevant communications.
- Discovery coordinator collects reports from different Boundary controllers and makes essential measure to protect system (example it can isolate affected hosts).

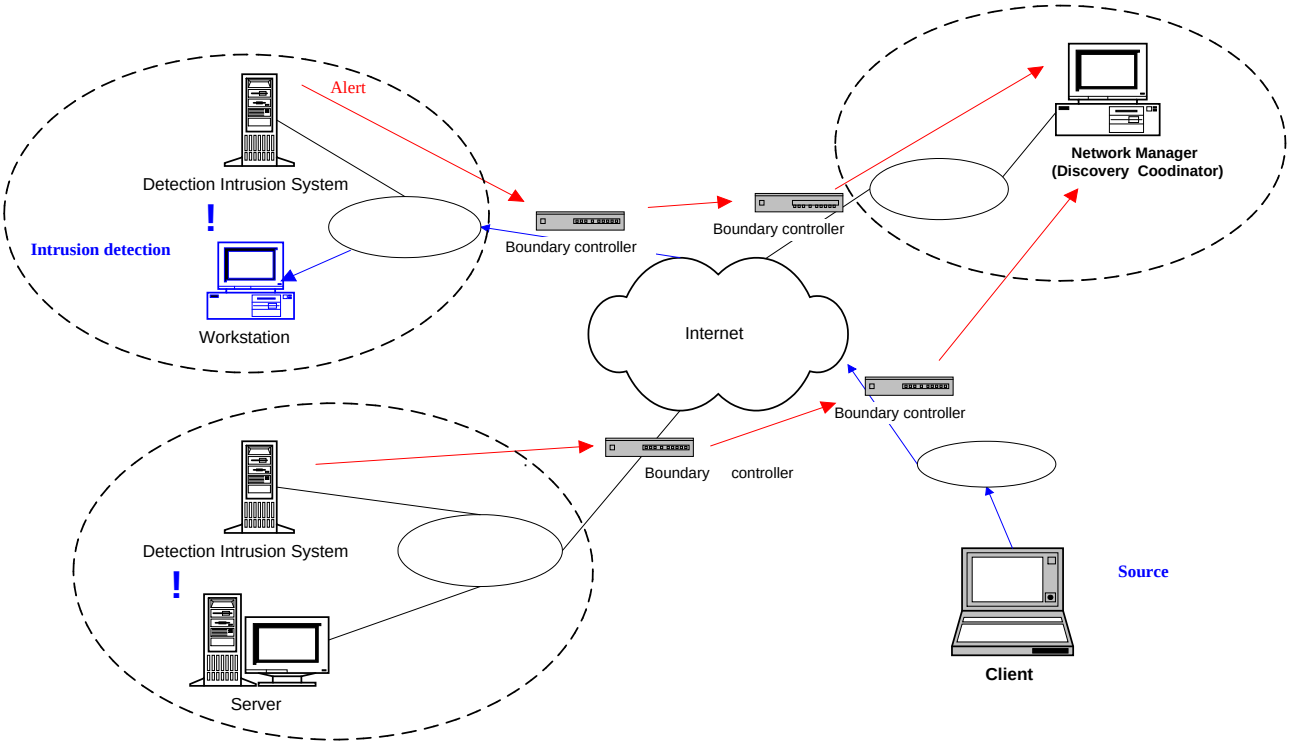


Fig 3 : IDIP protocol

Each IDIP node makes a local decision as to what type of response (killing the connection, installing filtering rules, disable the user account) is appropriate based on the attack type, attack certainty, attack severity relative to the type of attack and vulnerability of components under attack, what other IDIP nodes have already done.

The attack responses are forwarding to neighboring IDIP nodes.

This enables IDIP to trace the attack back to the edge of the IDIP-protected system, taking appropriate responses at each IDIP node along the attack path. Nodes that receive reports from neighbors determine if they are on the attack path (whether they have seen the connection described by the attack report) before forwarding the attack report.

Additionally, each IDIP node sends a copy of the attack report (along with the local responses) to the Discovery Coordinator. The Discovery Coordinator can then correlate reports to gain a better overall picture of the situation, and also issue directive responses back to individual nodes to either remove an unnecessary local response (firewall filtering rule), or add a response (firewall filtering rule along an alternate attack path). The Discovery Coordinator is expected to be co-located with the domain's network management facilities, providing the Discovery Coordinator with the network global topology, enabling the selection of the optimal points in the network to block harmful connections.

4. Software Architecture

The IDIP software was designed for portability and is currently executing on Solaris, BSDI, Linux, and Windows NT platforms. Operating system dependencies were minimized during the development and have been encapsulated in a single file. The IDIP software components consist of the IDIP Backplane and IDIP applications. IDIP applications developed to date include a generic agent and various Discovery Coordinator applications

- **IDIP Backplane**

  The IDIP backplane executes on all IDIP nodes, providing reliable, secure communication between IDIP applications. It provides the neighborhood management (removing duplicate message), the cryptographic key management.

- **IDIP Applications**

  The IDIP applications manage the IDIP message content that is sent or delivered by the IDIP backplane. One IDIP node in a community executes the Discovery Coordinator application. All IDIP nodes execute an IDIP agent application. The latter enable the collection of local data, the monitoring and recording of traffics, the detection of local intrusion and the reporting of attacks description to the Discovery Coordinator.

5. Discovery coordinator's Components :

Due to its engines Discovery coordinator plays a fundamental role in the management of components of system and in the maintaining of its survivability.
These integrated engines are : Correlation engines "Grids", Response engines, Response Manager and other application engines.

- **GrIDS** (Graph based Intrusion Detection System):

  GrIDS builds a graph that represent nodes activities on the network at a given moment. It correlates incoming data from different components including IDS reports to detect distributed coordinated attacks and to monitor the large network.

- **Response engines**

  Response engines produce possible responses to cover the attack damage. They are focused on the designed data provided by the GrIDS. Each one present some cost on the system functionalities. The aim is to choose that has the least cost.

- **Response Manager**

  Response Manager receives a list of effective responses from response engines
  It selects from the list a response with minimal impact on the normal system operation and then sends the selected response to the IDIP enabled responders.
  The objective of this engine is to preserve the survivability of all the system after attack. The Response Manager provides the best solution to the attack.

The figure 4 below shows the internal structure of the discovery coordinator and the exchanged messages between its components

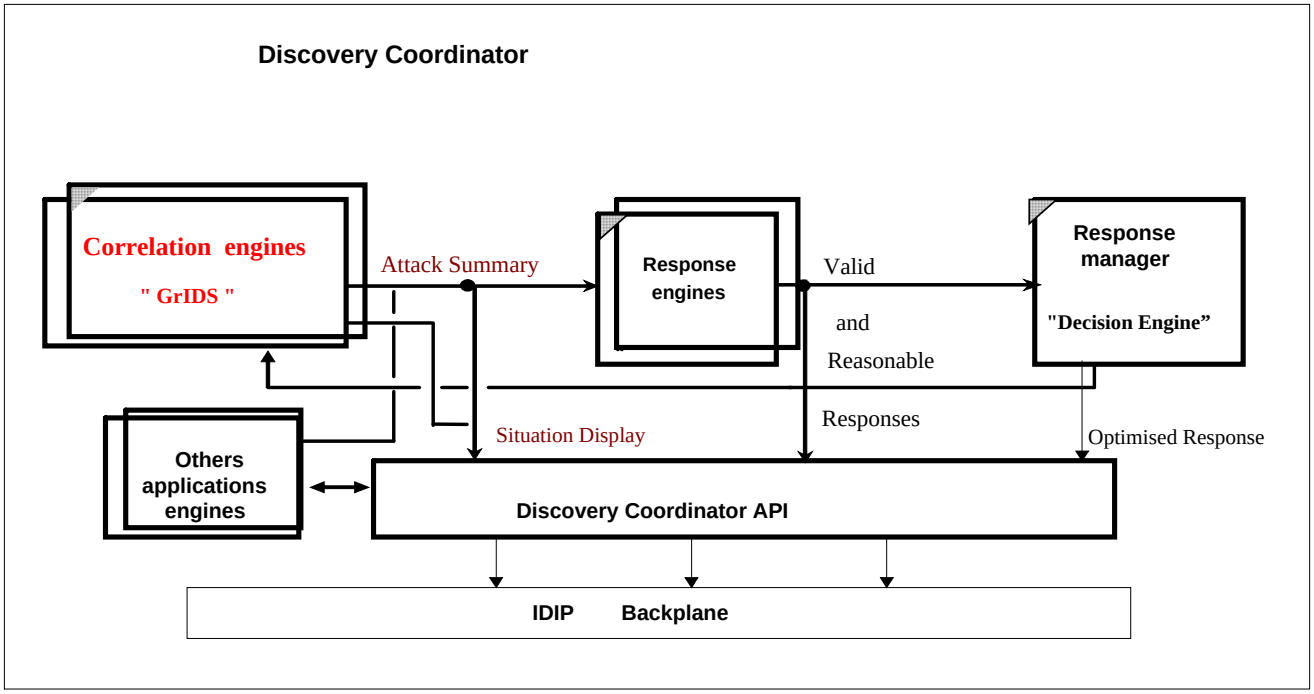


Fig 4 : Discovery coordinator's Components

Our approach is implemented in the first engine of the discovery coordinator which is the GrIDS engine. Through the following sections we will explain who we can apply the network technology to build a robust system's security.

V. The Graph based Intrusion Detection System (GrIDS)

GrIDS is designed to detect large-scale automated attacks on networked systems. The object is to build activities graph that show the intrusion propagation through nodes.

This engine plays a eminent role in the intrusion detection and isolation protocol, since it enables to filter input draw data, furnishs a graphical representation of network components behavior and assists the other engines of the discovery coordinator to take the optimized response to attacks.

His role is to unify the infrastructure of the network, to monitor and to correlate computational remote resources and to organize exchanged dataflow.

To implement the GrIDS, we suggest to benefit from the easy adaptability of neural network technology to an heterogeneous and instable environment.

VI. Neural Network Application in the GrIDS

Our approach presents a hierarchical model to support event correlation and real-time tracking and containment of distributed attacks that cross network boundaries.
We attempt to apply the neural network technology to model the first engine the GrIDS of the Discovery Coordinator.
In this level Neural Network accepts reports sent by different nodes, analyses them and generates a graphic representing repartition attack.

The contribution of our approach is two fold. First, we will provide a framework for distributed attack specification and event abstraction in an heterogeneous context.
Second, we will develop a centralized model that have a high ability to treat missing and noisy data implemented by the neural network. This latter provides the potential to identify network activity based on limited, incomplete, and non-linear data sources

VII. Motivation for the application of Neural Network into GrIDS

We have choose the neural network technology to implement the Graph based Intrusion Detection System because it has many advantages :

- The model is difficult to resolve with classic algorithmic based tool since we dispose of set of composed input variables to whish is associated output response without an accurate relationship between them.

- Input data are implicated with noisy related to the crossing data of the network or to missing data. Neural Network is capable to analyze data from the network, even if the data is incomplete or distorted.

- Neural network permit to visualize topology that exist between input variables.

- Neural Network is flexible to the variations of environment.

- Neural network is able to analyze network activities with data in a non-linear fashion.

VIII. Neural Network_ GrIDS Architecture

Our system is mad up of four principals modules :

- **The Data sensor :** captures the network traffics at the central site.
- **The Data storage :** records data in a specific structure for further interpretation.
- **The Analyses engine :** analyses data and compares them to stored signatures. The core of this engine is based on the neural network technology which treats incoming network data and detects whether they represent a stored attack.

- **The graph engine :** generates representative oriented graph that shows the attack assessment through network.

The figure below shows data streaming among system components.

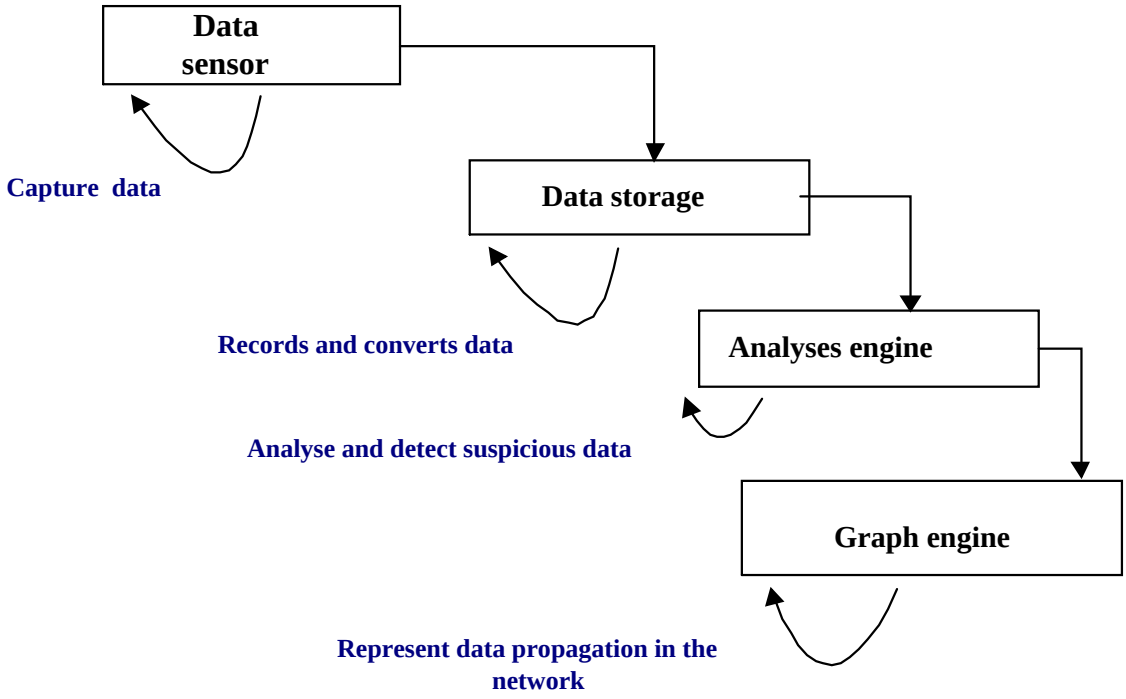


Fig 5 : Neural Network IDIP System components

I. Model description

We will presente below the basic functionalities of system components:

1. Data sensor

Data sensor enables to capture network events and to supply them to the others IDS elements. An event can be represented by a specific packet ,by a sequence of data tram or by a continuous stream of data. It can also be associated with temporal parameters such that date and hour of the tram capture.

This first component represents a boundary between the network and detection system.

In practice, data sensor have the role of a simple sniffer (example TCPDump) which collects raw data from network and send them to other devices without any modification.

2. Data storage

Data storage device has essentially two fundamental tasks :

- **Conversion of input data** : converts incoming network data into [01] interval to be treated next time by the neural network. This conversion is known as normalization. For each type of data there exist a normalization process :
  - Continuous (quantitative) data are converted using a simple rule of three that permit to reduce data interval into [0 1].
  - Enumerative (qualitative) data are represented by many inputs, one for each possible value (modality) ; one of these inputs will be worth one while all others are set to zero.

- **Storage of converted data**: binary data are wrote down into relational tables in a data base. Each column of these tables depict specific information extracted from network captured packets. Recorded data can be used as historic trace for subsequent attacks.

  Data in this stage are ready to be analyzed by the Analyses engine

3. Analyses engine

Analyses engine have a critical role to detect accurately the attack propagation in the network by the comparison between recorded current traffics and known attacks. The analyzes result can be memorised into data base for ulterior observations or simply displayed at the administrative consol.

This engine is under way implementation by others recent technologies such that decision tree and petri net (in the previous chapter we have demonstrate the implementation of the GrIDS by the Petri Net). Our adapted technology is the neural network which is capable to handle missing an incomplete data, to correlate distributed information in an unique support and to detect intrusion attempts.

Fig 6 shows the core of the Analyses engine which the Neural Network and its input and output data.

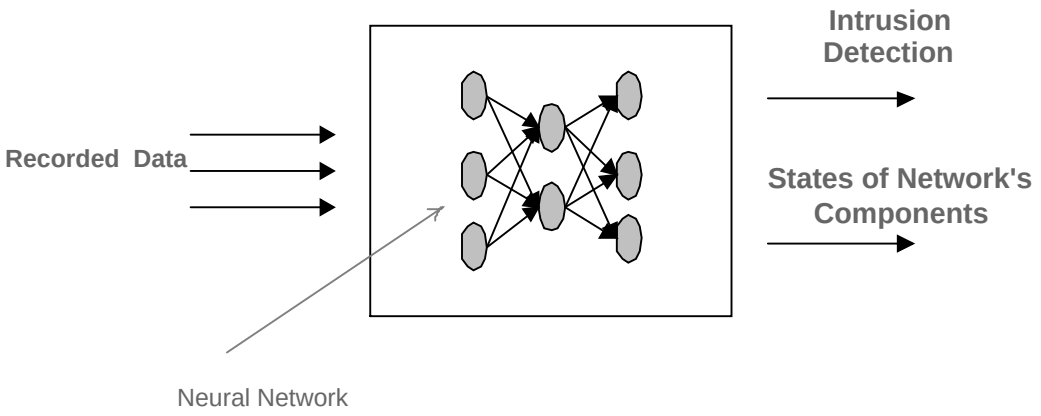


Fig 6 : Analyses engine data process

The working details of this engine is exposed in the next chapter.

4. Graph engine

Graphs consist of nodes, edges, attributes of nodes, attributes of edges. Node's attribute is made up of pair (Clef : Design) values that represent the Address and the Name of the network components. Edges are numbered to mark the order of attack propagation.

Graph engine constructs an illustrative oriented schema representing the spread of attacks among network from source to destination machines.

It assists responder tools to make the right and efficient decision about attacks.

The figure below shows an example of attack propagation through network

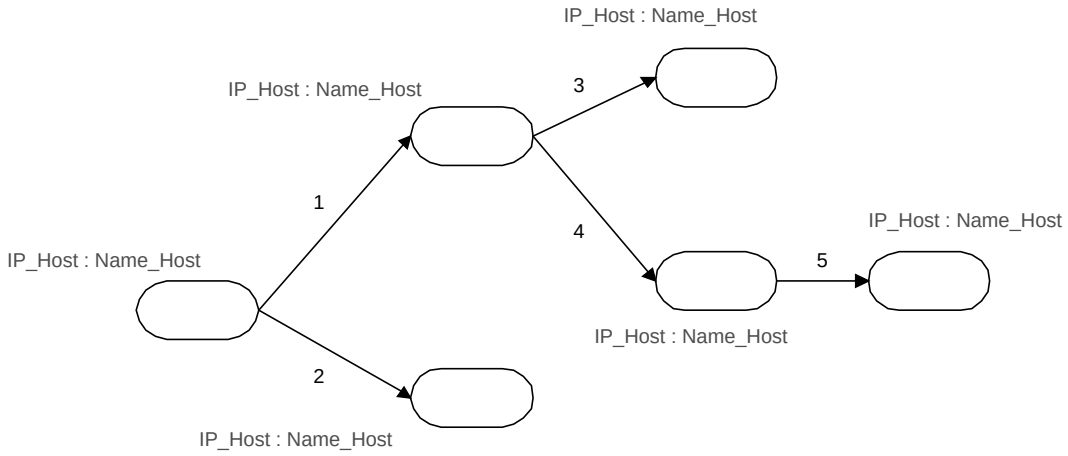


Fig 7: Attack propagation into network

II. Conclusion

The architecture of Intrusion Detection and Isolation Protocol presented here provides a foundation upon for correlated events and scattered information to support real-time tracking and containment of attacks. Neural Network furnish a number of advantage that permit to collect and to treat heterogeneous and missing network data at a central site for the detection of distributed intrusions.

The early prototypes of theses technologies show significant promise, and our future work will involves the refinement of this approach and the development of a full-scale demonstration system.

After the description of our approach and the brief presentation of critical modules of the future system, we propose in this chapter to reformulate our orientations and our objectives in an organized order by designing diagrams.

Chapter 4: Neural Network IDIP System modelling

I. Presentation of the Modeling Methodology

To develop our system we have adapted the Unified Process, a process for software developing around UML "Unified Method Language". This process includes four phases (inception, elaboration, construction and transition) and five activities, or workflows, (requirement, analysis, design, implementation and test) at each phase. [17]

The key tenets that underlie the process are:

- Iterative and Incremental : an iterative cycle allows to develop a complex system through many iterations by evolution. For each iteration we specify, analyze, conceive and implement the use cases. Resulted programs are prototypes whish are evolved throw iteration.
- Use Case Driven : users are the principles actors that define the system's needs. User's requirements present the directive lines for the software cycle development.
- Architecture-centric : defines the structure of the system to satisfy the system's functionalities expressed by use cases to be restricted to evolution and realization constraints.

II. Presentation of the Notation Language (UML)

Object Management Group (OMG) standardized the initial version of the Unified Modeling Language (UML) in 1997. Since then, UML has been very widely adopted by both industry and academia as the language of choice for describing the architectures of software systems. [18]

The Object Management Group (OMG) specification states :

"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.

The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components" . [17]
So the UML is a ' language' for specifying and not a method or procedure used to define a software system; to detail the artifacts in the system, to document and construct. To support the software development UML defines notations and semantics.

UML Diagrams

The combination of diverse diagram allows to have a complete view of the computing system. According to the problem to resolve, we can choice the suitable diagrams.

There exists nine diagrams classified into two categories: dynamic diagrams and static diagrams.

1. Static view

Static view consists of the following diagrams :

- Use Case Diagram: shows the structure of the model by the collection of static elements (classes, packages,..).
- Object Diagram : presents an instantiation of Class Diagram.
- Component Diagram : permits to specify software architecture for a specific development environment.
- Deployment Diagram : describes materiel organization of the operating system.

2. Dynamic view

Dynamic view includes the following diagrams :

- Collaboration Diagram : shows the interaction between objects.

- Sequence Diagram : represents the interaction between objects and their chronological exchanged messages during system's ruining.

- Transition States Diagrams : describes the variation of object's states in response to environment stimulation.

- Activities Diagram : represents different steps related to the execution of an object's operation.

In our approach we will use two static diagrams : the use case and the class diagrams and three dynamic diagrams : Analysis Diagrams, Collaboration Diagrams and Sequence Diagram.

III. Presentation of the Modeling tool : Rational Rose

To design and to develop your system we have adopted the Rational Rose as a modeling 's tool.

Rational Rose enables to model and to draw different types of UML diagrams, it have also others advanced functions that make it a CASE "Computer Assisted Engineering" complete tool.

Among the principal functions, this software allows to :

- Represent all diagram types with the possibility to define stereotypes.
- Depict predefined stereotypes that represent concepts of object programming paradigm (classes, interfaces, index,..).
- Generate codes and keep track of its evolution during system development cycles.

IV. Application of the Unified Process for Neural Network IDIP system development

1. Inception

We will present at this step the Context Diagram (Use Case diagram) and the Domain Model (Business Model).

- **Domain Model :** extracts pertinent objects for the realization of the use cases and their relationship.

  This model is a class diagram that represents general concepts of the application domain without attributes whish will be detailed next time during the conception phase.

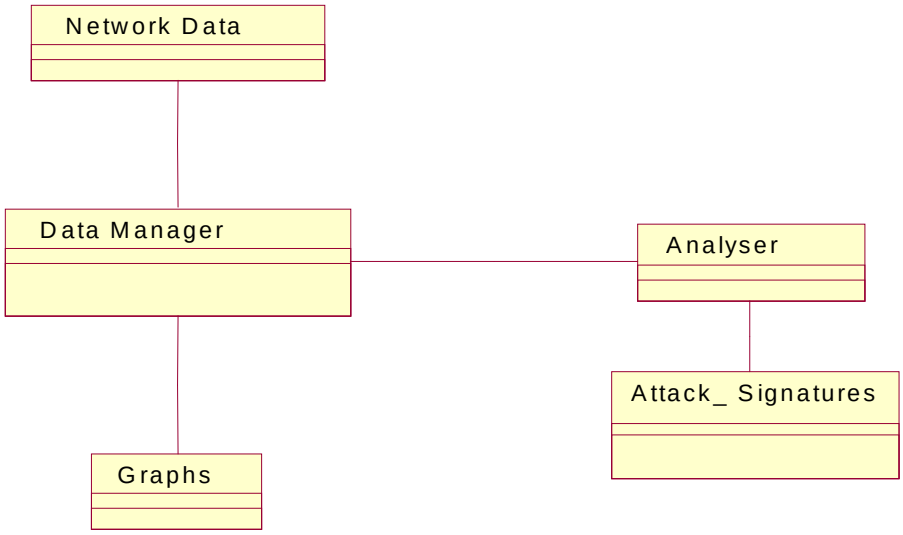


Fig 1: Domain Model of the Neural Network IDIP system

- **Use Case Model** : describes the critical functionalities that the system may support, the actors are called “Business Workers”.

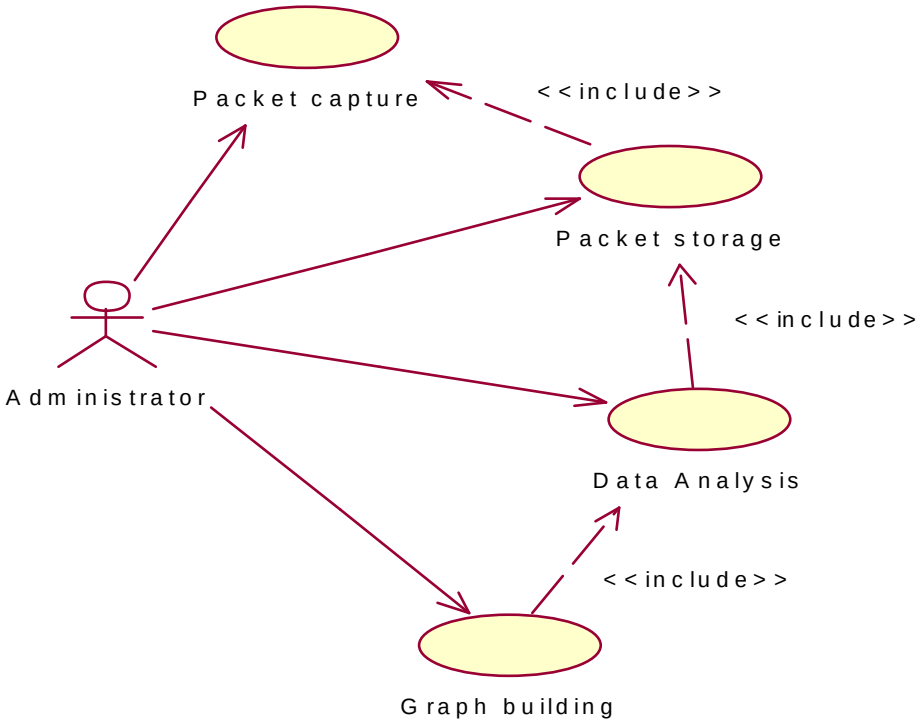


Fig 2 : Description of system Use Case Model

2. Requirement Capture

This phase permit to capture functional requirements of the system through use case.

- **Use case presentation :**

The use cases represent functionalities of a system or a classifier, like a subsystem or a class, as manifested to external interactions with the system or the classifier[18]. It describes a graph of actors, a set of use cases and the relationships between these elements.

- **Actors presentation :**

The principal actor of the system is the administrator that survey the right functioning of the system.

- **Use case description :**

The following figure groups all uses case that illustrate the functionalities of the Neural Network IDIP system.

To detect distributed intrusion, the system may capture network data packets, store them with suitable form in a data structure, analysis data to detect intrusion and build graph to visualize the network attack propagation.

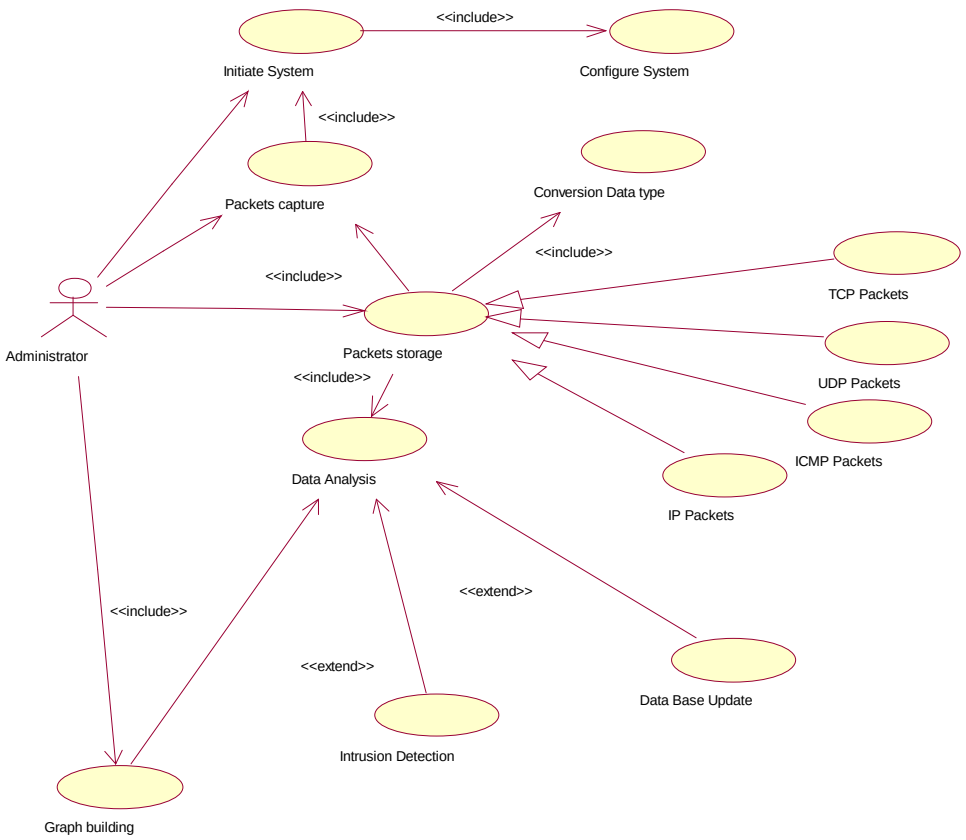


Fig 3 : Use cases description of Neural Network IDIP System

Use Case description :

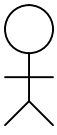
- Configure system : defines the system proprieties that enable it to work correctly.
- Initiate system : starts running system.
- Packet capture : collects network data packets.
- Packet capture : collects network data packet.
- Conversion Data type : converts data into suitable form.
- Data Analysis : analyses data to detect whether there exist an distributed intrusion.
- Intrusion detection : detects intrusion.
- Data Base update : updates the data base by altering or adding new data.
- Graph building : constructs graph that displays the activities of network nodes.

3. Analyze

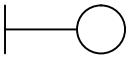
At this stage we analyze the most pertinent use cases by transmuting the use case model on an analysis model.

3.1 Analysis Model :

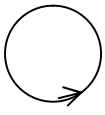
Analysis Model Provides a detailed comprehension of the system requirement and its intern structure. It permits the realization of the uses cases by the structure of connected analysis classes at a high level of abstraction. Each use case is depicted by a set of stereotyped analysis class.



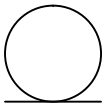
: User, interacts avec the system to get profit from its functionalities.



: Interface, presents the boundary of the system that permit the interaction with the environment.



: Control class, monitors and coordinates between other classes.



: Entity class, presents a persistent element stored in a data structure.

- **Packet Capture :**

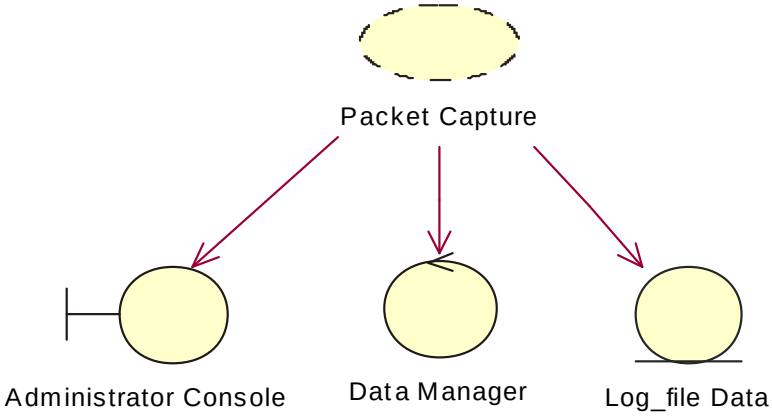


Fig 4 : Analysis model of Packet Capture

Model Description

To capture Packets, we require an administrative console, a principal control class Data Manager that collects Network Data Packets and put them into Log_file Data.

- **Packet Storage :**

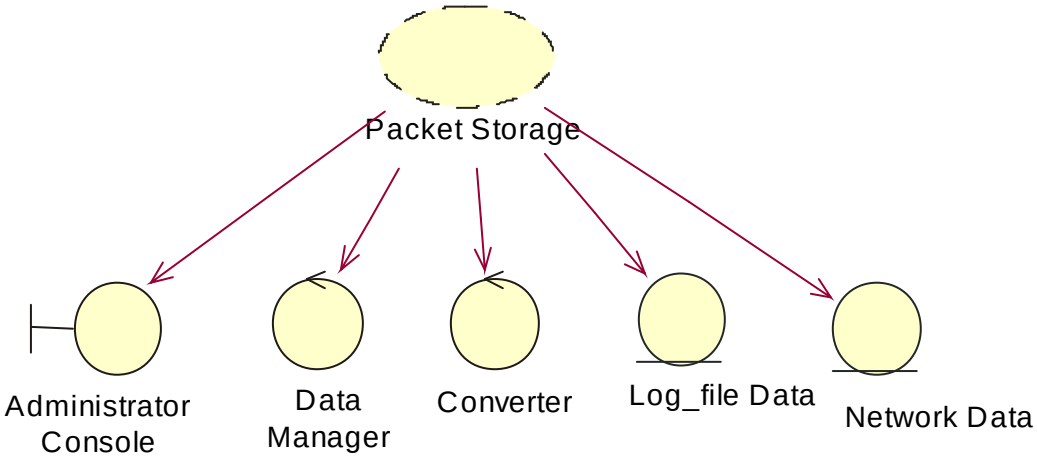


Fig 5 : Analysis model of Packet Storage

Model Description

The realization of Packet Storage use case requires an administrator console, a data manager control class, a converter control class, a log_ file Data entity and a Network Data entity that contains converted data.

- **Data Analysis**

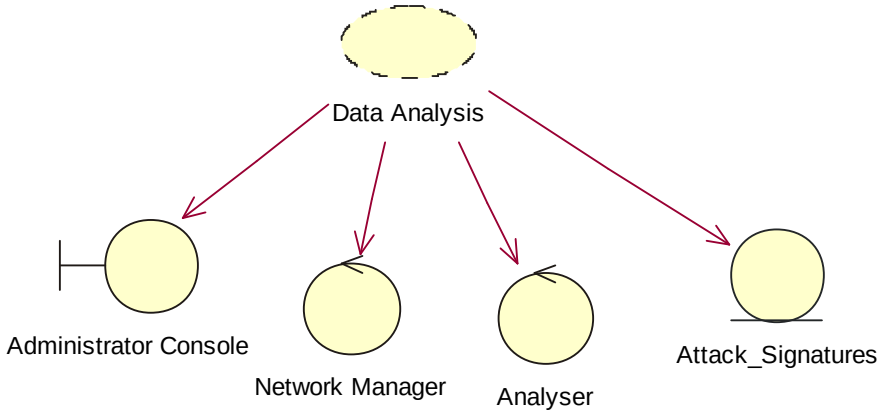


Fig 6 : Analysis model of Data Analysis

Model Description

The realization Data Analysis launch automatically, in consequence to intern stimulus. We need a Data Manager control class , an Analyzer control class, an Attack_Signatures entity that store information about known attacks and an Administrator Console.

- **Graph Building**

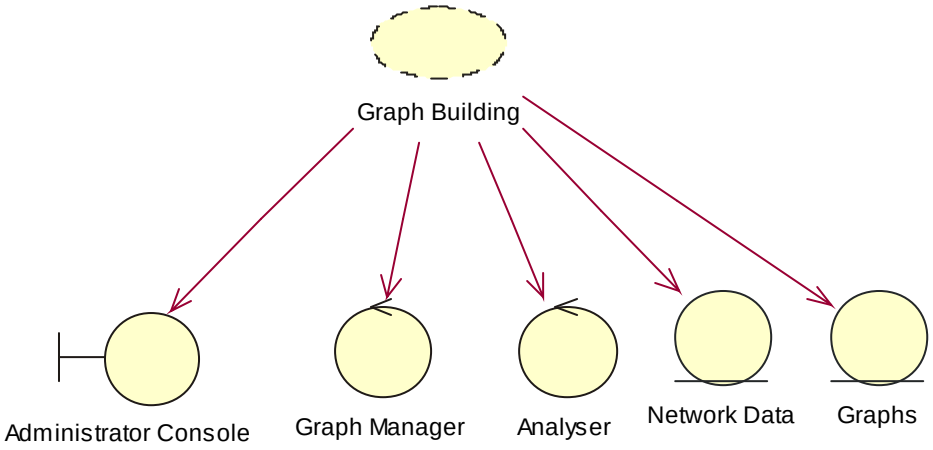


Fig 7 : Analysis model of Graph Building

Model Description

The graph building realization implicates an Administrator Console interface, a Network Manager control class that provides information about the network, an Analyzer control class that detects intrusion attacks and a Graphs entity that contains schemes of attack assessment.

3.2 Class Diagram of the Analyze Model

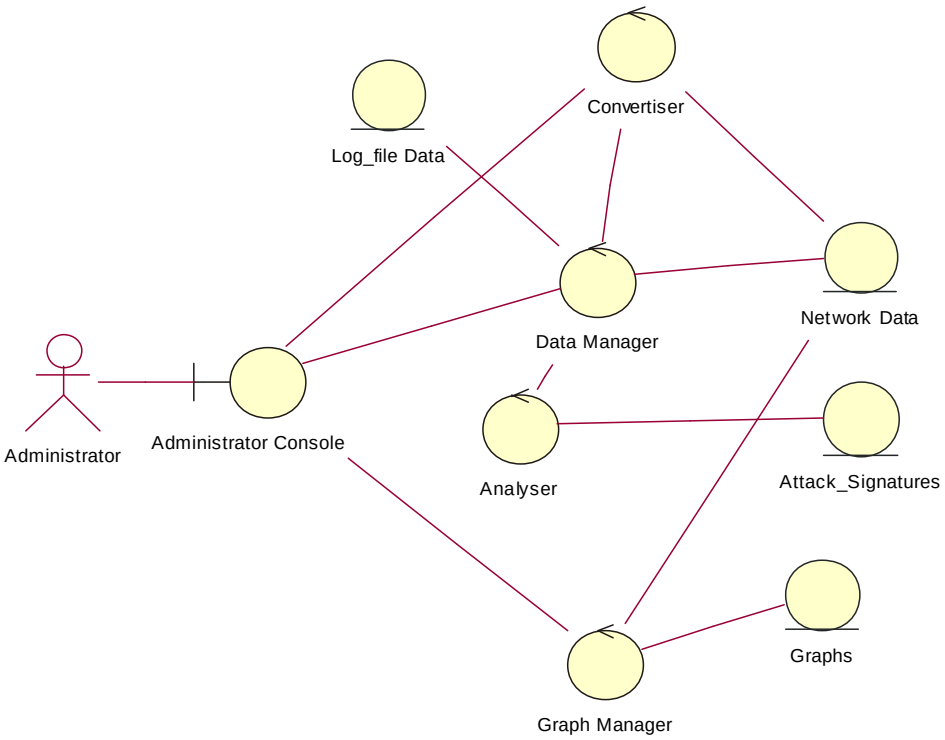


Fig 8 : Class Diagram of the Analyze model

Model Description

Class diagram present the most pertinent classes in the system. The latter is founded on an Administrator Consol interface, four control classes : Data Manager, Converter, Analyzer and Graph Manager, and four entity classes : Log_file Data, Network Data, Signatures Attack and Graphs

3.3 Collaboration Diagrams :

- **Packet Capture :**

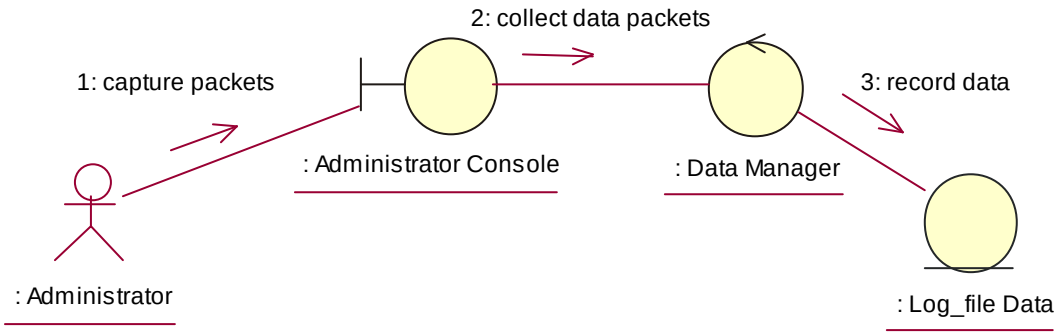


Fig 9 : Collaboration Diagram of Packet Capture

Collaboration Diagram Description :

1. After initialization, the administrator asks the system to capture packets, this operation can be automated by the system.

2. Data Manager collects data packets and put them into Log_file Data entity.

- **Packet Storage**

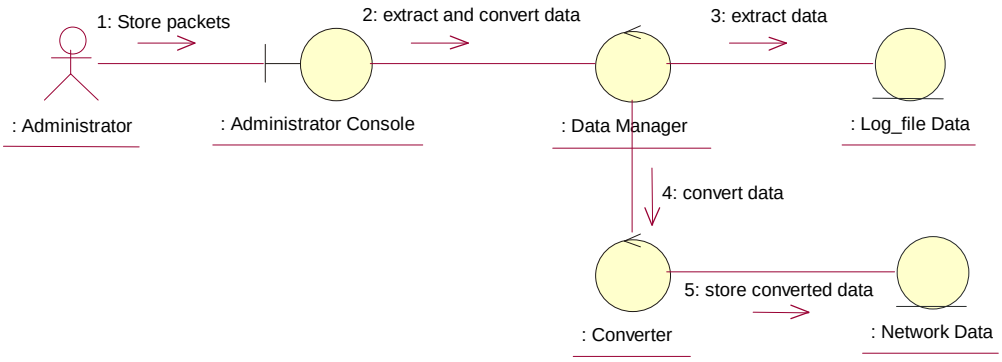


Fig 10 : Collaboration Diagram of Packet Storage

Collaboration Diagram Description :

1. Through an Administrator Console, the Administrator requests to store Log_file Data in a suitable form. This operation can be automated by the system.
2. Administrator Console transfers the message to the Data Manager.
3. Data Manager extract data from log file.
4. Data Manager asks converter to convert data.
5. Converter store data in the data structure Network Data.

- **Data Analysis :**

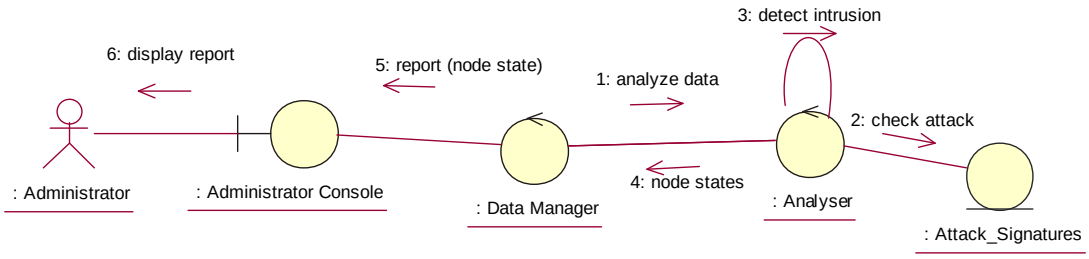


Fig 11 : Collaboration Diagram of Data Analysis

Collaboration Diagram Description :

1. Once the data is ready to be analyzed the Data Manager asks the Data Analyzer to analyses them (This process is automatic).
2. Analyzer checks attack into database Attack_Signatures.
3. Analyzer detects intrusion by opposing the Network Data to attack signatures
4. Analyzer send nodes state to Data Manager
5. Data Manager edits a report about nodes state
6. Administrator Consol displays report to the Administrator

- **Graph Building**

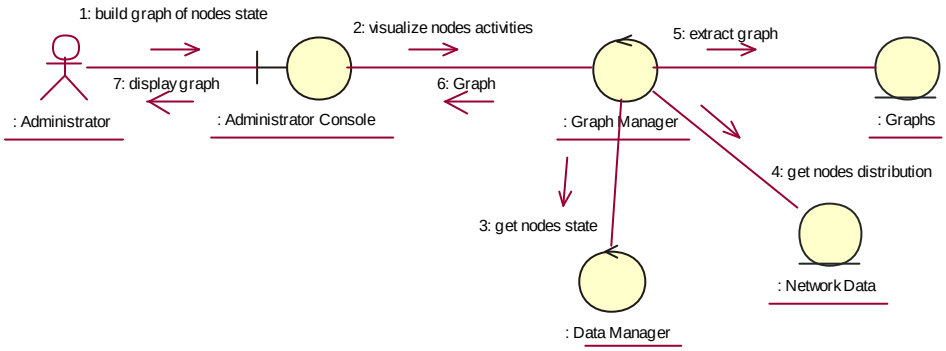


Fig 12 : Collaboration Diagram of Graph building

Description of Collaboration Graph Building :

1. Administrator request to build graph . This operation can be automated by the system.
2. Administrator Console calls the Graph Manager to visualize nodes activities nodes activities.
3. Graph Manager gets nodes states from Data Manager
4. Graph Manager gets network nodes distribution from the network data.
5. Graph Manager extracts graph.
6. Graph Manager sends graph to Administrator Consol.
7. Administrator Consol displays graph to the Administrator

4. Design :

At this stage we present at first two Class Diagrams that predict classes, modules and data structure of an implementation language and then the Sequence Diagram that shows the exchanged data stream among system objects.

Class Diagrams

A Class is a standard UML construct used to detail the pattern from which objects will be produced at run-time. A class is a specification - an object is an instance of a class. Classes may be inherited from other classes (that is they inherit all the behavior and state of their parent and add new functionalities of their own), have other classes as attributes, delegate responsibilities to other classes and implement abstract interfaces.

The Class Model is at the core of object-oriented development and design - it expresses both the persistent state of the system and the behavior of the system. A class encapsulates state (attributes) and offers services to manipulate that state (behavior). [17].

We will present the diagram class of the Neural Network IDIP system. This diagram shows the basic classes that perform system's functionalities.

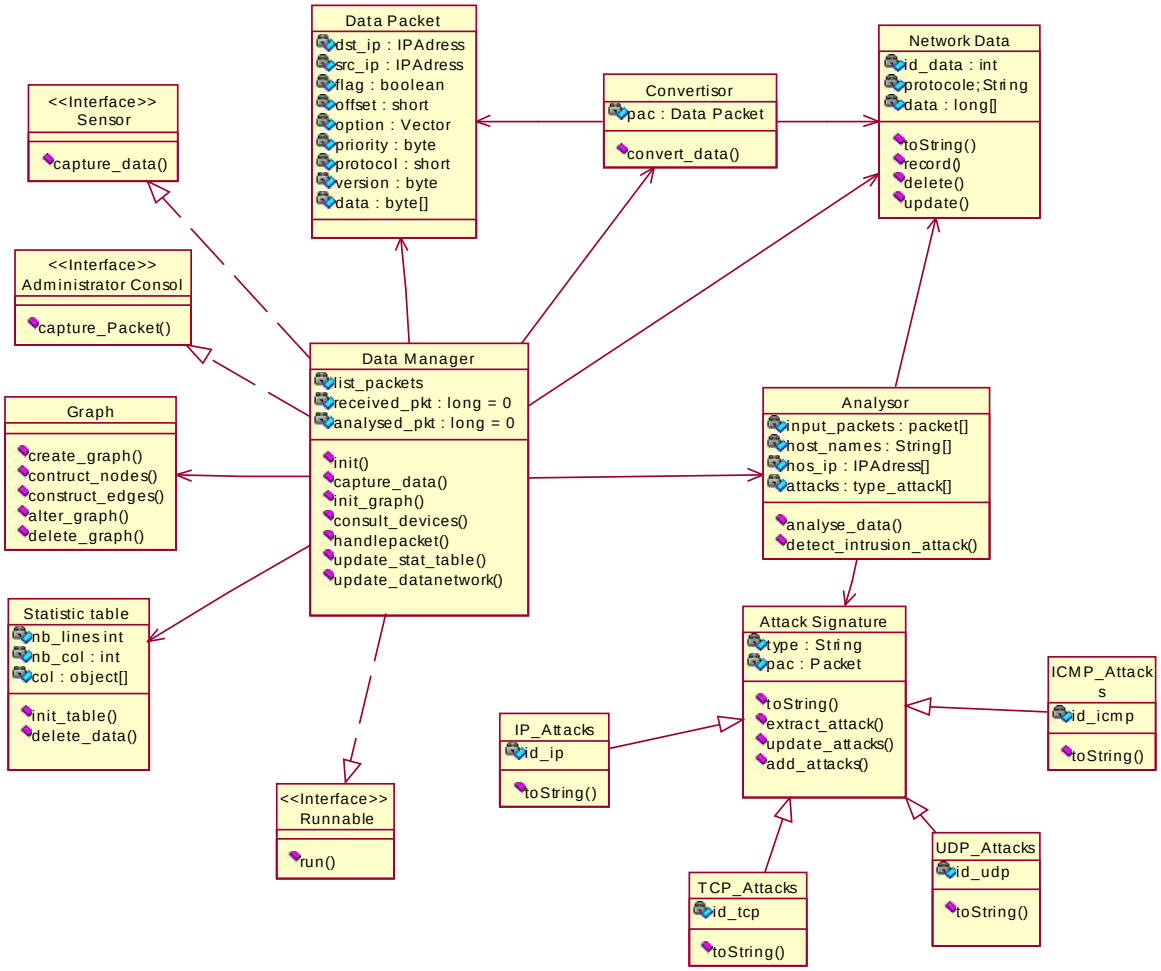


Fig 13 : Class Diagram of the Neural Network IDIP System

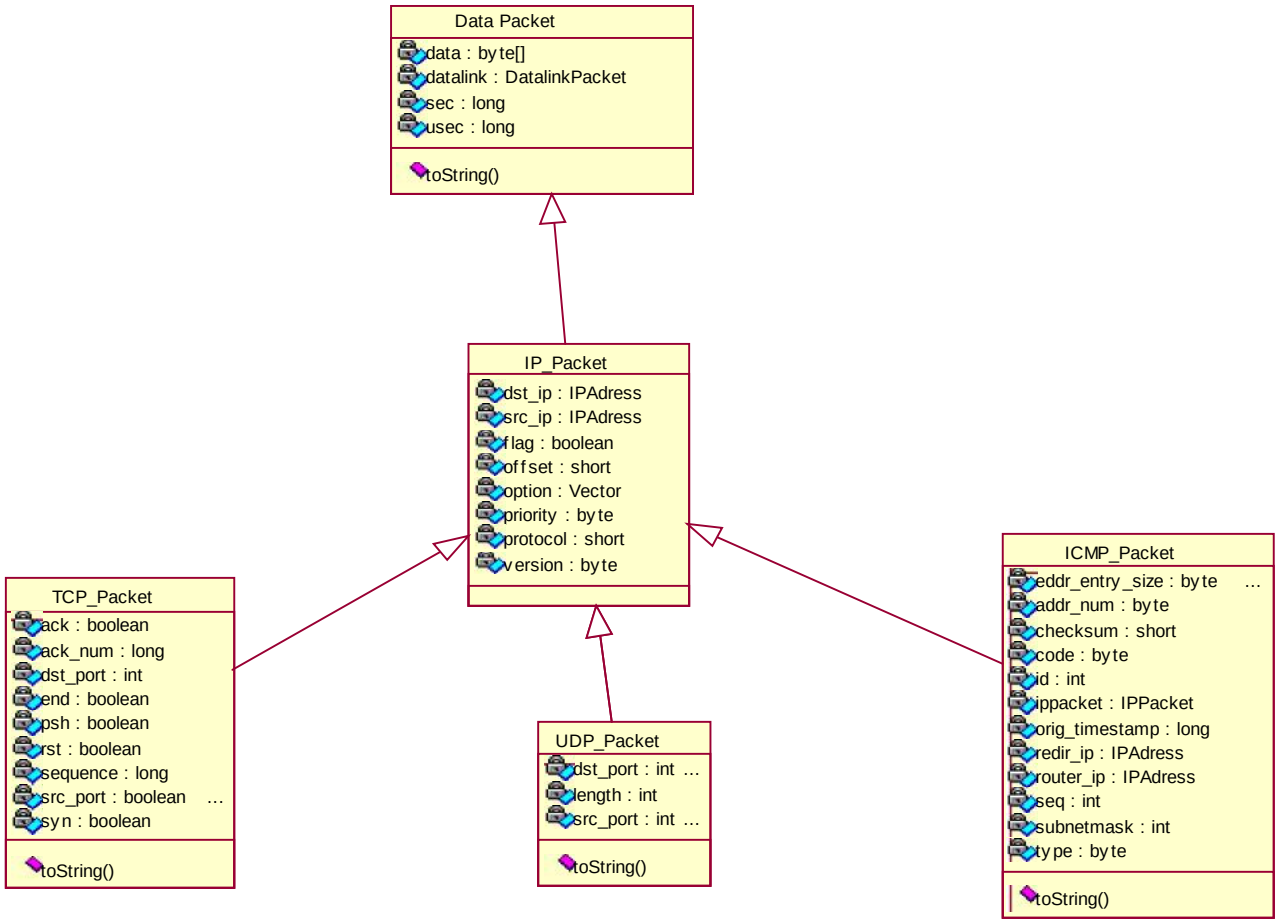


Fig 14 : Class Diagram of Data Packet type

Interaction Models

Interaction diagram provides models to describe how a set of objects interact with each other. The main components of interaction diagrams are the objects and the messages that are sent to each other. UML has 2 forms of interaction diagrams, namely the sequence diagram and the collaboration diagram.

Sequence Models

Sequence diagrams are used to display the interactions between users, screens, objects and entities within the system. It provides a sequential map of message passing between objects over time. Frequently these diagrams are placed under Use Cases in the model to illustrate the use case scenario - how a user will interact with the system and what happens internally to get the work done [17].

- **Packet Capture :**

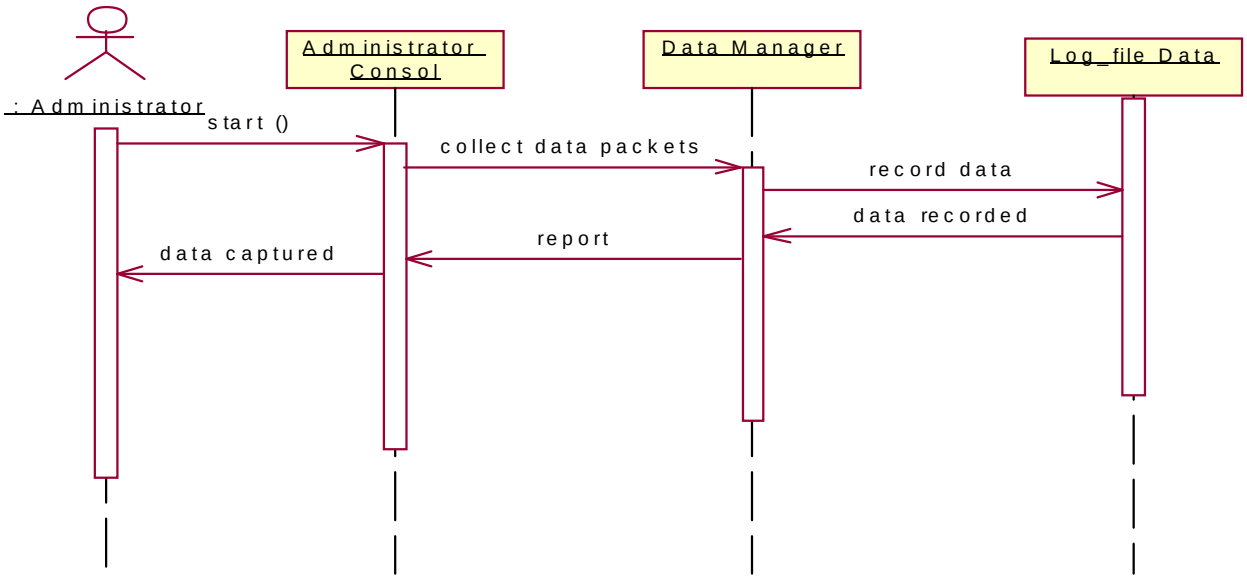


Fig 15 : Sequence Diagram of Packet Capture

Sequence Diagram Description

After system start up , the administrator console require Data Manager to collect network data packets, this latter records data packets into Log_file data structure and then return a report.

- **Packet Storage :**

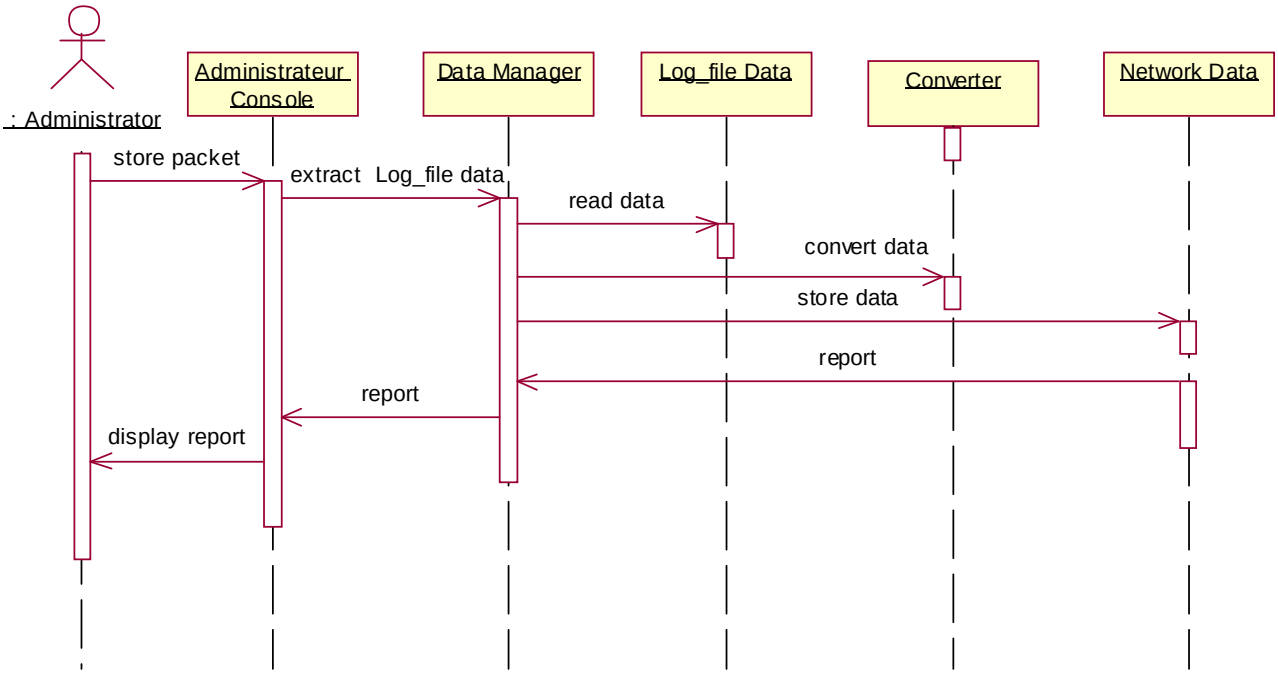


Fig 16 : Sequence Diagram of Packet Storage

Sequence Diagram Description :

Through an Administrator Console, the Administrator requests to store Log_file Data in a suitable form. The Administrator Console transfers the message to the Data Manager whish extracts data from log file, send them to the converter and then stores converted data in the data structure Network Data.

- **Data Analysis**

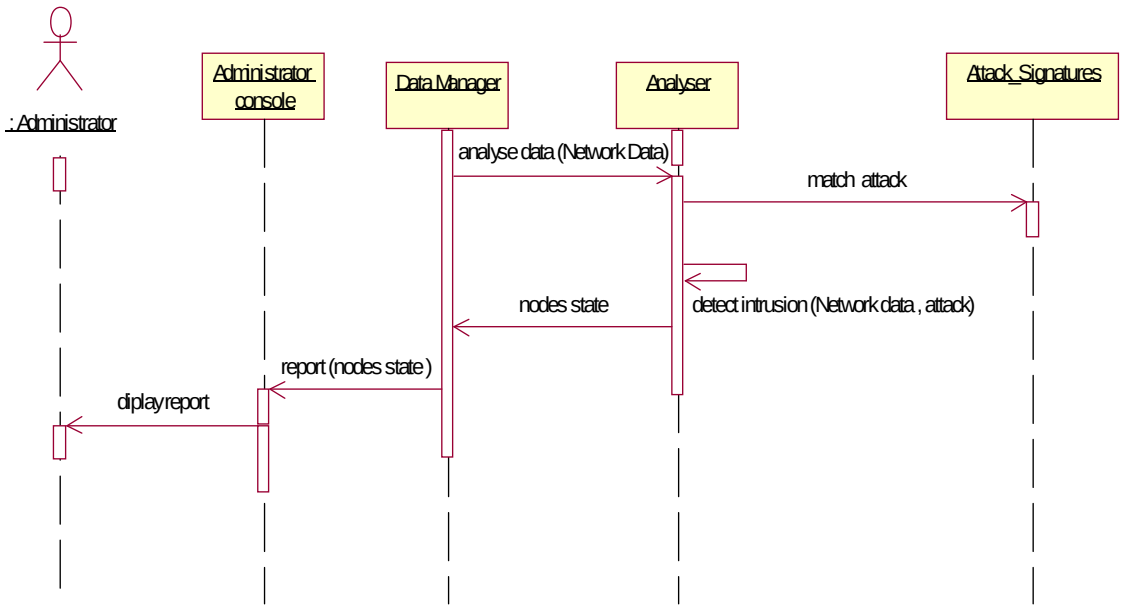


Fig 17 : Sequence Diagram of Data Analysis

Sequence Diagram Description :

Once the data is ready to be analyzed (data already exist on a suitable form), the Data Manager orders the Data Analyzer to process them. Analyzer checks attack into database Attack_Signatures, detects intrusion by opposing the Network Data to attack signatures and then sends nodes state to Data Manager. The latter edits a report about nodes activities and dispatch it to the Administrator Consol.

- **Graph building :**

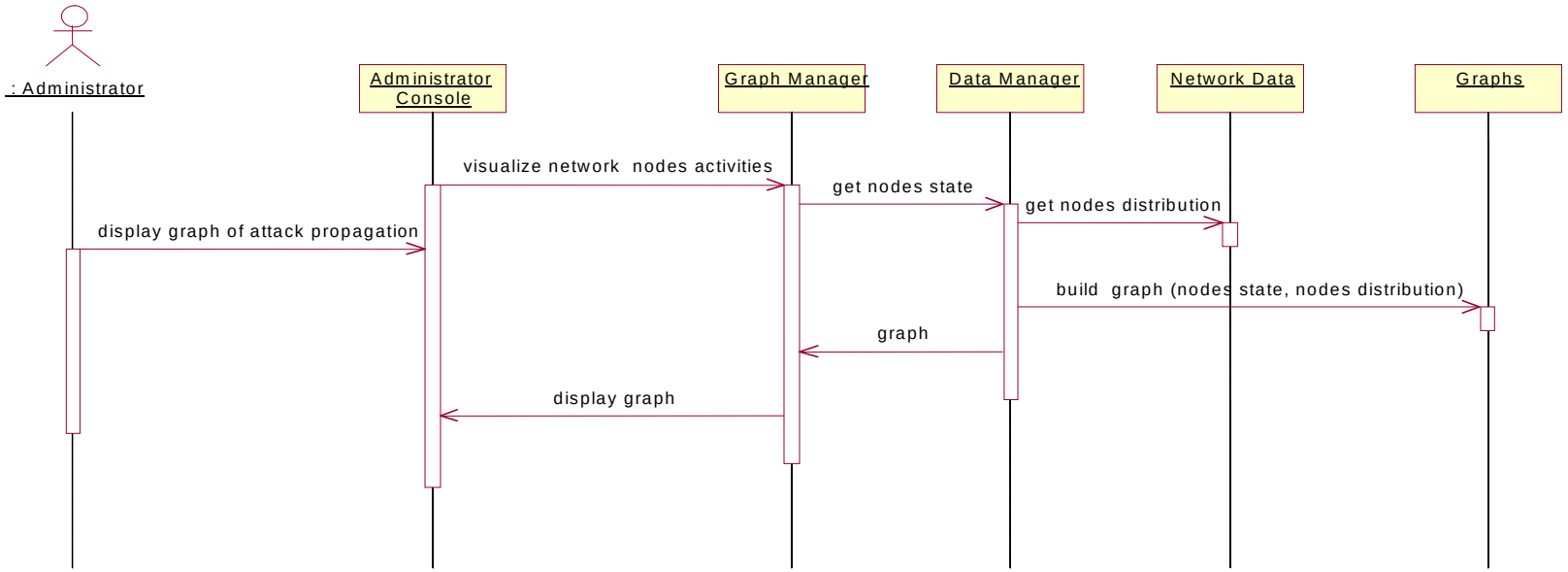


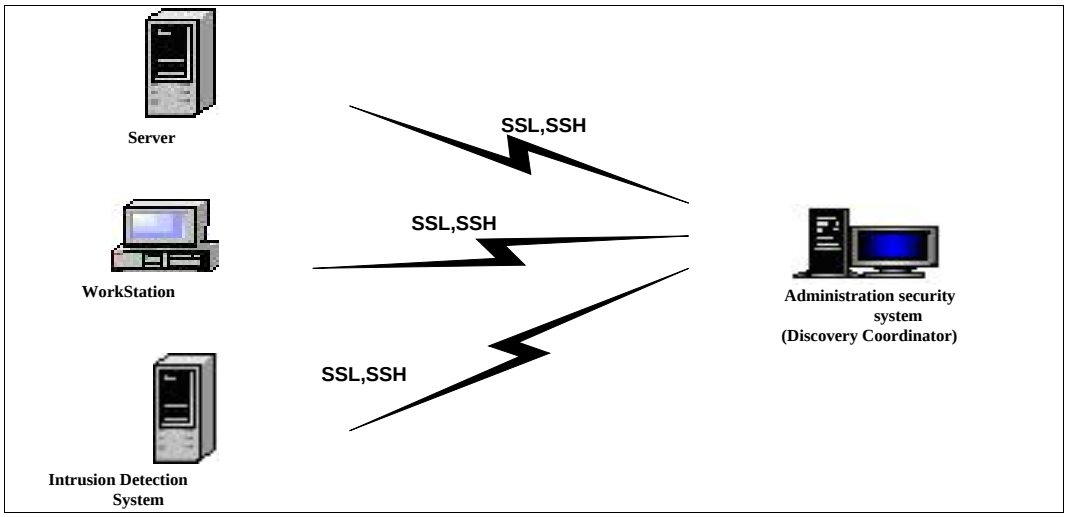
Fig 18 : Sequence Diagram of Graph Building

Description of Collaboration Graph Building :

Administrator requests to build graph. Administrator Console calls the Graph Manager for displaying network nodes activities. Graph Manager gets nodes states from Data Manager and nodes distribution from the network data, then builds graph and at last sends graph to Administrator Consol.

Deployment Diagram

Deployment Diagram indicates physical setting of different system material components and the imputed software. Diagram nodes represent materials components and connections represent communication supports.



SSL,SSH : secured transfer protocol.

Fig 19 : Deployment Diagram of Neural Network IDIP System

Diagram description :

The material system components are :

- Intrusion Detection System (IDS); detects local intrusion attack.
- Workstation: detects suspicious system behavior.

- Server : detects suspicious systems behavior.
- Administration Security System (Discovery Coordinator); detects and reacts to distributed intrusion attacks.

Conclusion:

We have adopted in this chapter the Unified Process to develop a highest-quality system with a minimum of fail and that is easy to maintain.

We intend to finalize different system modules defined into the previous chapter and to degage the domain classes of our system.

The next chapter will deal in detail with system components and how to build them.

Chapter 5 Development of Neural Network IDIP System

I. Introduction:

The dynamic change of network attacks requires a flexible system defense capable to analyze a great quantity of data traffic.

A miseuse detection system based on neural network technology can solve the problems encountered with others protection systems.

Neural network are implemented to identify at any time the nodes activities in the network and to show if their states represent a distributed attack.

So our system is based on the miseuse detection approach and the pattern matching technic to conceive an adaptable detection system in a distributed and heterogeneous network.

In this chapter we will describe the working of the NN_IDIP system to protect network and to preserve its autonomy after attacks.

We will illustrate how Neural Network process input data to produce a graphic representation of different nodes state.

II.Description of the Data Structure:

To build our system we have adapted the following process:

- At first we would construct a data base, which contains the known intrusion, attacks. Each type of attack is stored in a distinguish structure, example we can put the TCP attacks in a TCP_attacks relational table, our data base can include the TCP, UDP ICMP and IP attacks.

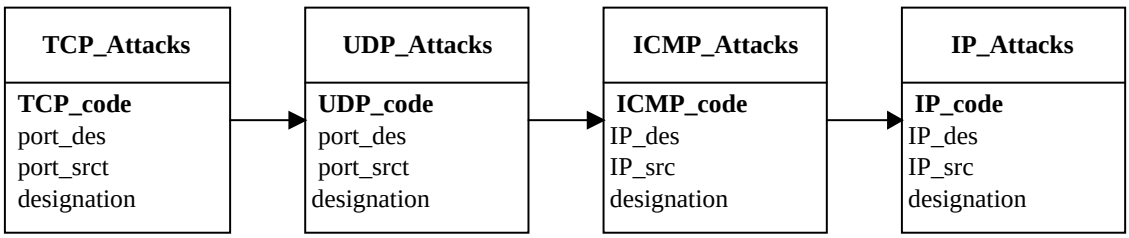


Fig 1: Stored relational attack tables

After the construction of data structure, we can trigger the subsequent correlated process:

- Capture of data process: this process has the role of a sniffer which collects data packets from the network interface. For the security reasons, this interface would be invisible to the users of network , to protect it from misuse.
- Storage of data process: it converts and records the collected data packets into specific data structures.
- Analyze of data process: this process has a delicate task to support the responders engines to make decision. It studies the incoming data to the network to detect the occurrence of distributed attack.
- It monitors activity in the network and compares them against known patterns of intrusive or hostile activities. If the result of the analysis is positives, then data packets are classified as suspect and can be added to the attacks base.
- Graph generation process: it draws a scheme to show the different states of network nodes. Altered nodes are filled with common color to indicate the propagation of a specific attack in the network.

We will now detail the neural network implication in the detection of the intrusion spreading attacks through network.

III.Neural Network Data Process :

Neural network treats incoming data packets from different network devices and try to find the presence of dispersed attacks.

The figure below shows data stream from network components forward neural network, and the output graphic representation of attack assessment generated by the latter.

The order of spreading intrusion through network devices is indicated by the colored and enumerated graphic nodes.

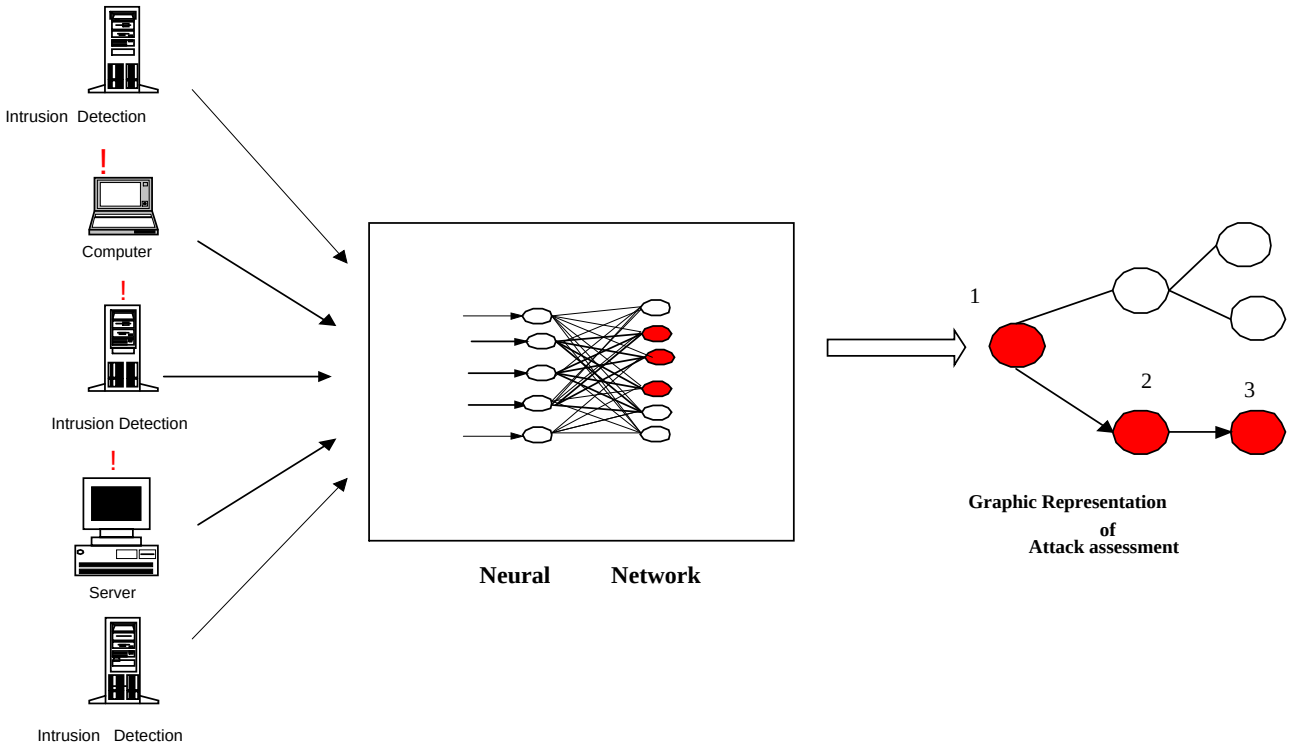


Fig 2 : Neural Network implication in The IDIP

So when an attack affects the network the intrusion detection system build a tree-like graph to show the propagation of the attack among network components.

There exist multiple class of graphs to solve a large type of attacks and each one represent a single kind of attack such as the TCP_Attacks, UDP_Attacks, ICMP_Attacks and IP_Attacks. That is a graph class contains only the diffrents instances of one type of attack, example for the TCP_Attacks we can mention the TCP_Flooding, Remote to Root.

The neural network implemented into intrusion detection system (GrIDS) looks for signatures (the explicit patterns) of known attacks, and any matched activity is considered as an attack. A signature is a distributed event pattern that represents a distributed attack on the instance of system view.

2. Neural Network process steps:

To make decision neural network looks for relationships among system entities and then processes incoming events sent by distributed network components. Events represent

what have happened or are happening in the system, while relationships among system entities represent the system state at certain times. To represent event we have extract the most important features that allow to well identifying an attack. These features are recapitulated into six attributes: SourceIP, Ref_attack, Protocol_ID, VictimIP, VictimPort, Begin_time and End_time.

- Ref_Attack, indicates the reference of the attack. Each attack is identified by an unique clef that distinguish it from the other attacks.
- Protocol_ID, indicates the protocol associated with the event(TCP,UDP,SMTP,..)
- Source_IP, indicates the adress IP of the attacker.
- Victim_IP, indicates the address IP of the victim target.
- Victim_Port, indicates the Port accessed by the attacker.
- Begin_time, indicate the starting_time of the attack.
- End_time, indicates the finishing time of the attack.

The table below shows an example of DoS (Deny of Services), U2R (User to Root) and R2L (Remote to Local) attacks, the SourceIP of attackers, the target VictimIP and the VictimPort of the designated network, and the duration of each event received by the Grids.

Event	Ref_Attack	ProtocolID	SourceIP	VictimIP	VictimPort	Begin_time	End_time
e1	DoS	TCP	128.178.15.8	10.0.0.5	80	18:45:01	19:00:00
e2	DoS	TCP	128.178.15.8	10.0.0.10	23	19:15:05	19:30:10
e3	U2R	UDP	128.178.15.7	10.0.0.71	80	18:59:18	19:16:22
e4	R2L	Unknown	128.178.15.6	10.0.0.80	53	19:45:01	20:00:50
...			…	...	...	...	...

Tab 1 : Historical event received by the Grids

So as mentioned above, when an attack procured in the network, each affected security component (IDS, firewall, router) sends an event, which encapsulates information about

attack, to the discovery coordinator and more precisely at its frontal neural network engine, to make decision about the truth of network intrusion.

The scheme below shows the directed data stream to the neural network, their propagation trough layers and the output graphic revealing the distribution of attack into network.

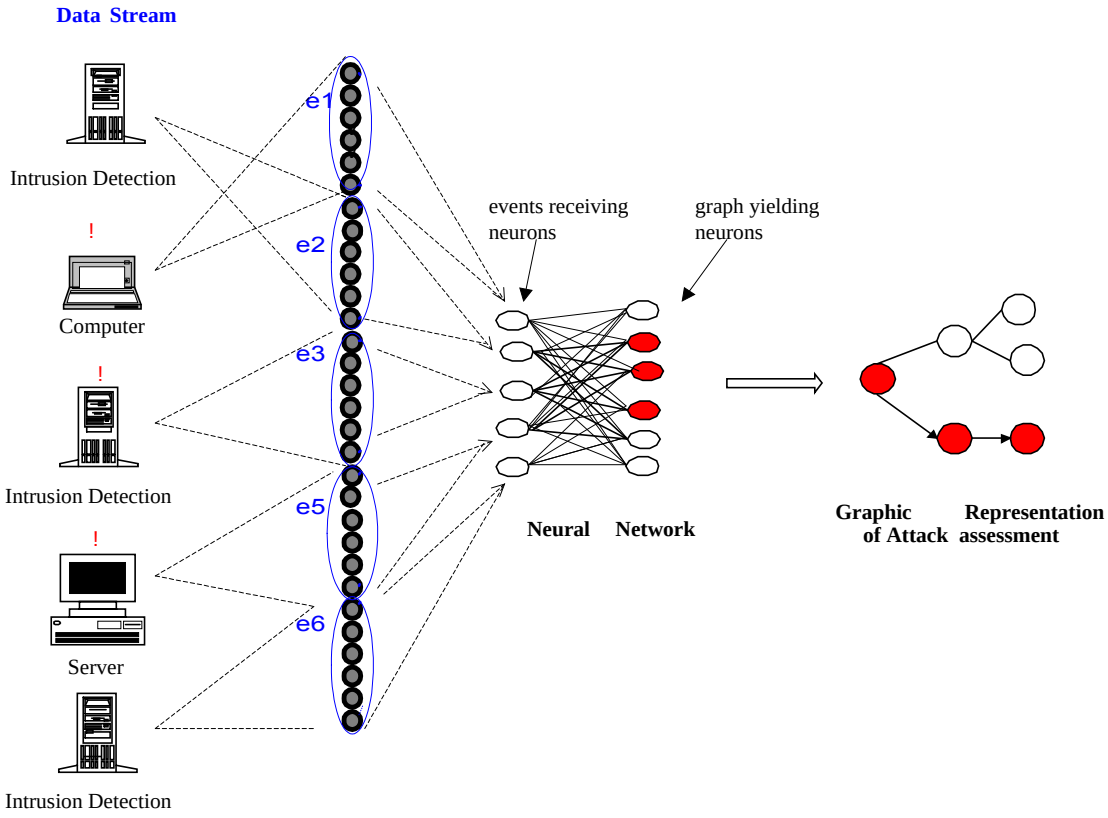


Fig 3 : Working process of Neural Network

3. Neural Network implementation on Graph based Intrusion Detection System:

The core of the Neural Network Graph based Intrusion Detection System (NN_GrIDS) consists of collection of simple processing units or nodes and connection which transform a set of inputs data to a set of searched outputs issues. Subsets of the units are input nodes, output nodes, and nodes between input and output form hidden layers, the connection between two units has some weight, used to determine how much one unit

will affect the other. The weights between neurons are adjusted during the learning process to produce accurate output responses.

The most important property of the NN_ GrIDS is to automatically learn coefficients between neurons connection according to data inputs and data outputs.

To apply the Neural Network (NN) approach, we first have to expose this latter to normal and then to suspected data to automatically adjust its coefficients during the training phase. Performance tests are then conducted with real network traffic and attacks.

4. Neural Network building stages:

To develop a NN_GrIDS ,we have pursued the following steps:

- **Specifying Input and Output Data:**

  In this first stage we choose firstly which variables would be used as input variables and outputs, and we either compute the minimum and maximum value for each variable. Input variables were used by the neural network to make the prediction. Output variables (the dependent variables) contained the results the network was expected to learn in order to rank auditable units.

  The second part of this phase consisted of converting discrete input variable (ProtocolID, Ref_attack) into a standardized numeric representation. The process involved the creation of relational tables for each of the data type and assigning sequential numbers to each unique type of element and then joining these tables to the table that contained the description of historical events records.

  The tables below show the conversion of input discrete variables and the construction of an unique table that describe historic events by digital input variables.

Ref_attack	
discrete form	**numeric form**
DoS	1
U2R	2
R2L	3

ProtocolID	
discrete form	**numeric form**
TCP	1
UDP	2
ICMP	3

SyF	4
NoN	5
..	..

Tab 2 : Ref_attack conversion into numeric fom

SMNTP	4
Unknown	5
..	..

Tab 3: Protocol_ID conversion into numeric form

Event	Ref_attack	ProtocolID	SourceIP	VictimIP	VictimPort	Begin_time	End_time
e1	1	1	128178158	10005	80	184501	190000
e2	1	2	128178158	100010	23	191505	193010
e3	2	2	128178157	100071	80	185918	191622
e4	3	5	128178156	100080	53	194501	200050
...			…	...	...	...	...

Tab 4 : Historical digitized event received by the Neural Network

So the sequence off input events e1, e2, e3,e4 ,.. forwarded to neural network is translated into sequence *111281781581000580184501190000, 121281781581000102319150519301 0 , 221281781571000718018591819162 2, 351281781561000805319450120005 0*. Each neuron in the input layer will receive a specific digitized event sent from one node in the network.

- **Signatures base building:**

Database Signatures includes the set of stored attacks which presents the source from where data set population (results), test and training data were extracted during training phase to construct a concrete system able to process new real enquiries.

A training set is extracted to drive neural network to produce the correct response to an incoming observation.

A test set randomly extracted by the neural net to compute average training error used to determine when to stop training.

Our base contain essentially two relationales tables, the first records events that show some attacks and the second records the description of attacks. Theses tables have on common the clef of attacks attribute.

To create a consistent DataBase Signatures we have proceeded the following algorithm.

Algorithms for building Database Signatures

To build a Database Signatures, we need to extract data from the real network traffic to keep track of attacks . There exist tools that monitor network, collect data from the network traffic and assist to identify the clefs of revealing attacks (example Sniffers).

Also to simulate attacks on the network, we can utilize tools that simulate attacks such as the Security administrator's Tool for Analyzing Network SATAN which is an example of a publicly available sweeping tool that scans for vulnerabilities.

So the two parameters "evens" and "attacks" used in the algorithms below are respectively generated by a monitor network and a simulated attack tool.

Algorithm building _signature_base (events, attacks)

```
{
Debut
fill_attacks(attacks);
fill_events(events);

Fin
}
```

Algorithm fill_attacks(attacks)

```
{
```

```
Begin
Repeat

Att     ←     extract_attack (attacks) ;
if Att ∈ base_attacks
then exit  "success "  ;
else
Base_attacks    ←    Base_attacks ∪  Att ;
end if

End Repeat
End
}
```

Algorithm fill_events(events)

```
{
Debut
Repeat
even   ←     extract_even(events) ;
if even ∈ base_events
then exit   "success "  ;
else
att    ← extract_Ref_attack(even) ;
if att ∈ Base_attacks;
then
 evenc ← convert(even)              /* convert () permit to convert data into suitable
                                       form before being stored in the base_events*/

 base_events  ←    base_events ∪  even ;
else exit  "failure";

end if
```

End repeat
End
}

- **Neural Network Architecture selection:**

Neural network can provide a variety of algorithms and architectures to process data in a definite target. For each class of problem, there exist a specific neural network architecture to resolve it. The number of layers in a net is defined based on the number of interconnected weight in the neuron.
The back propagation architecture is used for training in our project because of its ability to be exposed to a wide variety of input variables and to be generalized to heterogeneous environments.

During training, information is propagated back through the network to update connection weights.

There are different neural network architectures that use different algorithms to calculate the weight changes. Back propagation is a commonly used algorithm in Multi Layers Perceptron.

Back propagation networks mainly work with values between 0 and 1. So we should put numeric input variables into [0 1] intervals. We can also use a neural network simulator (like Neural Planner) whish scales each input variable minimum to 0 and maximum to 1.

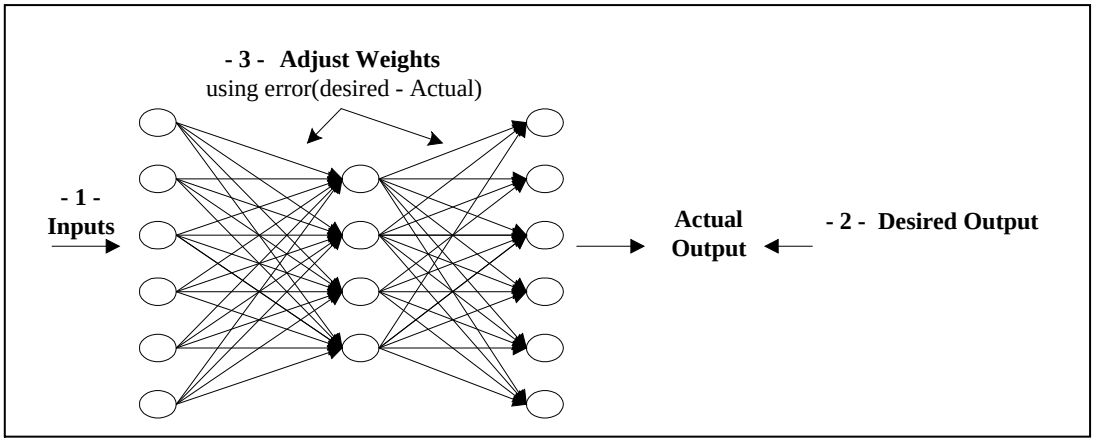


Fig 4: Back propagation Neural Network

- **Neural Network Learning:**

The aim of this stage is to converge neural network to a steady behavior on which we can entrust on its output responses, we have confront output data to estimated those stored on the data base. We have construct a data base called DataBase Attack-Signatures that contains a set of known attack associated with network and user's behavior.

During learning phase, weights are adjusted under the supervised learning algorithm. We expose the desired output variables to the network and we learn it through successive iteration to alter its intern parameters to reach the attended output result.

In each iteration, the output value produced by neural network in response to input observation is compared to the desired output result (already stored in data base) and the difference is used to modify neural network weights in order to correct its behavior to future input observations. We present each observation to neural network, then we propagate these values till output neurons. At this first time, the out prediction obtained is inaccurate, so we compute the error value (difference between expected values and predicted ones), then we back propagate this error through layers and we modify at each level the weights of connection between neurons proportionally to the total error. We repeat this mechanism for each example of data as much as the rate of mistake continually decrease.

The principle of learning algorithm is to go back layer by layer from output neuron to input neurons and to modify synaptic weight so that to minimize the output error. This process is iterative, for each iteration the global error decrease by the descent gradient method.

At the end of learning, knowledge acquired by neural network is represented by the strength and weakness of connection between neurons.

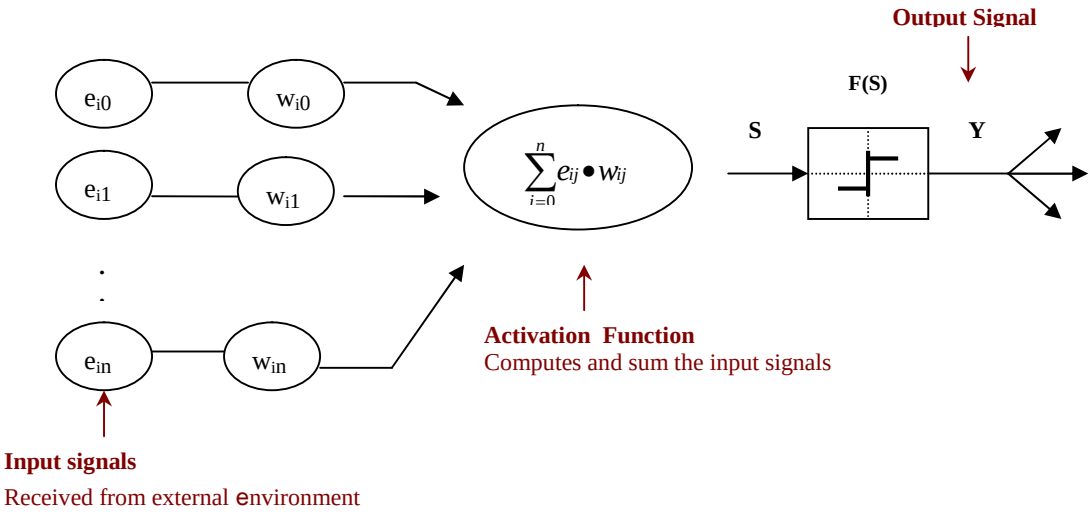


Fig 5: A single Neuron learning

S: Sum of weighted input signals.

F(S): Transfer Function that computes the final output value Y of a neuron. Y : forwards to neurons in next layer until reaching the output layer.

a. **Learning process:**

Assume we have n input units, $X_i,\ldots,X_n$, the first layer plays the role of events receiver, that captures incoming events from network.

Events are propagated to the next layer of the neural network, each hidden unit i receives n input signals $e_{1i},e_{2i},\ldots e_{in}$ and calculates their weighted sum S $= \sum_{i=1}^{n} e_{ij} \cdot w_{ij}$.

The output neuron Y is calculated using the sigmoid transfer function F(S),

F(S) = (1/(1 + exp (-s))).

The computed weight from the training is stored and will become the information or knowledge for the future application.

b. Learning algorithm

Learning in neural networks is done by changing the weighting factors at each element to reduce the global output error. This latter can be computed as the square average of the difference between desired and output values.

Let E denote this error : $E = (D - O)^2$, Or $E = (D - F(\sum_{i=1}^{n} e_{ij} \cdot w_{ij} .))^2$.

D design the desired values and O the computed output values.

E depends from the connection between layers, so to minimize the global error ,we try to reach the lowest point on the curve (E,W).

To find this point , its necessarily to calculate the estimated error relative to each neuron by mean of the gradient of the error function. It can be considered as the contribution of a single neuron to the global error.

At each go back, we calculate the gradient of each neuron from output to input layers of the neural network.

The gradient is function of the derivative of the transfer function F (in our case the latter is the sigmoid function).

F(x) = (1/(1 + exp (-x))).

$$F'(x) = \frac{e^{-x}}{(1/(1+\exp(-x)))} = (F(x)).(1 - F(x)).$$

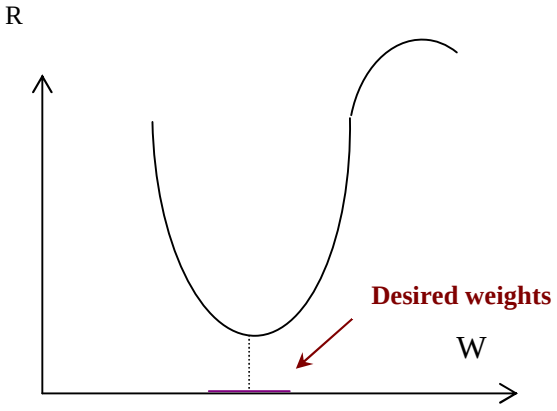
As the global error is function of neural network output, we have to calculate at the first time the partial error at each node start by the output to the input neurons.

Let $¢i$ designs the gradient of the neuron i.

- For the output neuron i: $¢i = (Di - Oi) .\underbrace{Oi . (1 - Oi)}_{F'(Oi)}$

- For the hidden neuron j : $¢_j = Oj .(1 – Oj) . \sum_k W_{jk} \cdot ¢_k$

  K is the index of superior layer.



R: the differences between computed and desired output values.
W: the synaptic weights

Fig 6 :Variation computed output error and synaptic weights during learning phase

We will process an incremental learning algorithm that allow to update synaptic neural network weights on each iteration:

Let:

- j be an index of cases.
- x_j be an input vector for a case j.
- y_j be an output vector for a case j.
- W be a collection of synaptic weights.
- L be the number of network layers.
- N be the number of neurons in each layer.
- F(X,L) output function computed by neural network.
- M(X,W) be the output matrix generated by the network.

- p_j be the predicted output values for a case j.
- $R = p_j - y_j$ be the difference between computed and desired output. values.
- NL be the number of cases in the training (learning) set
- NT be the number of cases in the test set

```
Algorithm learning_neural_network()
{

 Begin
Selecte_training_data_set;
Initialise_weights_coefficients;

for j = 1 to NL
M(X,W,R) = Process_case_ j (xj, pj,); /*this function generates a matrix that contain
                                       the output value at each neuron ,the synaptic
                                       weights between neurons and the errors
                                       between computed output and desired values,
                                       it has as parameters, the input vectors and the
                                       predicted values for each case j.*/
for k = L to 1
Update_weights(M(X,W,R),k) ;

End
}
```

```
Algorithm Update_weights(M(X,W,R),k)
{
 Begin
if (k is last_layer)
{
∂   =R · F'(X);
}
else
```

```
{
∂    = Σ(k+1) ∂·W · F'(X);

}

Δ W = ¦ ·∂· X;


End
}
```

```
Algorithm testing_neural_network()
{
 Begin

for j = 1 to NT
Process_case_ j (xj, pj, M(X,W,R)); /* this function uses the matrix generated
                                       during learning phase*/

for k = L-1 to 1
Update_weights(M(X,W,R),k) ;

End

}
```

During the recall phase of operation the network will respond to inputs that exhibit features similar to those learned during training. Incomplete or noisy inputs may be completely recovered by the network.

Learning improvement:

We can improve the learning performance by the use of algorithms that look for getting the minimum global of the error_function. These algorithms attempt to minimize the total error of the network and to find the corresponded weights connection. Example of algorithms that search the global optimum of a solution is the “Recuit simulé”.

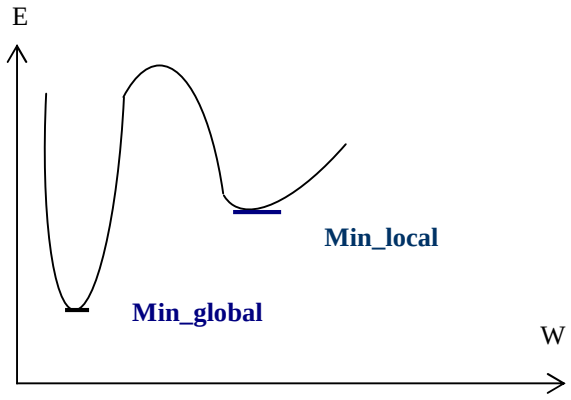


Fig 7 : Variation of the minimum error value

Algorithm Recuit simulé (minimisation)

Begin

t = 0 ; Initialize T=Tmax /* Initial Temperature */ ;

learn neural network;

Vc ← compute_ sum_ square_error (E) /* set of errors*/

Repeat

Repeat

Vn ← Generate_ aleatory_ neighbor_ to _Vc();

If Vc > Vn then Vc = Vn

else if random[0,1] < $e^{\frac{-(Vn-Vc)}{T}}$

```
                then  Vc = Vn
Until (ending_condition)

T=g(T,t) ;
t = t+1 ;                    /* g(T,t) : cooling procedure which attempt to reduce
                                progressively the temperature */
Until (stop criterion )
End.
```

5. Example :

Suppose that the presented attack on the network is Deny of Service whish is referenced by the Ref_Attack DoS, After conversion mechanism the **DoS** reference can be transferred into the binary sequence **111**.

Then the input data to the neural network that contain the sequence 111 shows that the resource component is attacked by the DoS. The nodes of neural network represent network hosts and the edges represent the traffic among them. Activated node indicate that the correspond network host is affected by this attack.

Onward input data would be called events which represent an abstraction of the network occurrence. An input event ei to the neuron i may consist of a single data network packet or a collection packets (for more information about network protocols and traffic you can see the annex).

We will illustrate how neural network handles incoming data in the following schema.

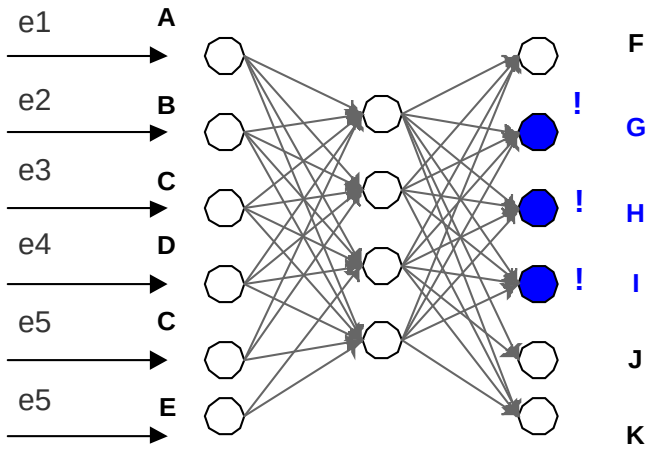


Fig 8 : Neural Network processing information

The input nodes {A;B;C;D;E} receive events {e1;e2;e3;e4;e5} from correspondent network hosts, the output nodes {F;G;H;I;J;K} show the states of these hosts and between them there exist a hidden layer that compute the incoming data to perform the output result.

The nodes {F;G;H;I;J;K}can be displayed into graph that present the relationship between them and the order of the intrusion spread.

Weighted connections between nodes are updated during learning phase to build a generic structure capable to react to coming events.

Patterns to be learned is:

INPUT					**OUTPUT**				
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**K**
110**111**010	001110101	101**111**011	100001110	010**111**010	**1**	0	**1**	0	**1**
110011110	001**111**111	101101011	100111010	010**111**010	0	**1**	0	0	**1**
110**111**010	001011011	101**111**011	100**111**011	000110010	**1**	0	**1**	**1**	0
110100100	001**111**111	111101010	101**111**110	000011000	0	**1**	0	**1**	0
110**111**110	001011111	011010110	100011000	010**111**010	**1**	0	0	0	**1**
110110010	001**111**011	101111011	1010**111**01	000101010	0	**1**	**1**	0	0

110**111**110	001101011	111110010	1010**11**01	010**111**010	**1**	0	0	**1**	**1**
110**111**010	001**111**011	101**111**011	101011101	010110110	**1**	**1**	**1**	0	0

Conclusion:

Across different rubrics of this chapter, we have explained the steps of building the Neural Network IDIP system and its function mode to protect system from distributed intrusion attacks.

Conclusion

To reach a best level of system security we have attempted through this memory to get profit from the advantages of the two high level technology: the Neural Network technology whish has the capability to handle missing and noisy data in an heterogeneous environment without a complex programming fashion, and the Distributed Intrusion Detection System that aims to mitigate the detection of network propagated intrusion attack with a great level of confidence.

The combinations of these technologies induce to construct a strong protected operating system with a centralized monitoring site: the discovery coordinator. Our approach is be implemented in this latter.

Annex : Internet Protocols

I. Introduction:

The Internet protocols consist of a suite of communication protocols, of which the two best known are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The Internet protocol suite not only includes lower-layer protocols (such as TCP and IP), but it also specifies common applications such as electronic mail, terminal emulation, and file transfer. This chapter provides specifications about the Internet protocols.

II. Background:

Internet protocols were first developed in the mid-1970s, when the Defense Advanced Research Projects Agency (DARPA) became interested in establishing a packet-switched network to facilitate communication between dissimilar computer systems at research institutions. TCP/IP later was included with Berkeley Software Distribution (BSD) UNIX and has since become the foundation on which the Internet and the World Wide Web (WWW) are based.

Documentation of the Internet protocols (including new or revised protocols) and policies are specified in technical reports called Request For Comments (RFCs), which are published and then reviewed and analyzed by the Internet community. Protocol refinements are published in the new RFCs. [19]

To illustrate the scope of the Internet protocols, fig 1 maps many of the protocols of the Internet protocol suite and their corresponding OSI layers.

This chapter addresses the basic elements and operations of these and other key Internet protocols.

<table>
<tr><th>OSI Reference Model</th><th colspan="2">Internet Protocol Suite</th></tr>
<tr><td>Presentation</td><td rowspan="2">SMTP,SNMP</td><td>XOR</td></tr>
<tr><td>Session</td><td>RPC</td></tr>
<tr><td>Transport</td><td colspan="2">TCP, UDP</td></tr>
<tr><td>Network</td><td>IP</td><td>ICMP</td></tr>
<tr><td>Link</td><td colspan="2">ARP</td></tr>
<tr><td>Physical</td><td colspan="2">Non Spécified</td></tr>
</table>

fig 1 : Internet protocols and OSI model layers

III. Internet Protocol (IP)

The Internet Protocol (IP) belongs to the Layer 3 of OSI model that contains addressing information and some control information that enables packets to be routed. IP is documented in RFC 791 and is the primary network-layer protocol in the Internet protocol suite.

IP has two primary responsibilities: providing connectionless, best-effort delivery of datagrams through an internetwork; and providing fragmentation and reassembly of datagrams to support data links with different maximum-transmission unit (MTU) sizes.[20]

The protocol IP determines the receiver of the message by means of 3 fields:

- The Address field: present the address of the device
- The subnet Mask : permit to protocol IP to determine the field of local network
- The default gateway field: permit to protocol IP to know the destination machine if the target is out of the local network.

An IP packet contains several types of information, as illustrated in fig 2

Version	IHL	Type-of-service	Total length	
Identification			Flags	Fragment offset
Time-to-live		Protocol		Header checksum
Source address				
Destination address				
Options (+ padding)				
Data (variable)				

fig 2: fields of IP packet

The following discussion describes the IP packet fields illustrated in the figure above.

- **Version:** Indicates the version of IP currently used.
- **IP Header Length *(*IHL):** Indicates the datagram header length in 32-bit words.
- **Type-of-Service**: Specifies how an upper-layer protocol would like a current datagram to be handled, and assigns datagrams various levels of importance.
- **Total Length**: Specifies the length, in bytes, of the entire IP packet, including the data and header.
- **Identification**: Contains an integer that identifies the current datagram. This field is used to help join together datagram fragments.
- **Flags**: Consists of a 3-bit field to control fragmentation.
- **Fragment Offset**: Indicates the position of the fragment's data relative to the beginning of the data in the original datagram, which allows the destination IP process to properly reconstruct the original datagram.
- **Time-to-Live**: Maintains a counter that gradually decrements down to zero, at which point the datagram is discarded. This keeps packets from looping endlessly.
- **Protocol**: Indicates which upper-layer protocol receives incoming packets after IP processing is complete.
    - ICMP: 1
    - IGMP: 2
    - TCP: 6

- UDP: 17

- **Header Checksum**: Helps ensure IP header integrity.
- **Source Address***:* Specifies the sending node.
- **Destination Address**: Specifies the receiving node.
- **Options**: Allows IP to support various options, such as security.
- **Data**: Contains upper-layer information.

1. IP Address Format

Each host on a TCP/IP network is assigned a unique 32-bit logical address that is divided into two main parts: the network number and the host number. The network number identifies a network and must be assigned by the Internet Network Information Center (InterNIC).

The host number identifies a host on a network and is assigned by the local network administrator. The figure below illustrates the basic format of an IP address.

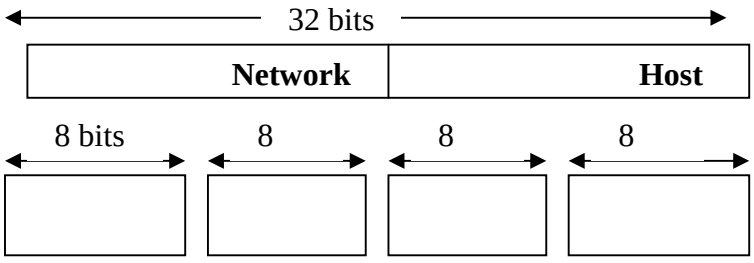


Fig 2 : Decomposition of IP address

2. IP Datagram fragmentation

The maximum size of a datagram is 65535 byte, but this value is never reached because network haven't enough capability to send so big data packets.

Also the maximum size of a datagram varies according to the network type. The maximum size of a packet is called MTU (Maximum Transfer Unit).

The table below shows some network types and their correspondent MTU.

Network type	**MTU (byte)**
Arpanet	1000
Ethernet	1500
FDDI	4470

Table 1: Network type and correspondent MTU

The fragmentation of a datagram occurs at level of router during the transition of data packets from a network with a greater MTU; if the datagram is too large to pass along the network , the router divides them into fragments with fixed size lower then the MTU of the network ,this size may be a multiple of 8 bytes.

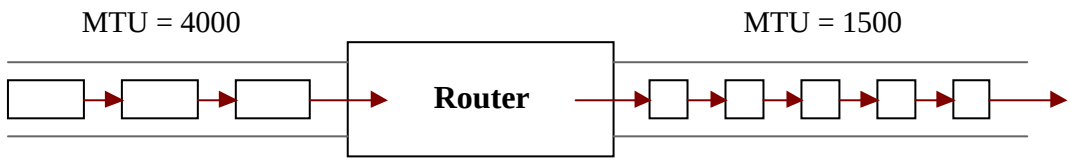


fig 3: Router fragmentation of data packets

The router will then send these fragments in an encapsulated form; it adds a header to each fragment and others information that allow the gathering of fragments in a right order by the destination device.

To assemble again dispersed fragments, each datagram processes the following field:

- Data Offset
- Identification field
- Total length
- Flag: this field is composed of three bits:
    - The first bit is not used
    - The second (called DF) indicates whether the datagram can be divided. If a datagram has this bit equals to one and that the router can't dispatch without dividing, then the datagram is rejected with an error message.
    - The last bit (called MF: More Fragments) indicates whether the datagram is data fragment. If the index is equal to zero value, it shows that the

fragment is the **last (so the router may possess the previous fragment) or the datagram is not** fragmented.

IV. Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol (ICMP) is a network-layer Internet protocol that provides message packets to report errors and other information regarding IP packet processing back to the source.

1. ICMP Messages

ICMPs generate several kinds of useful messages, including Destination Unreachable, Echo Request and Reply, Redirect, Time Exceeded, and Router Advertisement and Router Solicitation. If an ICMP message cannot be delivered, no second one is generated. This is to avoid an endless flood of ICMP messages.[21]

A ICMP Message can be represented as follow:

Header	ICMP Message			
	Type(8 bite)	Code(8 bits)	Checksum(16 bite)	Message (variable size)

Fig 8 : ICMP encapsulated into Datagram IP

V. Transmission Control Protocol (TCP)

The TCP provides reliable transmission of data in an IP environment. TCP corresponds to the transport layer (Layer 4) of the OSI reference model. Among the services TCP provides are stream data transfer, reliability, efficient flow control, full-duplex operation, and multiplexing.[22]

With stream data transfer**,** TCP delivers an unstructured stream of bytes identified by sequence numbers. This service benefits applications because they do not have to chop data into blocks before handing it off to TCP. Instead, TCP groups bytes into segments and passes them to IP for delivery.

TCP offers reliability by providing connection-oriented, end-to-end reliable packet delivery through an internetwork. It does this by sequencing bytes with a forwarding acknowledgment number that indicates to the destination the next byte the source

expects to receive. Bytes not acknowledged within a specified time period are retransmitted. The reliability mechanism of TCP allows devices to deal with lost, delayed, duplicate, or misread packets. A time-out mechanism allows devices to detect lost packets and request retransmission.

TCP offers efficient flow control, which means that, when sending acknowledgments back to the source, the receiving TCP process indicates the highest sequence number it can receive without overflowing its internal buffers.

Full-duplex operation means that TCP processes can both send and receive at the same time.

Finally, TCP's multiplexing means that numerous simultaneous upper-layer conversations can be multiplexed over a single connection.

1. TCP Connection Establishment

To use reliable transport services, TCP hosts must establish a connection-oriented session with one another. Connection establishment is performed by using a "three-way handshake" mechanism.

A three-way handshake synchronizes both ends of a connection by allowing both sides to agree upon initial sequence numbers. This mechanism also guarantees that both sides are ready to transmit data and know that the other side is ready to transmit as well. This is necessary so that packets are not transmitted or retransmitted during session establishment or after session termination.

Each host randomly chooses a sequence number used to track bytes within the stream it is sending and receiving. Then, the three-way handshake proceeds in the following manner:

The first host (Host A) initiates a connection by sending a packet with the initial sequence number (X) and SYN bit set to indicate a connection request. The second host (Host B) receives the SYN, records the sequence number X, and replies by acknowledging the SYN (with an ACK = X + 1). Host B includes its own initial sequence number (SEQ = Y). An ACK = 20 means the host has received bytes 0 through 19 and expects byte 20 next. This technique is called forward acknowledgment. Host A then acknowledges all bytes Host B sent with a forward acknowledgment indicating the next byte Host A expects to receive (ACK = Y + 1). Data transfer then can begin.

2. TCP Packet Format

fig4 illustrates the fields and overall format of a TCP packet.

Source port			Destination port
Sequence number			
Acknowledgement number			
Data offset	Reserved	Flags	Windows
Checksum			Urgent pointer
Options (+ padding)			
Data (variable)			

fig 4: fields of TCP packet

3. TCP Packet Field Descriptions

The following descriptions summarize the TCP packet fields illustrated in the figure above

Source Port *and* Destination Port : Identifies points at which upper-layer source and destination processes receive TCP services.

Sequence Number: Usually specifies the number assigned to the first byte of data in the current message. In the connection-establishment phase, this field also can be used to identify an initial sequence number to be used in an upcoming transmission.

Acknowledgment Number: Contains the sequence number of the next byte of data the sender of the packet expects to receive.

Data Offset :Indicates the number of 32-bit words in the TCP header.

Reserved :Remains reserved for future use.

Flags: Carries a variety of control information, including the SYN and ACK bits used for connection establishment, and the FIN bit used for connection termination.

Window: Specifies the size of the sender's receive window (that is, the buffer space available for incoming data).

Checksum :Indicates whether the header was damaged in transit.

Urgent Pointer : Points to the first urgent data byte in the packet.

Options : Specifies various TCP options.

Data : Contains upper-layer information.

VI. User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is a connectionless transport-layer protocol (Layer 4) that belongs to the Internet protocol family. UDP is basically an interface between IP and upper-layer processes. UDP protocol ports distinguish multiple applications running on a single device from one another.

Unlike the TCP, UDP adds no reliability, flow-control, or error-recovery functions to IP. Because of UDP's simplicity, UDP headers contain fewer bytes and consume less network overhead than TCP.

UDP is useful in situations where the reliability mechanisms of TCP are not necessary, such as in cases where a higher-layer protocol might provide error and flow control.

UDP is the transport protocol for several well-known application-layer protocols, including Network File System (NFS), Simple Network Management Protocol (SNMP), Domain Name System (DNS), and Trivial File Transfer Protocol (TFTP).

The UDP packet format contains four fields, as shown in fig 5. These include source and destination ports, length, and checksum fields.[23]

Source Port	Destination Port
Length	Cheksum

Figure 5: A UDP packet consists of four fields

Source and destination ports contain the 16-bit UDP protocol port numbers used to demultiplex datagrams for receiving application-layer processes. A length field **s**pecifies the length of the UDP header and data. Checksum provides an (optional) integrity check on the UDP header and data.

VII. Internet Protocols Application-Layer Protocols

The Internet protocol suite includes many application-layer protocols that represent a wide variety of applications, including the following:

- File Transfer Protocol (FTP)—Moves files between devices
- Simple Network-Management Protocol (SNMP)—Primarily reports anomalous network conditions and sets network threshold values
- Telnet—Serves as a terminal emulation protocol
- X Windows—Serves as a distributed windowing and graphics system used for communication between X terminals and UNIX workstations
- Network File System (NFS), External Data Representation (XDR), and Remote Procedure Call (RPC)—Work together to enable transparent access to remote network resources
- Simple Mail Transfer Protocol (SMTP)—Provides electronic mail services
- Domain Name System (DNS)—Translates the names of network nodes into network addresses

table 2 lists these higher-layer protocols and the applications that they support.

Application	**Protocols**
File transfer	FTP
Terminal emulation	Telnet
Electronic mail	SMTP
Network management	SNMP
Distributed file services	NFS, XDR, RPC, X Windows

Table 2: Higher-Layer Protocols and Their Applications

VIII. Example of attacks :

- TCP SYN Flooding and IP Spoofing Attacks: causes a deny of services, so that target machine can't accept another connections reminds out of services.

- User Datagram Protocol (UDP) diagnostic port attack: A sender transmits a volume of requests for UDP diagnostic services on the router, which causes all CPU resources to be consumed servicing the phony requests.
- Scanners: exploits network hosts and tries to break in.
- Password sniffing : cracks password of legitimate user for ulterior use.
- User to Root(U2R) and Remote to Local(R2L) attacks; intruder attempts to acquire privilege and to access system root.

Example of attack rules :

attack-responses.rules:

alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any (msg:"ATTACK RESPONSES http dir listing"; content: "Volume Serial Number"; flags:A+; classtype:bad-unknown; sid:1292; rev:4;)

alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any (msg:"ATTACK RESPONSES command completed"; content:"Command completed"; nocase; flags:A+; classtype:bad-unknown; sid:494; rev:5;)

alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any (msg:"ATTACK RESPONSES command error"; content:"Bad command or filename"; nocase; flags:A+; classtype:bad-unknown; sid:495; rev:5;)

backdoor.rules:

alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR subseven 22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|"; reference:arachnids,485; reference:url,www.hackfix.org/subseven/; sid:103; classtype:misc-activity; rev:4;)

alert udp $EXTERNAL_NET 60000 -> $HOME_NET 2140 (msg:"BACKDOOR DeepThroat 3.1 System Info Client Request"; content:"13"; reference:arachnids,106; sid:122; classtype:misc-activity; rev:3;)

alert icmp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR SIGNATURE - Q ICMP"; itype: 0; dsize: >1; reference:arachnids,202; sid:183; classtype:misc-activity; rev:3;)

ftp.rules:

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP EXPLOIT stat overflow"; flags:A+; dsize:>1000; content:"stat "; nocase;

reference:url,labs.defcom.com/adv/2001/def-2001-31.txt; classtype:attempted-admin; sid:1379; rev:3;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP EXPLOIT STAT * dos attempt"; flags:A+; content:"STAT "; nocase; content:"*"; reference:bugtraq,4482; classtype:attempted-dos; sid:1777; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP EXPLOIT STAT ? dos attempt"; flags:A+; content:"STAT "; nocase; content:"?"; reference:bugtraq,4482; classtype:attempted-dos; sid:1778; rev:1;)

icmp.rules:

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP ISS Pinger"; content:"|495353504e475251|";itype:8;depth:32; reference:arachnids,158; classtype:attempted-recon; sid:465; rev:1;)

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP L3retriever Ping"; content: "ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"; itype: 8; icode: 0; depth: 32; reference:arachnids,311; classtype:attempted-recon; sid:466; rev:1;)

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Nemesis v1.1 Echo"; dsize: 20; itype: 8; icmp_id: 0; icmp_seq: 0; content: "|00000000000000000000000000000000000000000|"; reference:arachnids,449; classtype:attempted-recon; sid:467; rev:1;)

scan.rules:

alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"SCAN cybercop os probe"; flags: SF12; dsize: 0; reference:arachnids,146; classtype:attempted-recon; sid:619; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy attempt"; flags:S; classtype:attempted-recon; sid:618; rev:2;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS Proxy attempt"; flags:S; reference:url,help.undernet.org/proxyscan/; classtype:attempted-recon; sid:615; rev:3;)

telnet.rules

alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"TELNET solaris memory mismanagement exploit attempt"; flags:A+; content:"|A0 23 A0 10 AE 23 80 10 EE 23 BF EC 82 05 E0 D6 90 25 E0|"; classtype:shellcode-detect; sid:1430; rev:4;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"TELNET SGI telnetd format bug"; flags:A+; content: "_RLD"; content: "bin/sh"; reference:arachnids,304; classtype:attempted-admin; sid:711; rev:4;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"TELNET ld_library_path"; flags:A+; content:"ld_library_path"; reference:cve,CVE-1999-0073; reference:arachnids,367; classtype:attempted-admin; sid:712; rev:4;)

Glossaries

A

Anomaly Detection : One of the two methodologies of intrusion detection. Anomaly detection is based on the normal behavior of a subject (e.g., a user or a system); any action that significantly deviates from the normal behavior is considered intrusive. The other methodology is misuse detection.

Application : A particular kind of work that a user performs on a computer; for example, a payroll application, an airline reservation system. Abridged term for application software.

application layer : The layer that provides means for the application processes to *access* the OSI environment. This layer provides a means for the application processes to exchange information and it contains the application-oriented protocols by which these processes communicate

artificial neuron : A primitive processing element in a artificial neural network, with several inputs and one output, the output value of which is a non-linear function of a linear combination of the input values with adjustable weighting coefficients. Artificial neurons are modelled according to the functioning of neurons in the nervous system and are interconnected in order to exchange messages. Each artificial neuron is a node of the artificial neural network, that cooperates and communicates with other neurons. A artificial neural network can also have input nodes which are not artificial neurons.

Audit Trail : Records a chronology of system resource usage. This includes user log-in, file access, other various activities, and whether any actual or attempted security violations occurred.

Availability : computing system provide proper service to authorized users when expected.

C

Confidentiality : limit access to computing systems to only authorized users (could be a person, process, or system)

F

False Negative : An actual intrusive action that the system allows to pass as no intrusive behavior.

False Positive : Classification of an action as anomalous (a possible intrusion) when it is legitimate.

I

IDS : Intrusion detection system.

Integrity : limit the possibility to alter or destroy computing resources (both maliciously and accidentally).

Intrusion : Any activity that violates the security policy of an information system.

Intrusion Detection : The process of identifying intrusions by observing security logs, audit data, or other information available in computer systems and/or networks.

IP spoofing : where a host from the void sends out packets which claim to come from an inside trusted host

L

Layer : In distributed data processing, a group of capabilities, functions, and protocols considered as a whole, that belongs to a given level in a hierarchical arrangement, of such as features of a given network architecture, and that extends across various data processing systems.

Learning : The process by which an artificial neural network improves its performance by adjustment of its parameters in response to a succession of input patterns. In general, learning consists in connection weight adjustments.

learning algorithm : In machine learning, an algorithm that adjusts a set of parameters or a network structure in response to learning. Such algorithms include genetic algorithms, back propagation in artificial neural networks, and credit/blame assignment in the learning-apprentice strategy.

learning rate : In artificial neural networks, a parameter that regulates the magnitude of changes to connection weights during learning. The amount of change of a connection weight is the product of a value given by the learning algorithm and a coefficient which is the learning rate.

M

Machine learning : The process by which a functional unit improves its capability or performance by acquiring new knowledge or skills, or by reorganizing existing knowledge or skills.

Misuse Detection : One of the two methodologies of intrusion detection. Catches intrusions in terms of the characteristics of known patterns of attacks or system vulnerabilities; any action that conforms to the pattern of a known attack or vulnerability is considered intrusive. The other methodology is anomaly detection.

Misuse Signature : A known pattern or attack or vulnerability, usually specified in a certain attack specification language.

N

Network : An arrangement of entities and their interconnections. In network topology or in an abstract arrangement, the interconnected entities are points on a scheme, and the interconnections are lines on the scheme. In a computer network, the interconnected entities are computers *or* data communication equipment, and the interconnections are data links.

neural connection : In artificial neural networks, a link between two artificial neurons that is defined by the source neuron, the destination neuron, and a connection weight. Synonymous with neural link.

O

Object : In computer security, an entity to which access is controlled ; for example, a file, a program, an area of main storage ; data collected and maintained about a person. In programming languages, a set of operations and data that store and retain the effect of the operations. In artificial intelligence, a physical or conceptual entity that may have one or more attributes.

P

Packet : In data communications, a sequence of bits arranged in a specific format, containing control data and possibly user data, and that is transmitted and switched as a whole.

packet sniffer : A functional unit that monitors the network traffic to recognize and decode packets of interest. A packet sniffer is used for network management.

Pattern : In artificial intelligence, a set of features and their relationships used to recognize an entity within a given context. These features could include a geometrical shape, a sound, a picture, a signal, or written text.

Penetration : A successful attack, the ability to obtain unauthorized (undetected) access to files and programs or the control state of a computer system.

Perceptron : An artificial neural network consisting of one artificial neuron, with a binary output which is determined by applying a monotonic function to a linear combination of the input values and with error-correction learning.

Port : To make the programming changes necessary to allow a program that runs on one type of computer to run on another type of computer. A termination point through which signals can enter or leave a network.. In a network, a functional unit through which data can enter or leave a network.

Profile : A set of parameters used to capture the pattern of a subject's (a user or a program) normal behavior. It is normally used to conduct anomaly detection.

Protocol : A set of rules that determines the behavior of functional units in achieving communication. In programming languages, the set of rules that determines the behavior of objects in the exchange of messages. In *OSI*, a set of semantic and syntactic rules that determine the behavior of entities in the same layer in performing communication functions.

protocol data unit (PDU) : In OSI, a block of data specified in a protocol of a given layer and consisting of protocol control information of that layer, and possibly user data of that layer.

R

Risk : Accidental or unpredictable exposure of information, or violation of operations integrity due to the malfunction of hardware or incomplete or incorrect software design.

S

Scan : to examine an object or synthesize an image according to a predetermined sequence.

Scanner : a vision sensor that scans its environment in a systematic pattern; for example: a device that optically scans printed or written data and generates their digital representations.

Schema : In artificial intelligence, a formalism for representing information about a simple concept, an entity, or a class of objects by means of its possible uses. The schema shows ways of using a concept. It does not describe typical instances of that concept.

Security Policy : A set of rules and procedures regulating the use of information, including its

processing, storage, distribution, and presentation.

System : A set of elements and relations among them considered as a whole and for which there is a recognized purpose and capability. Such elements may be both material objects and modes of thinking as well as the results thereof

Source Routing : where a intruder mimics an IP packet comes from a trusted system

T

TCP : Transmission Control Protocol is a connection-oriented protocol designed to provide a communications protocol used in the Internet. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. TCP corresponds approximately to transport layer protocols within the OSI reference model. TCP is the abbreviation for.

TCP/IP : Two interrelated protocols that are part of the Internet Protocol suite. TCP operates in the transport layer and breaks data into packets and uses IP to route them between and into networks to produce an end-to-end capability. TCP/IP is the abbreviation for Transmission Control Protocol/Internet Protocol.

V

Vector : A quantity usually characterized by an ordered set of scalars.

Vulnerability : A known or suspected flaw in the hardware or software or operation of a system that exposes the system to penetration or its information to accidental disclosure.