REAL TIME HYBRID INTRUSION DETECTION SYSTEM USING APACHE STORM

By

SESHA SAI GOUTAM MYLAVARAPU

Bachelor of Technology in Computer Science and Engineering

Jawaharlal Nehru Technological University

Hyderabad, India

2012

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
MAY, 2015

REAL TIME HYBRID INTRUSION DETECTION SYSTEM USING APACHE STORM

Thesis Approved:

Dr. Johnson Thomas

Thesis Adviser

Dr. Christopher Crick

Dr. David Cline

Name: SESHA SAI GOUTAM MYLAVARAPU

Date of Degree: MAY, 2015

Title of Study: REAL TIME HYBRID INTRUSION DETECTION SYSTEM USING APACHE STORM

Major Field: COMPUTER SCIENCE

Networks are prone to intrusions and detecting intruders on the internet is a major problem. Many Intrusion Detection Systems have been proposed to detect these intrusions. However, as the internet grows day by day, there is a huge amount of data (big data) that needs to be processed to detect intruders. For this reason, intrusion detection has to be done in real-time before intruders can inflict damage, and previous detection systems do not satisfy this need for big data.

Using Apache Storm, a Real time Hybrid Intrusion Detection System has been developed in our thesis. Apache Storm serves as a distributed, fault tolerant, real time big data stream processor. The hybrid detection system consists of two neural networks. The CC4 instantaneous neural network acts as an anomaly-based detection for unknown attacks and the Multi Layer Perceptron neural network acts as a misuse-based detection for known attacks. Based on the outputs from these two neural networks, the incoming data will be classified as ”attack” or ”normal.” We found the average accuracy of hybrid detection system is 89% with a 4.32% false positive rate. This model is appropriate for real time detection since Apache Storm acts as a real time streaming processor, which can also handle big data.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Intruder attacks can be detected and restricted using several mechanisms like Cryptography, Authentication, etc., and the Intrusion Detection System is one of these. An intrusion detection system is an attempt to detect the intruders in a computer system or a network. As the use of sophisticated computing tools in almost every aspects of life increases, risks and malicious intrusions are also increasing. It is, therefore, very important to design security mechanisms to prevent malicious attacks on system resourses and data. We view network traffic data as big data that has the properties of high volume, high velocity and high variety. Therefore, network intrusion detection is a real-time big data problem.

1.1 Computer Security and Intrusion Detection

Computer security is defined as technological and managerial procedures applied to computer systems to ensure the availability, integrity and confidentiality of information managed by the computer system. It can be divided into three areas: prevention, detection and reaction[21]. Intrusion detection falls into the second catogery. As Edward[3] states, "Intrusion Detection is the process of identifying and responding to malicious activity targeted at computing and networking resources." Techniques like Authentication are normally adopted to prevent the computing system from unauthorized and malicious attacks. If, somehow, they fail, the intrusion detection system acts as a next layer of security for the system by attempting to detect those attacks. There are two types of intrusion detection systems: signature based and heuristic or anomaly based. In a signature based system, intruders are detected from the previously known attacks, but, in anomaly based systems,

intruders are detected from the unusual behavior of the user. Intrusion detection systems can also be Network based and Host based, depending on whether the intruders are being detected in a network or in an individual computer system.

1.2 Artificial Neural Networks

An artificial neural network is an information processing paradigm that is inspired by the biological nervous systems, such as the way the brain also processes information. It tries to represent the physical brain and the thinking process through electronic circuits and software. An Artificial neural network is a network of individual neurons. Each neuron in a neural network acts as an independent processing element. Like the brain, neural networks learn by example or training. They can be configured for any specific application through a learning process.

There are two different learning methods for the neural networks: supervised and unsupervised. In the supervised learning method, a neural network learns from the presence of both input and desired output data. On the other hand, in unsupervised learning, a network learns only from the presence of input data. The field of artificial neural networks can also be recognized using many names, such as connectionism, neuro-computing, natural intelligence systems, parallel distributed processing etc.

1.2.1 Neural Networks in General

Researchers, for the last couple of decades, have attempted to invent a system that can learn like people. An Artificial neural network first requires training, which can be performed through some learning process, and then it works independently.

A number of artificial neurons are connected together to form the artificial neural network. Neurons can be considered as small processing units that are the basic elements of the neural network. Neurons receive information on their incoming connection and transport the

processed information to other neurons through their outgoing connections. These connections are called ”weights”. The numerical information in the artificial neural networks is simulated with specific values stored in those weights. The connection structure of the networks can be changed by simply changing these weight values.

The neurons are grouped in layers in a neural network. Each neuron, in every layer, except in the input and the output layers of the net, is connected with all the neurons of both the preceding and succeeding layers. The input layer neurons are only connected to the neurons of the succeeding layer and the output layer neurons are only connected with the preceding layer neurons. Information propagates from the input layer to the output layer through intermediate layers called ”hidden layers”. Backward propagation of information i.e. from the output to input layer, is also possible for some neural networks. There are normally no fixed number of hidden layers in a net; it can be one or more. However, some types of neural networks do not have a hidden layer. An Artificial Neural Network is loosely modeled on the human brain. It resembles the brain in two respects: one is that knowledge is acquired by the network through a learning process and the other is that the inter-neuron connection known as synaptic weights are used to store this knowledge. This knowledge is used to perform the intended operation of the neural network.

The Natural Neuron

The most basic element of the human brain is a specific type of cell called ”neuron”. The brain is composed of about 10 billion neurons and each of the neurons is connected to about ten thousand other neurons. The power of the brain comes from the number of these basic components and the multiple connections between them. There are four basic components (see Figure 1.1) in all natural neurons: dendrites, soma, axon, and synapses. Each neuron receives electro-chemical inputs from other neurons at the dendrites.These inputs come from the nerves or other neurons. These signals first come at the synapses, which determines to what extent the signals are transmitted to the dendrites, and hence, the

signal is amplified by the synapses in some way. This may result in excitation, a positive contribution to the dendrites' signal, or inhibition, a negative contribution. The dendrites then transport the signals to the soma where they are accumulated. If the sum of these electrical inputs is sufiiciently powerful to activate the neuron, an output spike (i.e. neuron fires at this stage) is produced and is transmitted along the axon to other neurons whose dendrites are connected to any of the axon terminals. Then, these attached neurons may also fire. A neuron only fires if the total signal received exceeds a certain level. The learning process of the brain is accomplished by modifying the synaptic connections by growing new dendrites and lengthening the axon.

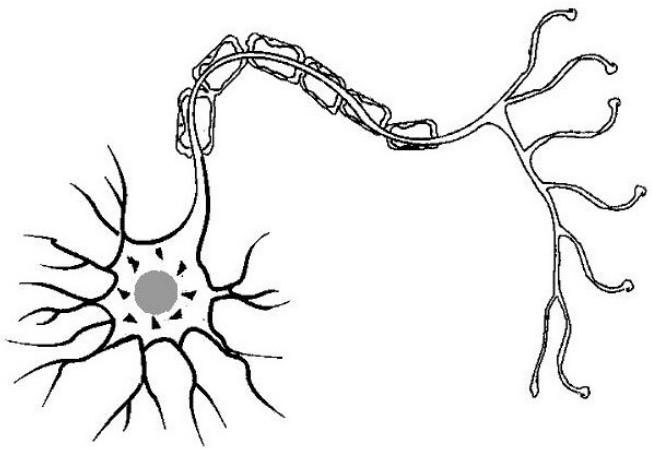

Figure 1.1: A Natural Neuron

The Artificial Neuron

The Artificial neuron simulates the basic functions of the natural neuron. In a biological neuron, dendrites receive an electrical signal from the axons of other neurons; these electrical signals are represented by numeric values in artificial neurons. In an artificial neuron (see Figure 1.2), dendrites are replaced by the input lines that receive inputs from other neurons or the input file. With each of these lines, is associated a weight, which acts like a synapse. Input signals are multiplied by the weights. The soma is modeled by a unit

consisting of a summation followed by a thresholding function. This thresholding function is a step function and modern neural networks usually use the sigmoidal functions. As soon as the sum of the input signals reaches the threshold, the neuron fires, i.e. an output signal is given on the output line, which is connected to other units.

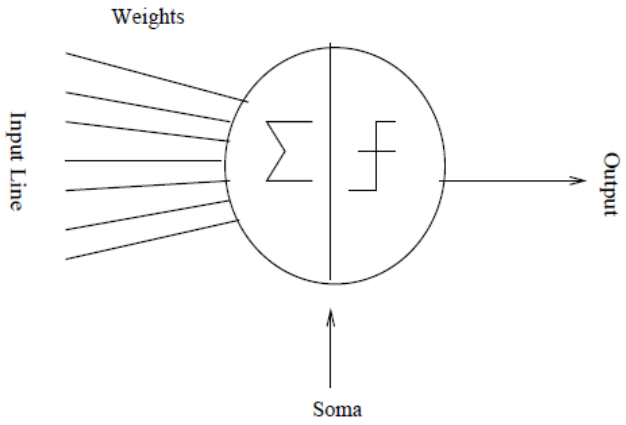

Figure 1.2: An Artificial Neuron

1.2.2 Connection Structures

The connection structure of a neural net deals with how the neurons are connected between layers. There are normally two types of connection structures: Feedforward and Feedback.

Feedforward

In this connection structure of a neural net, neurons of one neuron layer may only have connections to neurons of other layers. An example of such a net type is the Perceptron. Figure 1.3 shows a neural network with feedforward connection structures.

Feedback

Here the neurons of one neuron layer may have connections to neurons of other layers and also to neurons of the same layer. An example of such a net type is the Hopfield Net. Figure 1.4 shows a neural network with feedback connection structures.

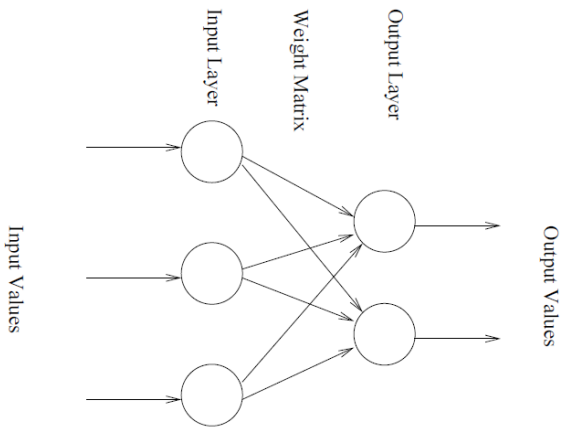

Figure 1.3: Perceptron: An example of Feedforward network

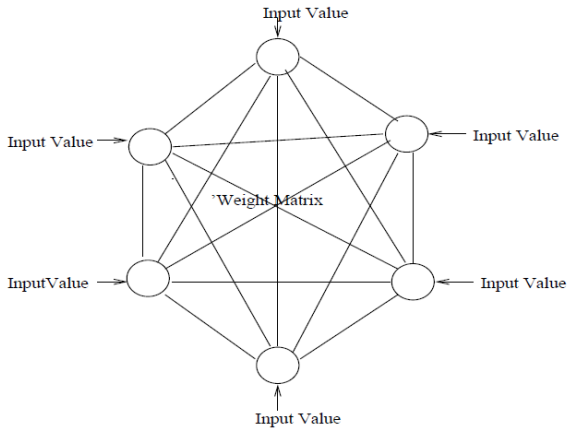

Figure 1.4: Hopfield net: An example of Feedback Network

1.2.3 Types of Neural Networks

There are many types of neural networks available. Researchers are inventing new types of neural networks or at least the variations of the old ones. The types of neural networks can be distinguished by their physical structure, learning method, connection structures etc. A taxonomy of neural networks followed by a selection of neural networks is given below.

Taxonomy of Neural Networks

The field of artificial neural network is evolving so fast that it is difficult to find a concrete taxonomy to classify different types of neural networks. Paplinski has given a taxonomy that begins with two phases of neural networks, active or decoding phase and learning or encoding phase. Artificial neural networks can be classified into feedforward (static) and feedback (dynamic, recurrent) systems from the viewpoint of their active phase, and supervised and unsupervised systems from the viewpoint of their learning phase.

Perceptron

The Perceptron, an invention of F. Rosenblatt in 1958[19], was one of the earliest neural network models. This is a very simple neural network whose weights and biases could be trained to produce a correct target vector when presented with the corresponding input vector. Bias is a constant value (generally 1.0) used as a single input to the network. During training, when all the input values in the network are equivalent to zero, bias helps to update the weight values. A Perceptron has only two layers: input and output. It does not have any hidden layers. It is a feedforward neural network that learns with a supervised learning method. However, Perceptrons have several limitations. First of all, the output values of a perceptron can take only one of two values (true or false). Secondly, they can only classify linearly separable sets of vectors. A set of input vectors are linearly separable if a straight line or plane can be drawn to separate the input vectors into their correct categories. The problem of boolean AND and boolean OR functions are linearly separable, but boolean XOR function is not linearly separable, and hence, a perceptron cannot solve the XOR problem. Figure 1.5 shows the separation of AND and OR functions, it also shows that the XOR function cannot be separated.

Perceptrons are mainly used in simple logical operations and pattern classification.

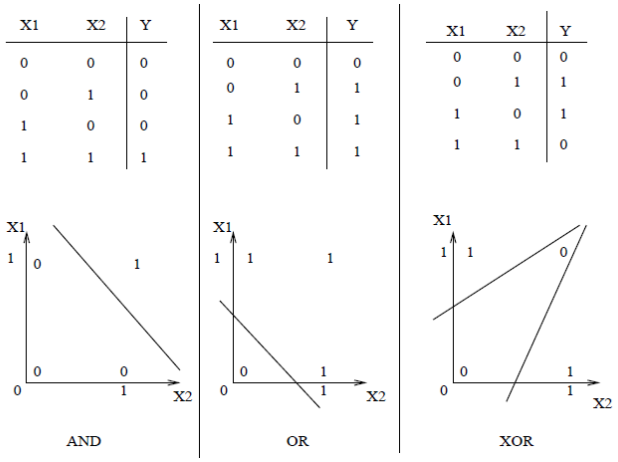

Figure 1.5: Boolean functions AND and OR are linearly seperable, XOR is not

Multilayer Perceptron (MLP)

This is the most common neural network model. It is an extension of the perceptron with one or more hidden layers between its input and output layers. With its extended structure, it has more capability, and thus, it can solve more complex problems, including the boolean XOR function. This type of neural network is used for comparatively complex logical operations and pattern classification.

Backpropagation Net

The Backpropagation neural network is one of the most powerful neural networks. It has the same structure as a multilayer perceptron and is mainly used in complex logical operations, pattern classification and speech analysis.

Like the multilayer perceptron, the backpropagation neural network has three layers: input, output and hidden layers. Before the training of the net, i.e. any data has been run through the network, weights are simply random numerical values, much like a newborn's brain, developed but without knowledge. When presented with an input pattern, each input node takes the value of the corresponding attribute in the input pattern. Then, each node in the hidden layer multiplies each attribute value by a weight and adds them together. If this is

above the node's threshold value, it fires a value of "1", otherwise, it fires a value of "0". The same process is repeated in the output layer with the value from the hidden layer, and, if the threshold value is acquired for this layer, it represents the classified pattern for the corresponding input pattern, which is compared with the actual classification pattern and the error value is calculated. This error value is then backpropagated through the network, and the weights of output and hidden layers are adjusted with these error values. This process is repeatedly carried out until it satisfies a predefined termination condition. The net is then assumed "trained", and the weights are stored. However, the weight change is performed in such a way that the current point on the error surface will descend into a valley of the error surface in a direction that corresponds to the steepest (downhill) gradient or slope at the current point on the error surface. For this reason, backpropagation is also called "gradient descent method."

The advantages of backpropagation neural nets are that they perform well for prediction and classification. On the other hand, there is always a lack of explanation of what the net has learned. Moreover, they are slow compared to other learning algorithms.

Hopfield Net

Hopefield nets are principally used for auto-association. If a distorted input vector is given, the Hopfield Net associates it with an undistorted pattern stored in the network. A Hopfield net consists of a set of neurons and all of them are connected to each other by a unique weight. It means it has only one layer; there is no difference between the input and output layers.(see Figure 1.4) The main applications of a Hopefield Net is storage and recognition of patterns e.g. image files.

1.3 Intrusion Detection Systems

In general, IDS works like a burglar alarm system. It attempts to detect intruders when all preventive devices fail in a computer system. In a computer system, intruders may

penetrate through the firewall and other preventive materials. IDS works by detecting the presence of unauthorized users.

1.3.1 Purpose of IDS

The underlying reasons for using IDS are to protect the integrity of the data and systems from intruders[5]. These intruders can be unauthorized outsiders or authorized insiders trying to do an unauthorized job in the system. Real world security includes prevention, detection and response.

Adequate system security is the first step of ensuring data protection. For example, it is important to prevent access to vulnerable files and authentication databases(e.g. /etc/password or /etc/shadow files) in the system except by the authorized system administrators. Furthermore, a firewall can always be put in place to enforce access control. However, all these preventive mechanisms are not perfect, and hence, the necessity of intrusion detection.

Intrusion detection provides an extra layer of protection to the system. When all the preventive systems such as those mentioned above, fail to prevent the inruder, the intrusion detection system tries to detect him/her and inform the administrator to take appropriate action.

1.3.2 Functions of Intrusion Detection Systems

IDSs perform numerous functions. The following is a list of functions performed by IDSs, although no single IDS performs all of these functions[15]:

- Intrusion Detection Systems continuously monitor the network traffic and behavior of computers and detect irregular and suspicious activities.
- They can audit the system configuration for vulnerabilities and misconfigurations.

- They can recognize known attacks patterns with the help of a continuously updated database of known attacks.
- They can identify abnormal activities through statistical analysis.
- They immediately detect malicious activity and perform defensive tasks.
- They can detect activities originating from the external network, which are allowed by the firewall configuration (such as web browsing) but, in fact, are harmful (such as exploiting a bug of the web browser).
- They can identify unauthorized access attempts by internal users.

1.3.3 Use of Artificial Intelligence in IDS

Issues relevant to intrusion detection include data collection, data reduction, behavior classification, reporting and response[7]. Because of the huge amount of collected data like audit data or network traffic data, it is impossible to process them by hand. Even if we use a computer, it still takes lots of time to analyze this huge amount of data. This is why data reduction is required for further processing like classification, etc. Artificial Intelligence techniques for reduction and classification have been used in many intrusion detection systems for performing these tasks.

Many AI techniques have been used for solving intrusion detection tasks. Some AI techniques that have been used are: Expert systems, Rule based induction, Classifier systems such as Neural Networks and Decision tree, Feature selection, Clustering, etc.[8]

An expert system solves problems by using the computer model of expert human reasoning and it requires continuous maintenance and upgrading for performing well. Behavior classification of intrusion detection systems can be done using expert systems techniques by encoding policy statements i.e. security policy and known attacks as well as system vulnerabilities as a fixed set of rules. User behavior that matches those rules indicates that

an attack is under way. However, the rules must be modified by hand.

Unlike expert systems, Rule based induction derives rules that explain the set of instances describing the problems and steps of solutions. The system acquires knowledge from these rules to solve the problem. In an IDS, rule based systems create and manage rules corresponding to anomalous behavior.

Generally, Classifier systems categorizes different types of patterns from a set of patterns. A classifier like a Neural Network uses a biological system model to perform classification. Another type of classifier called ”Decision Tree” tries to separate the data into two or more groups. Then, it tries to separate these groups into further groups, and so on, until a small groups of examples are left. All these classifiers can be used to classify misuse or anomaly in an IDS.

Feature selection is accomplished by searching subsets of features, or information sources, and testing the ability of those features to perform the intended task. It normally reduces the amount of information required for a particular task. For instance, some data may hinder the classification process, which can be eliminated by feature selection. Moreover, some features may be redundant as the information belongs to those features is already held by other features. Thus, feature selection reduces the computation time eliminating the extra features holding the same information.

In data clustering, data are grouped together according to some common characteristics or criteria. It is used to find the hidden patterns in data that might be missed. Data clustering techniques are also used for reducing the amount of data by dealing with the characteristics of the clusters instead of the actual data.

1.4 Big Data

Big data means very large and complex data sets which are difficult to process using traditional and sequential data processing applications. To deal with large datasets, more com-

pute resources are required to access large amounts of data and perform many calculations across multiple machines concurrently. Data-intensive, parallel and distributed approaches are typically employed, such as the MapReduce programming paradigm.

1.4.1 Apache Storm: Real time data processing

Apache Storm was developed by Nathan Marz in December 2010. The idea behind this development is that there is a need for a real time stream processing system that can be presented in a single program since all the other Big data processors like Apache Hadoop handles only batch processing. Apache Storm empowers developers to build real-time distributed processing systems, which can process unbounded streams of data very fast. A benchmark clocked it at over a million tuples processed per second per node. It is also called ”Hadoop for real-time data.” Apache Storm is highly scalable, easy to use, and offers low latency with guaranteed data processing. A system without real-time facilities cannot guarantee a response within any time frame[17], whereas a real time system like Apache Storm aims to satisfy time constraints. It provides a very simple architecture to build applications called Topologies.

1.5 The Problem

Network intrusions are common on the internet. However, as the technology improves day by day, vulnerabilities to the internet also increases, allowing intruders to apply new mechanisms to damage the internet data. In addition, detecting intrusions in a huge amount of growing data (big data) on the internet is a major problem. Since there is an immense growth in network data and detecting intrusions is difficult, more intruders take advantage of this and perform illegal activities over the internet, making the detection even harder. So, in order to detect intrusions instantaneously in big data network streaming, intrusion detection has to be done in real-time before intruders damage network data.

1.6 The Problem with existing IDSs

There are some intrusion detection systems developed, which can handle real-time streaming of data. These detection systems are trained under the circumstances of limited network connection types; hence, these are not suitable for networks involving multiple connection types. Failing to handle big data and instantaneous detection of intrusions, the real-time streaming implemented earlier supports only the sequential flow of input data. However, a proposal was made to use Apache Hadoop for Intrusion detection in big data. Even though it can handle big data, Apache Hadoop supports merely batch processing. So, this detection system cannot perform real-time streaming and detection where more possibility for attacks exists.[2] Most the intrusion detection systems are trained and tested using KDD '99 intrusion dataset, which is old and not suitable for the current network environment.

1.7 Proposed Solution

According to [23], few requirements of real-time processing are handling stream imperfections, guarantee safety and availability, respond instantaneously etc,. To fulfill these requirements, there is a need for an Intrusion Detection System which can handle real-time streaming big data. In our thesis, a real time hybrid intrusion detection system has been developed using Apache Storm. It is a distributed, fault tolerant, real time stream processor which can handle big data network packets. The hybrid intrusion detection system consists of two neural networks, the CC4 neural network and the MLP neural network. The CC4 instantaneous neural network acts as an anomaly based intrusion detection for unknown attacks. It uses perspective learning and produces instantaneous results which is useful for real-time detection. The Multi Layered Perceptron neural network acts as a misuse based intrusion detection for known attacks. The MLP neural network produces accurate results than any other neural network, provided with sufficient training. The outputs from these two neural networks are used to classify the incoming packet as atatck or normal. We use

these two neural networks in order to avail the instantenous and unkown detection fetaure of the CC4 neural network and accurate detection feature of the MLP neural network.

The rest of the document is divided into four sections: The Chapter 2 includes the review of literature along with the description of underlying neural networks and Apache Storm with a brief description of the intrusion dataset used in our thesis. Chapter 3 describes the pre processing phase of the intrusion dataset and methodology of the hybrid intrusion detection topology and its simulation. Chapter 4 includes the results and findings of the simulation. Chapter 5 includes the conclusion and possible future work.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Related Work on IDS

Research for designing both anomaly based and signature based intrusion detection systems has focused on statistical based, knowledge based and machine learning based IDS. All have their advantages and disadvantages. The most widely used is machine learning based IDSs. In our thesis we use a machine learning based IDS to develop a real-time hybrid intrusion detection system.

There are very few works which are closely related to our model. In 2003, Ravindra Balupari et al[4], proposed a real-time network based anomaly intrusion detection system which analyses data streams generated by Real-time Tcptrace. The Real-time Tcptrace periodically reports statistics on all the open TCP/IP connections in the network. Then, using the Abnormality Factor method, statistical profiles are build for the normal behavior of the network services. Abnormality activity is then flagged as an intrusion. This approach has the advantage of being able to monitor any service without the prior knowledge of modelling its behavior. Although it is efficient, it fails in detecting flows from different network streams.

In 2013, Devikrishna K.S et al[10], proposed an Intrusion detection system that is able to perform live data capturing, so that the system can act as real-time IDS. For testing purpose they used KDD'99 benchmark dataset which is indeed captured by the model for detection of intrusions. The processing of each packet flow is performed in sequence i.e each packet is tested for intrusion and only when processing is complete, a new packet is captured. The

technique for detection used in this model is signature based.

In 2011, Marcelo D. Holtz et al[9], proposed an architecture for distributed Network Intrusion Detection Systems where comprehensive data analysis is executed in a cloud computing environment. Network traffic, operating system logs and general application data are collected from various sensors in different places in the network, comprising networking equipment, servers and user workstations. The data collected from different sources are aggregated, processed and compared using the Map-Reduce framework, analysing event correlations which may indicate intrusion attempts and malicious activities. The detection mechanism proposed is hybrid and above all it can easily handle Big Data streams as it is using Hadoop. However the model is not implemented and is a proposal only. Furthermore it cannot handle real-time streaming of network data to detect intrusions.

2.1.1 Problems in earlier works

There exists some minor and major problems in the earlier works and these are listed below. This helps in developing a more advanced and better Intrusion detection system.

- Even though Marcelo D. Holtz et al [9] proposed a Big data handling Intrusion detection system, Apache Hadoop which is a big data batch processing tool is used in this work. Therefore this architecture fails to handle real time network streaming.
- Real time processing is possible in the work proposed by Devikrishna K.S et al[10]. However, the processing of streams is done sequentially which is not suitable for obtaining instantaneous or real-time results. In addition, this model fails to handle big data because of no parallel processing and absence of big data handling framework or environment.
- The intrusion detection model in [4] is restricted to work on only few network streams like TCP/IP. It cannot guarantee to show similar results for other network protocol environment, even though it may support real time streaming.

- Almost all the architectures discussed above are tested using KDD 99 DARPA dataset which is not suitable in the current network environment[6], since it does not have any recent attacks and the current network routers can easily handle the attacks in KDD '99 dataset.

- Devikrishna K.S et al[10], used live data capturing of network streams for their model. However the detection model is not hybrid. It involves only signature based detection which may have good accuracy but is not instantaneous, which is the primary requirement for real time intrusion detection.

2.2 CC4 Neural Network

The CC4 Neural Network is an instantenously trained neural network proposed by Kak[11]. Such a memory requires quick learning. CC4 uses a feedforward network architecture consisting of three layers of binary neurons. The number of input neurons is equal to the length of the input vector plus one, the additional neuron being the bias neuron having a constant input of 1. The input layer requires the input to be in unary format which makes the algorithm faster but hinders the applicability of the algorithm. The number of hidden neuron is equal to the number of training samples, where each hidden neuron corresponds to one training sample. The hidden layer is connected to all the neurons in the input layer and similarly the ouput layer is fully connected to the hidden neurons. The general architecture of CC4 network is shown in figure 2.1

The activation function is the binary step function, whose output equals to 0 if the sum of all weighted inputs does not exceed its threshold, and equals to 1 otherwise. A bias neuron has been added to the input layer, so the thresholds for all hidden and output neurons are taken as 0 for simplicity.

The weights on hidden neurons are assigned by using the training sample itself. For each training vector presented to the network, if an input neuron receives a 1, its weight to the

hidden neuron corresponding to the training vector is set to 1. Otherwise it is set to -1. The bias neuron is treated differently. If s is the number of 1's in the training vector, excluding the bias input, and the desired radius of generalization is r, then the weight between the bias neuron and hidden neuron corresponding to this training vector is r - s + 1. Thus, for any training vector x_i of length n including bias, the input layer weights are assigned according to following equation:

$$y = \begin{cases} 1 & \text{if } x[n] > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Similarly, if the training vector produces a 1 at an output neuron, the weight from its hidden neuron to that output neuron is set to 1. Otherwise, it is set to -1.

The CC4 algorithm can be formally stated as follows[16]:

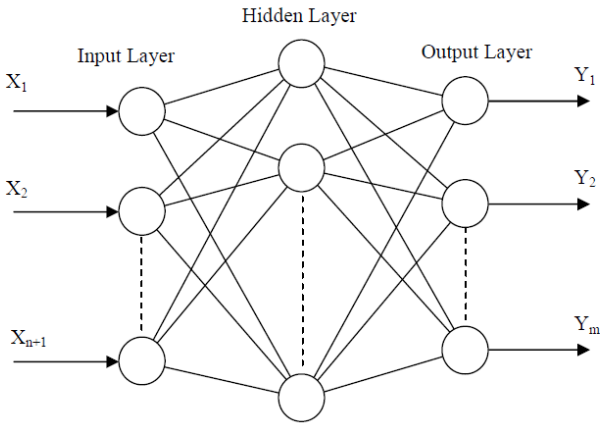

Figure 2.1: General CC4 Network Architecture

2.2.1 Radius of Generalization

Generalization vector is the user defined value with which there will be changes in the output[18]. Therefore the value should be chosen carefully as it helps in classification of

Algorithm 1 CC4 training algorithm

for each training vector $x_i[n]$ **do**
    $s_i \leftarrow no.\ of\ 1s\ in\ x_i[1 : n-1]$
    for j = 1 to n -1 **do**
        if x_i[j] = 1 **then**
            $w_i[j] \leftarrow 1$
        else
            $w_i[j] \leftarrow -1$
        end if
    end for
    $w_i[n] \leftarrow r - s_i + 1$
    for k = 1 to m **do**
        if y_i[k] = 1 **then**
            $u_i[k] \leftarrow 1$
        else
            $u_i[k] \leftarrow -1$
        end if
    end for
end for

the input vectors based on the class of stored vectors. If the hamming distance between the new input vector and any of the stored vectors is less than or equal to the user defined radius, the outputs of all such stored vectors is considered for generating output of the input vector. The number of 1s and 0s in every bit location of the output vector of all these stored vectors is calculated and added up. If the result is positive, the corresponding output neuron outputs 1 otherwise the output is 0.

2.3 Two Layered Feed Forward Neural Network

The Real time intrusion detection system uses a two layered feed forward neural network. This network is Multi layered perceptron and we call it as MLP. There are many activation functions and error minimizing mechanisms for MLP. We use sigmoid activation function which works best for backpropagation algorithm and gradient descent mechanism for error minimizing. The flow diagram of two layer feed forward network is show in figure 2.2.

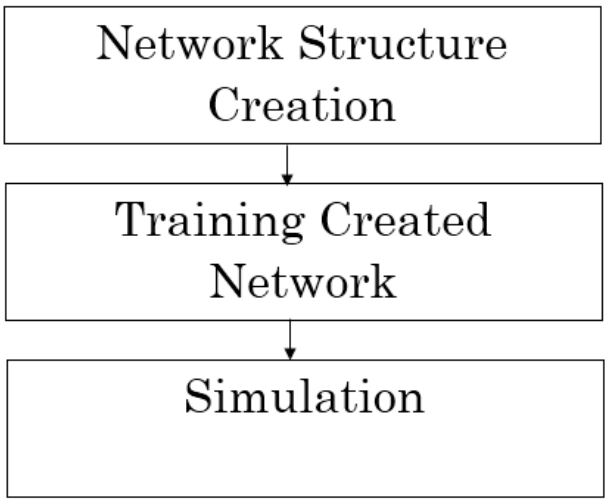

Figure 2.2: Flow diagram of Two layered Feed Forward Neural Network

2.4 Apache Storm

The job inside a storm cluster is distributed to different components, where each of these are responsible for processing a simple, specific task. The input stream of the cluster is handled by a component ”spout”. The data is then processed to another component called ”bolt” by the spout. The transformation of the data in a bolt can be typically of two ways, i.e either a bolt produces the result or a bolt passes the transformed data to another bolt[1].

2.4.1 Terminology

Tuple

The tuple is the main data structure in the Storm. A tuple is the named list of values, where each value can be any type. Tuples are dynamically typed – the types of fields do not need to be declared.

Stream

Stream is the core abstraction in Storm cluster. A stream is an unbounded sequence of tuples. In Storm a stream can be converted into distributed and reliable way.

Spout

A spout is the source of streams. A spout can read a list in any format and convert them in the form of streams. Example: Spout may connect to the Twitter API and emit a stream of tweets.

Bolt

A bolt consumes any number of streams, does some processing and emits either in two ways as discussed. If the streams are complex there will be more computing than usual and hence a bolt will pass the data to another. In this case there will be multiple bolts. Bolts

can do anything from run functions, filter tuples, streaming joins, connecting to database and more.

Topology

Combination of spouts and bolts in a network is called a topology. A storm runs a ”topology” as a Hadoop cluster runs a ”map-reduce” job[1]. So, everything submitted to a storm cluster will be a topology. There can be links between any spout and any first layer bolts and any first layer bolt to any next layer bolt depending upon the need for the job. A simple storm topolgy is shown in figure 2.3.

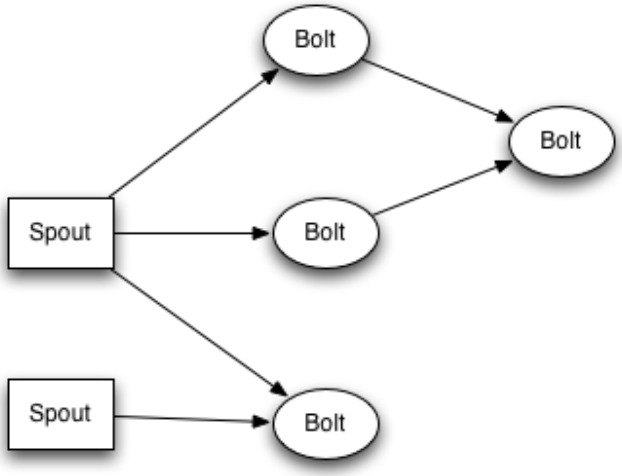

Figure 2.3: A Simple Storm Topology

2.4.2 Grouping

As seen in the figure 2.3, a storm topology can have any links between spouts and bolts. The links between spouts to bolts and bolts to bolts are done based on the ”Grouping”. The components are grouped based on the need and so there are different grouping mechanisms for different situations[12].

Shuffle Grouping

This is the most commonly used grouping. It takes a single parameter (the source component) and sends each tuple emitted by the source to randomly chosen bolt making sure that each consumer will receive the same number of tuples.

Fields Grouping

Fields Grouping allows to control how tuples are sent to bolt, based on one or more fields of the tuple. It guarantees that given set of values for a combination of fields is always sent to same bolt.

All Grouping

As the name suggests, this grouping sends a single copy of each tuple to all instances of receiving bolt.

Direct Grouping

This is a special grouping where the source decides which component will receive the tuple.

Global Grouping

Global Grouping sends tuples generated by all the instances of the source to a single target instance.

Custom Grouping

This kind of grouping allows the user to create their own style of grouping based on their need which is possible using an interface in storm cluster.

2.4.3 Components of Storm

In a storm cluster, nodes are organized into a master node that runs continously.

There are two kinds of nodes in a storm cluster: *master node* and *worker nodes*. Master node run on a daemon called *Nimbus,* which is responsible for distributing code around the cluster, assigning tasks to each worker node, and monitoring for failures. Worker node runs on a daemon called *supervisor,* which executes a portion of a topology.

Storm keeps all the cluster states or data either on a local disk or in *Zookeeper*. Since this is the case, the daemons are stateless and can fail or restart without affecting the health of the system. Figure 2.4 gives the idea of components in storm cluster[12].

Figure 2.4: Components of Storm Cluster

2.5 Intrusion Dataset

We use the dataset for Intrusion Detection System created by Information Security centre of Excellence(ISCX) in the year 2012[22]. It is the benchmark dataset collected for seven days under practical and systematic conditions[22]. It is a labeled dataset consisting of normal and attack flows. The total number of flows are 2,450,324 among which the normal flows constitutes of 2,381,532 and the attack flows are 68,792. The total size of the dataset is 85 Giga Bytes and so we can consider as Big Data compared to remaining datasets. Also, we use only the sessential attributes of the dataset that helps for better and fast detection of the intrusions which includes the attribute ”tag” which is only used for training, since it specifies whether the flow is attack or normal. The attack scenarios of the dataset are as follows:

- Infiltrating the network from inside
- HTTP denial of service
- Distributed denial of service using an IRC botnet

- Brute force SSH.

CHAPTER 3

METHODOLOGY

In this chapter we present a real-time IDS architecture based on CC4 neural network and MLP neural network in Apache Storm. We also discuss intrusion dataset processing.

3.1 Intrusion Dataset Pre-Processing

The total number of features available for each packet of ISCX Intrusion dataset are 19[22]. In order to extract the essential attributes out of these 19 attributes, we use the feature extraction method built in the Weka Data mining software[13]. The feature extraction is a process by which we search for the best subset of features in our dataset. In our work, best subset typically means the subset that gives highest accuracy. State space searching process in WEKA is used to select the feature subset. This feature extraction method suggested 11 important features from all the features. A similar feature extraction mechanism was used in [20], where they used the same extracted features. However, in our work we did an essential change by creating a new feature called ”duration”, which contains the same result which is obtained from combining two features ”startDateTime” and ”stopDateTime”. The nature of packet depends on how long the packet is in the network and it is essential in detecting attacks, such as, User to Root attack where the intruder spends lot of time to gain root access. So, in order to consider this essential factor we need the time the packet was active, which refers to a calculated time like ”duartion”, rather than start and stop times seperately. The list of features used for the simulation can be seen in the table 3.1. The proportions of normal and attack flows we use are 76% and 24% respectively and the proportions used for training and testing are 51% and 49% respectively. The packet

distribution for training and testing from each day of the dataset collection can be observed in table 3.2.

S.No	Features
1	appName
2	totalDestinationPackets
3	totalSourcePackets
4	direction
5	source
6	protocolName
7	destination
8	duration
9	tag

Table 3.1: List of Extracted features from ISCX Intrusion Dataset.

Date	Training		Testing	
	Normal	Attack	Normal	Attack
12th	20673	2086	0	0
13th	25692	1386	0	0
14th	16318	1978	0	0
15th	0	0	30809	37241
16th	33747	11	0	0
17th	0	0	22588	5219
Total	**101891**		**95857**	

Table 3.2: Distribution of Training and Testing data.

3.1.1 Conversion into decimal format

Initially the dataset is converted into decimal format. Each unique feature value is assigned a decimal number starting from 0, which will be its Unique Identification Number. This list

is different for each feature in the dataset, resulting in the format where a packet contains the values for each feature indicating its unique identification number. In our implementation, among the two neural networks i.e CC4 and MLP, the MLP neural network can work and indeed works faster with the decimal format, so there is no need for further processing of data for MLP. The CC4 neural network accepts only the unary format as input. The dataset therefore needs to be further processed for the CC4 neural network.

3.1.2 Conversion into floating point

In order to convert the resultant dataset into unary format, we consider each feature value and its contribution towards any kind of attack in the entire dataset.

We obtain the contribution values as follows:

1. For each feature of the dataset,

2. Consider each unique feature value of this feature, and

3. Calculate the ratio of total number of attack packets that contains this feature value to that of total attack packets.

The floating ratios ranges from 0.0 to 0.9 and the minimum, maximum and average contribution values among each feature is available in the table 3.3.

Feature	**Minimum**	**Maximum**	**Average**
appName	0.0	0.43	0.009
totalDestinationPackets	0.0	0.18	3.04 E -4
totalSourcePackets	0.0	0.27	4.15 E -4
direction	0.0	0.53	0.25
source	0.0	0.37	4.03 E -4
protocolName	0.0	0.9	0.166
destination	0.0	0.9	2.90 E-5
duration	0.0	0.26	1.47 E -5

Table 3.3: Contribution range of each feature towards an attack.

Further, based on the floating values we created a quantization mapping table which indicates an unary value for each range of floating values. Hence the entire dataset is converted into unary format based on the mapping table 3.4.

Range	Mapping
0.0	0000000000
0.00000001 - 0.0000001	0000000001
0.0000001 - 0.000001	0000000011
0.000001 - 0.00001	0000000111
0.00001 - 0.0001	0000001111
0.0001 - 0.001	0000011111
0.001 - 0.01	0000111111
0.01 - 0.1	0001111111
0.1 - 0.3	0011111111
0.3 - 0.55	0111111111
0.55 - 0.9	1111111111

Table 3.4: Quantization Mapping of floating point data to Unary data.

3.2 Approach

A Real-time Intrusion detection system is proposed in which Apache storm works as a Real-time streaming processor. A hybrid intrusion detection system which detects network attacks.

A. The IDS is composed of a CC4 neural network algorithm which serves as an anomaly IDS. This tells whether the attack is either unknown attack or Normal/Known attack.

B. The MLP serves as a Misuse IDS which specifies whether the attack is either Normal or harmful attack.

The outputs of the CC4 neural network and the MLP neural network is the input to a post-processing unit which defines the class of the attack. Thus a Real-time Intrusion detection system is proposed which not only detects the intrusion but also specifies the class of attack. The proposed system therefore detect ”on the fly” attacks based on network streams. The basic architecture of the hybrid IDS is shown in figure 3.1.

3.2.1 Advantages

There are a few advantages to using the specific event processor and algorithms for intrusion detection which makes the proposed Real-time IDS the best alternative to any existing IDS.

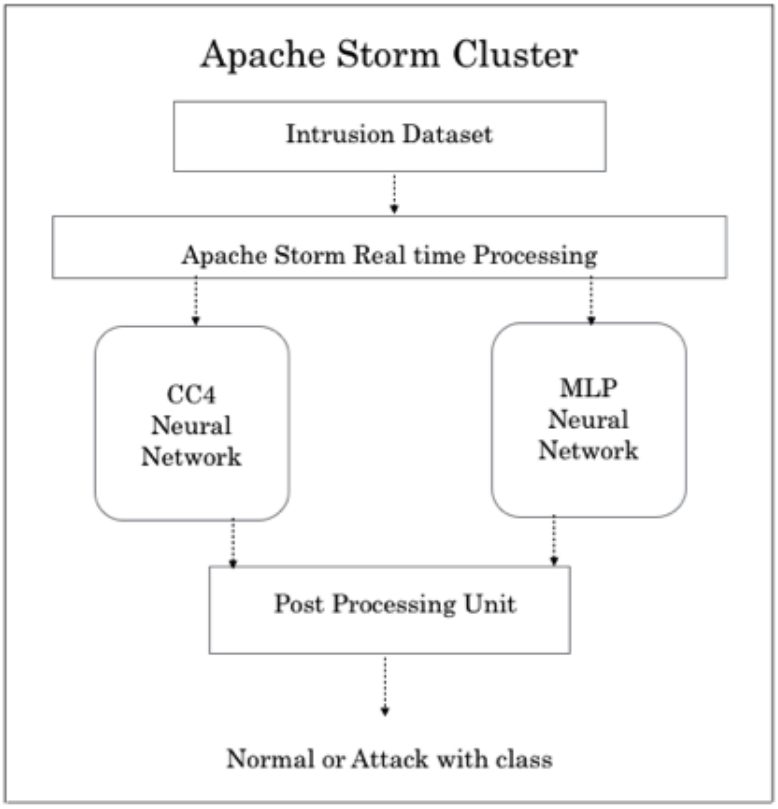

Figure 3.1: Basic architecture of proposed IDS

Apache Storm

Apache Storm is a real-time event processor. It has a simple architecture which handles data without much human intervention. Apache Storm is a dynamic real-time processor. Introducing a new worker in Storm while it is running is easy and all the connections are formed automatically, whereas hadoop is more complex when we introduce a new worker while it is running[14]. Another important advantage of using apache storm is any programming language can be used to perform a job in a Storm cluster.

A. CC4 Neural network

CC4 produces instantaneous training and generalization which does not require a lot of training data and computation is simple. The instantaneous training helps faster learning of new patterns, which help identifying known and unknown attacks in real time. The CC4 IDS is therefore used for learning new unseen patterns.

B. MLP Neural Network

The MLP neural network is used to detect known patterns. When MLP is well trained it shows better accuracy in differentiating normal and attack patterns than any other IDS. So, the accuracy along with CC4 is high and it gives low false negavtive and low false negavtive rates, which was proved in the work[16].

3.3 Proposed Storm Topology

The proposed Storm topology consists of a spout which does most of the real-time processing. After the pre-processing phase, the spout sends the tuple to two bolts by performing ”Shuffle Grouping”. One bolt acting as the CC4 Launcher and another bolt serves as a MLP Launcher. CC4 Launcher converts the decimal format input to floating, then to unary using quantization mapping table and sends the unary format of input to a bolt that executes the CC4 algorithm. MLP Launcher bolt obtaines the weights of best matched packet from the signatures and sends the data to a bolt that performs MLP intrusion detection. Each

and every tuple is sent to both bolts which individually run to produce their results. The outputs from both the bolts (CC4 and MLP) are sent to another bolt which does the final phase of the system i.e the post-processing unit. This determines the class of attack. The bolt running the CC4 algorithm, in addition to sending the output to post-processing bolt, also sends its output to another bolt called MLP-2 (discussed later). The Storm topology is shown in figure 3.2.

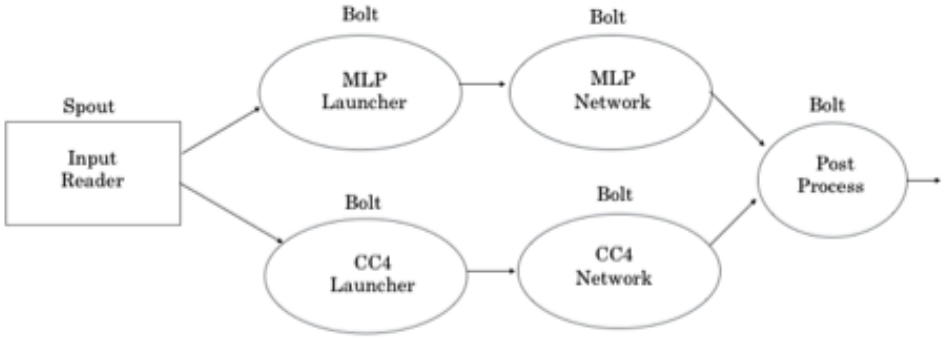

Figure 3.2: Proposed Storm topology

3.3.1 MLP Launcher

This bolt after receiving the tuple from the Input Reader spout, checks for the maximum match of the packet pattern from among the MLP signatures stored during the training process. After obtaining the signature i.e possible weights of neurons for the input packet, the MLP launcher sends this information as a tuple to the next bolt i.e the MLP Network.

3.3.2 CC4 Launcher

This bolt after receiving the tuple from the Input Reader spout, converts the decimal format input to floating, then to unary using quantization mapping table and then sends the unary input packet to its next bolt i.e CC4 Network.

3.3.3 MLP Network and CC4 Network

The MLP Network and CC4 Network bolts contains the MLP and CC4 neural network code respectively. After processing and obtaining the results, they send their outputs to the post-processing bolt, which gives us the classification i.e normal, infiltrating the network from inside attack, HTTP denial of service attack, distributed denial of service attack, brute force SSH attack, unknown attack.

3.4 Implementation of the CC4 Neural network

The intrusion dataset is converted into unary in the pre-processing phase, to prepare it for the CC4 neural network. Although the range for the contribution to attack into unary is not regular (see table 3.4), the CC4 algorithm shows better results in distinguishing between known and unknown attacks using the irregular range rather than a regular range as it was reported in the work [16]. This supports our purpose of using the CC4 neural network for anomaly intrusion detection.

The number of bits in the input to the CC4 bolt are 80 bits (8 features x 10 unary bits). The CC4 neural network size increases based on each unique input vector. Hence the size of network increases significantly if the input is given as 80 bits. This decreases performance. Hence, we use a divide and conquer strategy and implement the CC4 network individually for each input bit. This improves the performance since the unique input is minimal. The structure of the parallel CC4 network is seen in figure 3.3. Each CC4 network gives a one bit output i.e either a 0 or 1. The total input (80 bits) gives 80 bits as output since we run the individual CC4 networks in parallel.

During the CC4 network training process all the possible training packets are processed similarly and the resultant of each packet is stored as a CC4 signature for testing. That is,

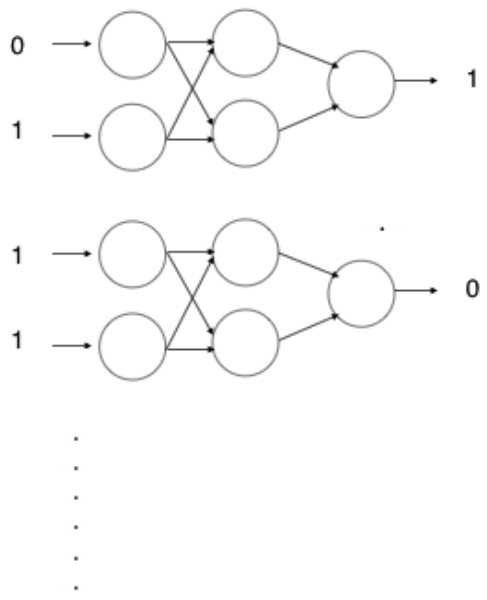

Figure 3.3: Parallel CC4 Neural Networks.

during the testing phase, after we obtain the 80 bit output we compare those bits with the signatures we have. We calculate the best percentage match for the output, given the different signatures and if the best percentage match is above the threshold (since CC4 works on the basis of threshold activation function) it outputs as 0 and if the percentage match is below the threshold the CC4 outputs the result as 1. A value above the threshold of CC4 threshold limit indicates the normal flow and known attacks flows. Whereas, the value below threshold indicates the unknown pattern or attack. After multiple simulations with the ISCX dataset we observed the threshold value to be best at 85%. The best percentage match graph for the CC4 network can be seen in figure 3.4. After obtaining the final output based on the threshold function the output is sent to the post-processing bolt.

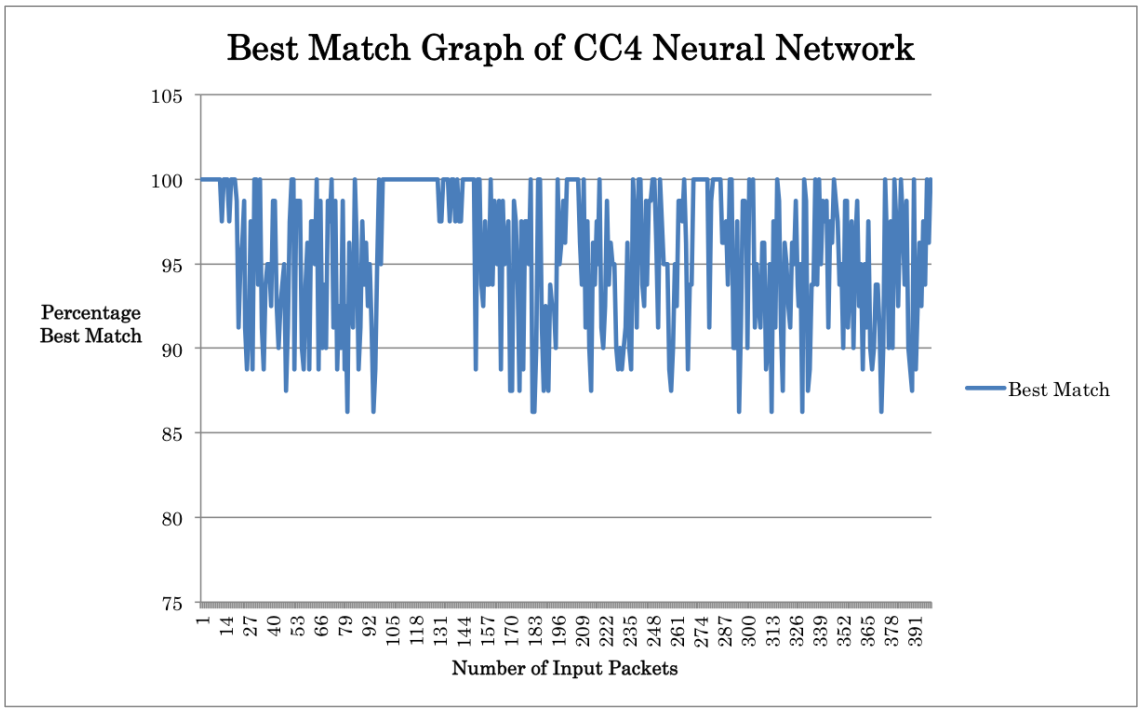

Figure 3.4: Best Match Graph of CC4 Neural Network.

3.5 Implementation of the MLP Neural Network

The second neural network that we use in our work is the multi layer perceptron which is a basic multi layer neural network containing input, hidden and output layers. We implemented the neural network with 8 input neurons, 8 hidden neurons and one neuron for the output layer indicating 1 for attack and 0 for normal. 8 neurons were used because each neuron of the input layer carries each feature from the input vector. During the training we used the gradient descent method for error minimizing and adjusted the weights accordingly. The activation function we used for the MLP neural network is the sigmoid activation function. Which has been proven to work very well for the backpropagation technique.

Since the MLP can accept input in decimal format as we discussed in the pre-processing phase, there is no requirement for further processing of the input. However, the accuracy of MLP is an isssue which is dependent on the level of training. Usually the MLP shows

higher accuracy with increased training. Hence, an improper training or insufficient training can lead to very low accuracy, making the usage of MLP unnecessary.

During the training process, after the MLP has been trained for a sufficient number of iterations in backpropagation method using the gradient descent mechanism for error minimization, the weights of each neuron in the network are stored based on the input. During the testing phase, the input is searched with available signatures obtained from training. Even though we may or may not find the exact input we consider the most/best matched input among all the stored ones and then consider their weights for testing the network. Based on the sigmoid activation function if the output is 1 it indicates attack and normal if the output is 0. However, in our simulation we send this output to the post-processing phase/bolt for classification of the input packet.

3.6 Post Processing Unit

The Post Processing Unit is responsible to determine whether the incoming packet from the network belongs to a normal class or a known attack class or an unknown attack. After receiving the output from both the bolts i.e the MLP Network and the CC4 Network with respect to an input packet, the combination of outputs from the two neural networks creates the following possible cases in the final output.

- **Normal Packets:** If the CC4 output indicates 0 i.e the best percentage match graph is above a threshold level, and if the MLP outputs as 0, the post processing unit declares the final output as a normal packet.

- **Known Attack Packets:** If the CC4 output indicates 0 i.e the best percentage match graph is above a threshold level, and if the MLP outputs as 1, the post processing unit declares the final output as a known attack packet. It also indicates the class of attack obtained from the signatures.

- **Unknown Attack Packets:** The detection of an unknown attack is difficult and in our simulation the CC4 neural network detects such attacks. The MLP is a purely training based neural network and so when there is a new attack for which the MLP is not trained, MLP will fail to detect it. However, according to [16], the CC4 neural network is prone to produce irregular percentage matches if the radius of generalization is not chosen properly. This radius varies from input to input i.e it changes according to the dataset. So, in order to confirm the correctness of the CC4 result, we iterate the packet in CC4 for a certain number of times if the result is 1. Even after the iterations, if the result is still 1, we introduce a new MLP called MLP-2 and implement this as a seperate bolt (see figure 3.5), which works offline. MLP-2 is trained with the new attack and signatures produced by MLP are updated so that the actual MLP will detect the new attack for which now it has the signatures. If after the iterations the CC4 produces the result as 0 by adjusting its errors, the system treates the packet as normal.

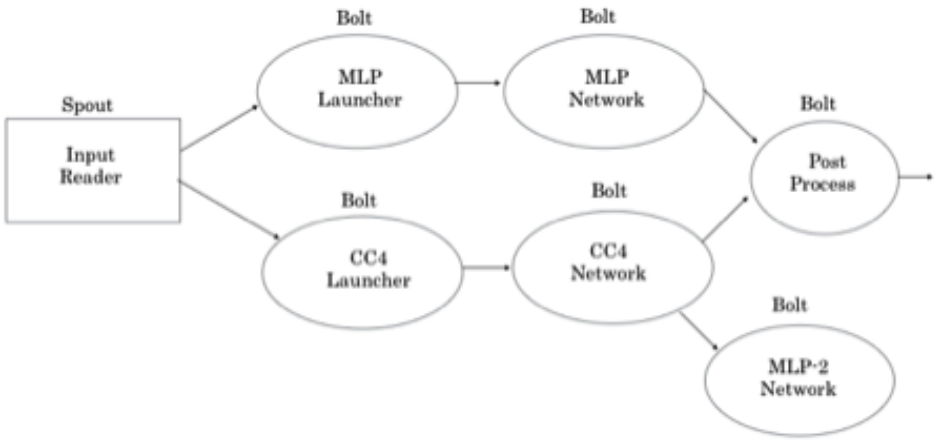

Figure 3.5: Improved Storm topology

3.7 Real time processing

In our work, we are not streaming input data from the network. Instead we are using a dataset to implement and prove the system. We therefore, introduce a ”real-time unit” which handles the tuples in the dataset. This means that when running the apache storm cluster we will have the ability to stop the storm cluster whenever we want i.e a storm cluster is a never stopping process until manually done or write storm code to shutdown the system (i.e. *cluster.shutdown()*). In order to test the real-time streaming of the storm cluster, we keep the cluster to sleep for long time after complete reading of the input file. Meanwhile another small program writes some more network input packets into the input file. After the sleep and mandatory wake up mode, the input file check phase is done as usual. Storm cluster identifies the next written packets and continues to stream them over the detection system. This proves that the intrusion detection system is real-time and handles heavy network traffic.

CHAPTER 4

FINDINGS

This chapter includes the experimental results that are obtained during the simulation. All the simulations were run on a system with 4.00 GB of memory, 2.5 GHz intel core i5 processor running with Machintosh OS X 10.9.5 operating system. The neural network coding and apache storm cluster development were written in Java.

4.1 Dectection of New attacks using CC4 Neural Network

The CC4 generates outputs according to best percentage match. After processing the input packet, the result is compared with the stored signatures of normal or known attacks. Based on threshold function if the best percentage match falls above the limit the output is 0 else it is 1. After multiple simulations we found the threshold to be about 85%. So, if the best match is below the threshold, CC4 alarms as unknown attack.

As explained in the previous chapters the CC4 neural network's primary purpose in our work is to detect anomaly attacks. In order to prove the capability of the CC4 neural network for anomaly detection, we intentionally let the CC4 train for all attacks expect the Distributed Denial of Service attacks which are found on the 5th day of our intrusion dataset collection. Hence when the CC4 is given this as input, it should detect this as an unknown attack, since there is no signature available for this attack.

Figure 4.1 shows the best match graph of CC4 before it is trained for the DDoS attack. A few packets fall below the threshold i.e 85% indicating that the signatures are not available. For this case the CC4 outputs the result 1.

When the CC4 is trained with Distributed Denial of Service attacks, the attack will be

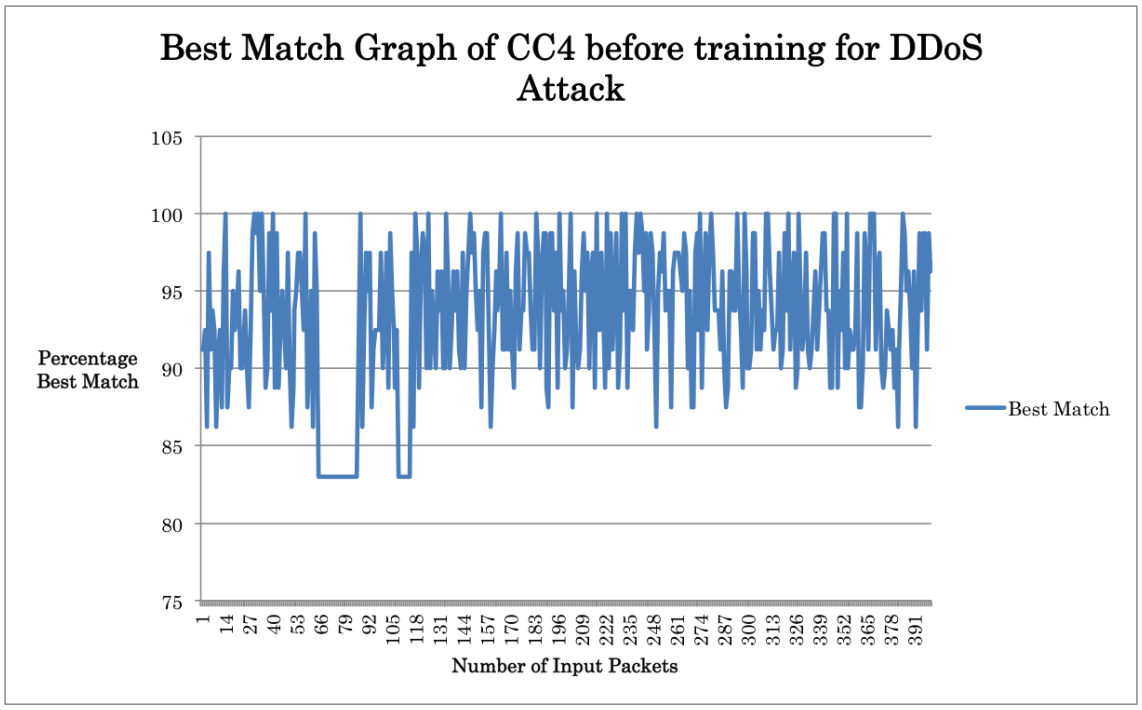

Figure 4.1: Best Match Graph of CC4 Neural Network before training for Distributed Denial of Service attack.

detected, since now it has the signatures stored for the DDoS attacks. After training for these attack packets, the CC4 is ran with the same dataset which was ran before but now the DDoS packets are detected since the best percentage match is above the 85% threshold, which can be observed in figure 4.2

4.2 Accuracy of MLP Neural Network

The training of the multi-layered perceptron is done using the training data where it contains the labeled tag indicating the packet as attack or normal. Based on that label the MLP is trained initially with random weights and the error is checked with the output label. Inorder to minimize error, backpropagation is implemented which changes the weights according to the error. The number of times this process is done is controlled by number of iterations set by the user. However after certain number of iterations there will be no

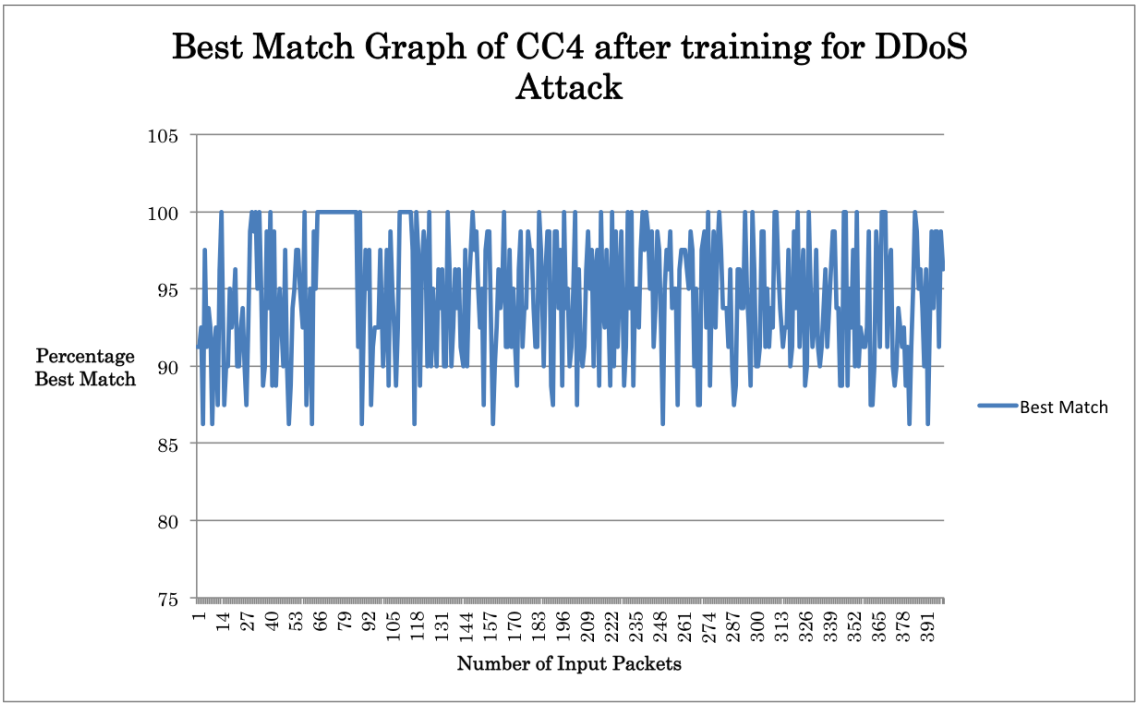

Figure 4.2: Best Match Graph of CC4 Neural Network after training for Distributed Denial of Service attack.

change in weights. This indicates that the number of iterations is directly proportional to accuracy of the system. It is obvious that MLP will not give best accuracy in detecting attacks when it is not trained well, because those weights may work only for few test packets but not to the maximum extent. We simulated the MLP and checked the accuracy of MLP at regular intervals of iterations. Figure 4.3 shows the accuracy of MLP in detecting attacks after each 50 iterations upto 200 iterations. The best accuracy of MLP was found after 200 iterations.

4.3 Accuracy of CC4 based on Radius of Generalization

The CC4 neural network can learn instantaneously unlike MLP. It uses perspective learning, which means an input packet is sent only once through the network during the training phase, which means it has no backpropagation. For this reason we use this neural network,

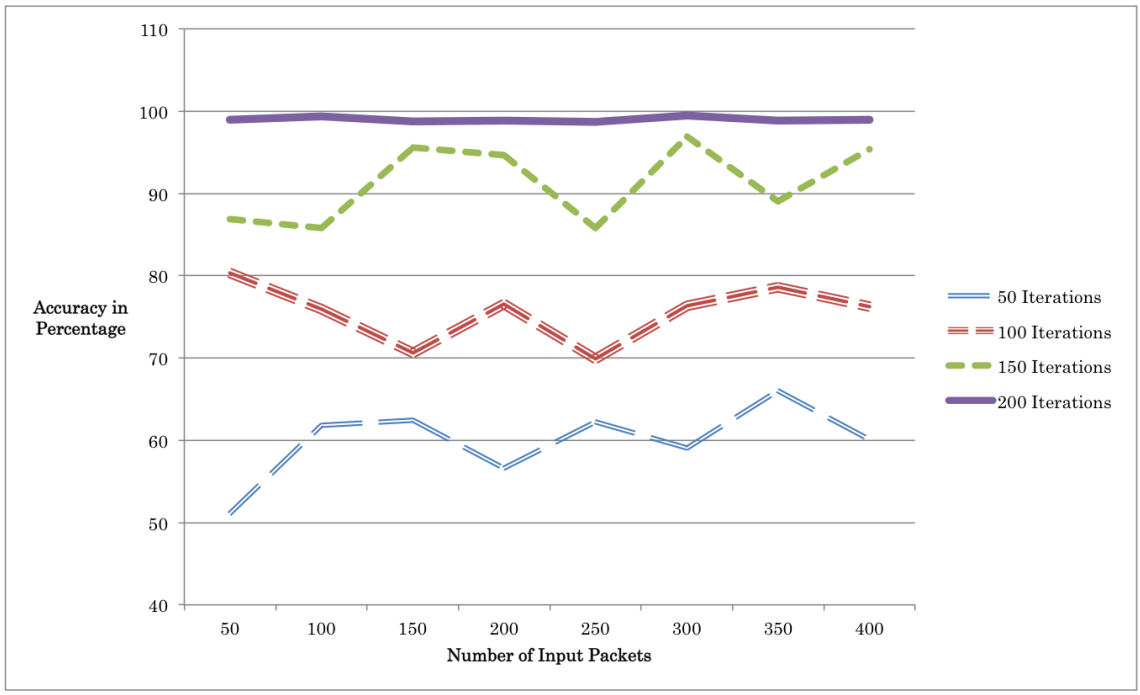

Figure 4.3: Accuracy of MLP Neural Network after certain iterations.

since it is instantaneous. Unlike MLP, CC4 does not depend on the length of training for accuracy. However, it has a factor, radius of generalization which distinguishes the trained packets to new packets and so this value cannot be static to any kind of input. After vigorous training and testing we found that, for ISCX intrusion dataset the CC4 has better accuracy in detecting unknown attacks for the radius of generalization, r = 1. It was different in the case of [16], since that work used KDD'99 DARPA dataset. The accuracy of the CC4 network can be observed in th figure 4.4

4.4 Use of Apache Storm in MLP Training

As discussed earlier, the MLP training phase is time consuming. This is because in order to improve the accuracy of the neural network, it needs to be trained over many iterations since MLP has backpropagation to adjust weights. We are using apache storm in our implementation, not only to avail the advantage of real-time streaming but also to handle big

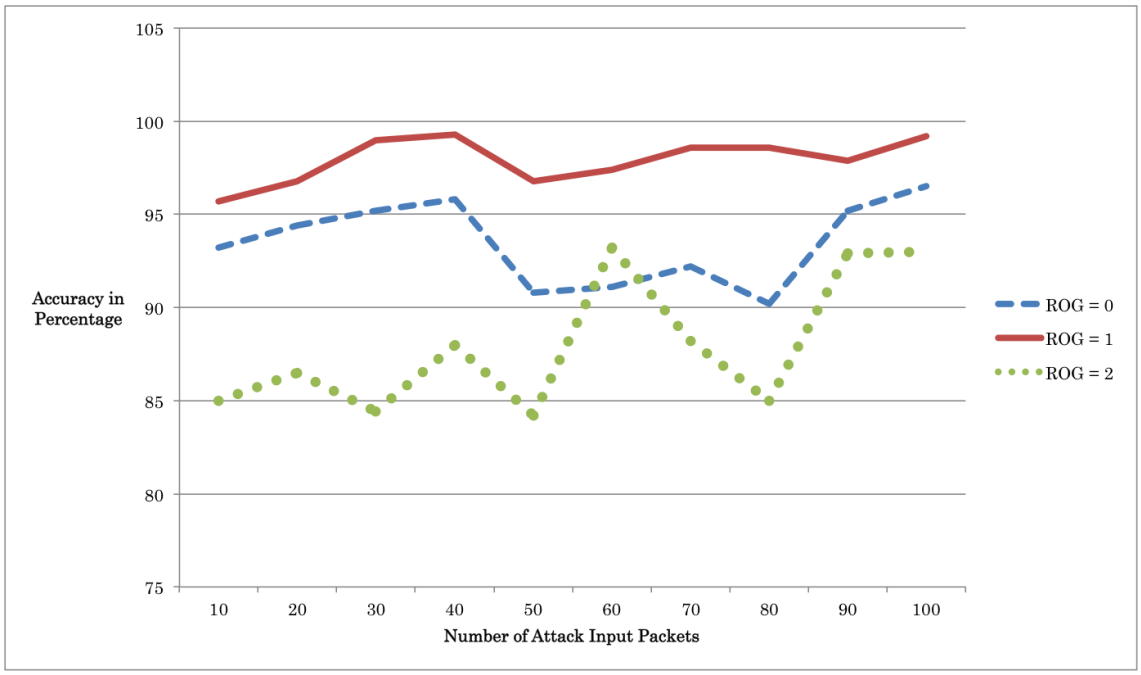

Figure 4.4: Accuracy of CC4 Neural Network for different radii of generalization.

data. Hence, we can expect faster processing in apache storm compared to normal training outside the apache storm cluster. As MLP training takes more time, we tested the training process in apache storm to take advantage of big data handling. The results in figure 4.5, shows that storm although shows normal execution time for small input (i.e. fewer number of packets) it reforms better and better with a increase in number of packets.

4.5 False Positives and False Negatives

Recognition Systems, in most cases cannot have 100% accurate results, since they are trained using a data which may or may not match the nature of data that is used for testing. Hence, there are possibilities for false positives and false negatives of the recognition systems. In our thesis, we used two neural networks for the hybrid intrusion detection system. Inorder to test the false positives and false negatives of the CC4 and MLP neural networks we used 400 random packets from the dataset which constitutes of 185 normal packets and 215 attack packets. These are tested for false positives and false negatives sequentially with

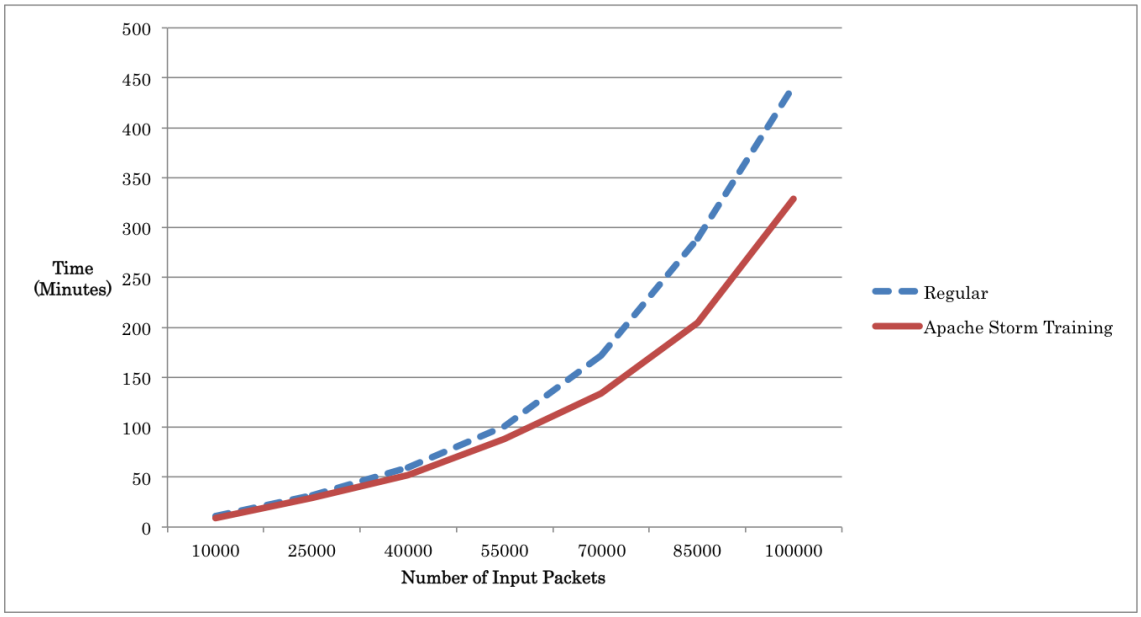

Figure 4.5: Training of MLP inside and outside Apache Storm.

normal and attack packets. The MLP neural network shows 0% false positive and 0% false negative rate with respect to atacks, which can be seen in figure 4.6. However the CC4 neural network has 4.32% of false positive rate and 0% false negative rate with respect to attacks. Figure 4.7 shows the false positives and false negatives of the CC4 neural network.

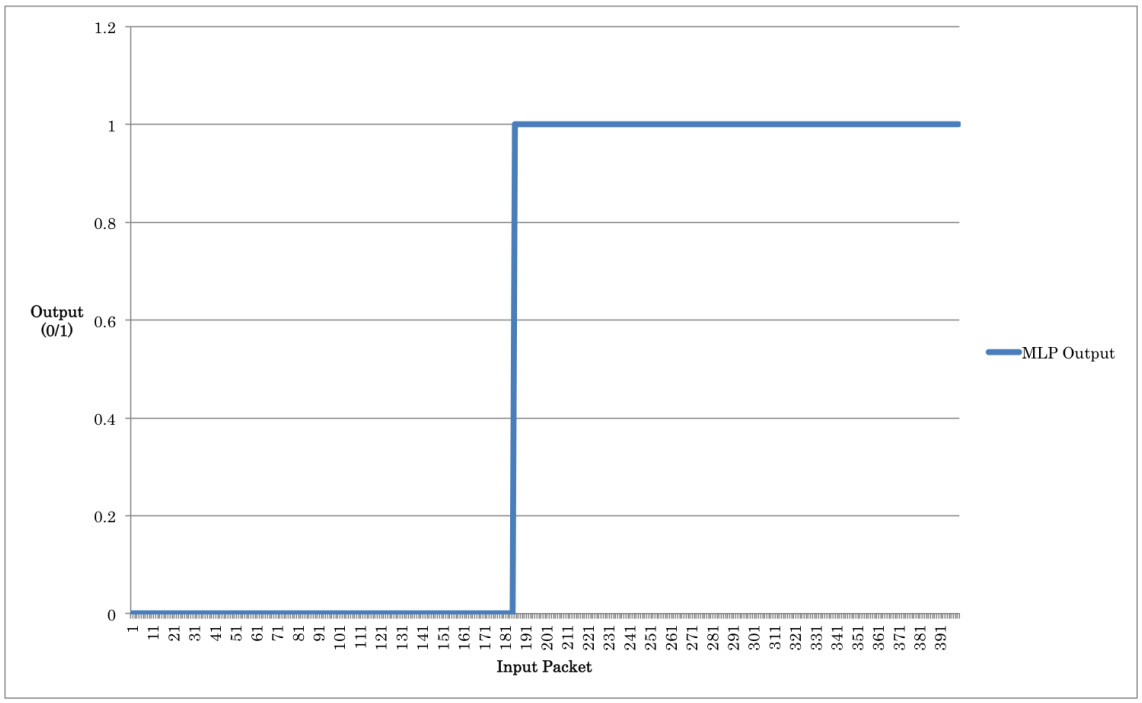

Figure 4.6: False Positives and False Negatives of MLP Neural Network.

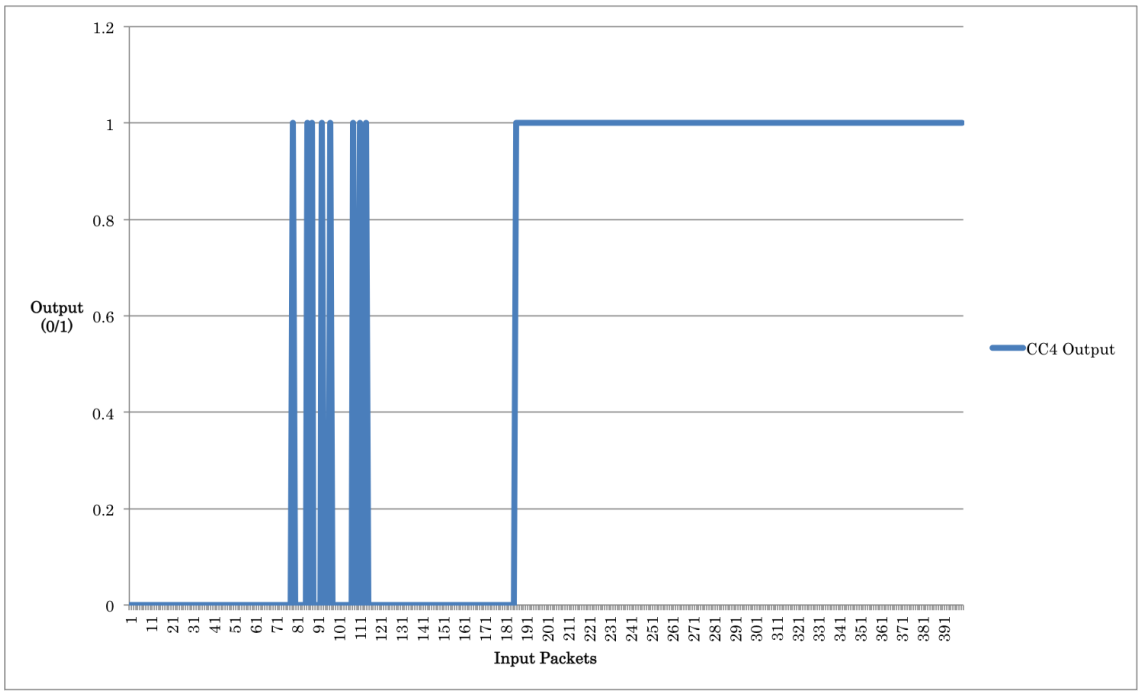

Figure 4.7: False Positives and False Negatives of CC4 Neural Network.

CHAPTER 5

CONCLUSION

5.1 Summary

A successful and efficient real time hybrid intrusion detection system has been developed that is a hybrid detection consisting of a anomaly detection for new attacks using CC4 neural network and misuse based attack detection using Multi layer perceptron implemention. The architecture of storm topology consisting of a spout which is an Input file reader sends the streams of packets to two bolts, one being a MLP Laucher and other a CC4 Launcher. The MLP Launcher obatins the closely appropriate weights from the trained signatures and sends them to its next respective bolt MLP Network. CC4 Launcher obtains the input's respective signatures from the training and sends the information to its respective bolt CC4 Network. MLP Network and CC4 Network produce their respective results which are both sent to another bolt called Post-Process. The Post-Process finally decides the output as either known attack with class or unknown attack or Normal action.

For this entire simulation, the ISCX Intrusion dataset which is a large dataset is used. Our work shows that the proposed model is applicable to current day network security environments.

5.2 Why the Proposed System Works?

The requirement for real-time stream processor for big data is fulfilled in our proposed model because it uses Apache Storm. Intrusion detection is performed instantaneously by this model, since apache storm is a distributed real-time cluster. In addition, the hybrid

detection mechanism in the proposed model provides instantaneous and accurate results because it uses the CC4 instantaneous neural network and the multi layered perceptron neural network. We can consider this model to be useful in current network environments, since it uses the latest intrusion dataset which contains current attacks, whereas the earlier models used KDD '99 which is old and do not contain latest attacks.

5.3 Future Work

- In order to be able to work in any network environment and to store large amount of signatures, we can take advantage of batch processing of big data using Apache Hadoop. Apache storm can be combined with Hadoop to get better results which supports the integrated data processing, combining real-time streaming and hadoop distributed file system.

- In order to test the accuracy of this model in real world, a network can be used which is able to introduce natural real time intrusions with numerous packets and diverse network scenarios.

- Since there is no limit to the number of neural networks, we can implement this model using different neural networks which requires less pre-processing unlike the CC4 neural network, which requires input data in unary format and conversion to unary format requires lot of steps.

REFERENCES

[1] Apache storm. `https://storm.incubator.apache.org/`. Last accessed 9 July 2014.

[2] Vijay Agneeswaran. *Big Data Analytics Beyond Hadoop: Real-Time Applications with Storm, Spark, and More Hadoop Alternatives*. Pearson FT Press, USA, 1st edition, 2014.

[3] Edward Amoroso. *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response*. Intrusion Net Books, 1999.

[4] Ravindra Balupari, Brett Tjaden, Shawn Ostermann, Marina Bykova, and Aaron Mitchell. Real-time system security. chapter Real-time Network-based Anomaly Intrusion Detection, pages 1–19. Nova Science Publishers, Inc., Commack, NY, USA, 2003.

[5] David Elson. Intrusion Detection, Theory and Practice. `http://www.securityfocus.com/`. Last accessed 15 June 2014.

[6] Vegard Engen, Jonathan Vincent, and Keith Phalp. Exploring discrepancies in findings obtained with the kdd cup '99 data set. *Intell. Data Anal.*, 15(2):251–276, April 2011.

[7] Jeremy Frank and Nsa Urp Mda-c. Artificial intelligence and intrusion detection: Current and future directions. In *In Proceedings of the 17th National Computer Security Conference*, 1994.

[8] John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. *Induction: Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, MA, USA, 1986.

[9] Marcelo Holtz, Bernardo David, and Rafael Timoteo. Building Scalable Distributed Intrusion Detection Systems Based on the MapReduce Framework. *Revista Telecommunications*, 2011.

[10] Devikrishna K and Ramakrishna B. An Artificial Neural Network based Intrusion Detection System and Classification of Attacks. *International Journal of Engineering Research and Applications*, 3(4), 2013.

[11] Subhash Kak. New algorithms for training feedforward neural networks. *Pattern Recognition Letters*, 15(3):295 – 298, 1994.

[12] Jonathan Leibiusky, Gabriel Eisbruch, and Dario Simonassi. *Getting Started with Storm*. O'Reilly Media, Inc., 2012.

[13] Zdravko Markov and Ingrid Russell. An introduction to the weka data mining system. *SIGCSE Bull.*, 38(3):367–368, June 2006.

[14] Nathan Marz. Storm: Distributed and fault-tolerant realtime computation. Emerging Technologies, 2012.

[15] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing (4th Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[16] Rohit Pillay. Instantaneous Intrusion Detection System. Master's thesis, Department of Computer Science, Oklahoma State University, 2010.

[17] Real-time Computing. Real-time computing — Wikipedia, the free encyclopedia, 2014. [Online; accessed 14 July 2014].

[18] Sumanth Reddy. Generalization and Efficient implementation of CC4 Neural Network. Master's thesis, Department of Computer Science, Oklahoma State University, 2008.

[19] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, November 1958.

[20] H. Sallay, A. Ammar, M. Ben Saad, and S. Bourouis. A real time adaptive intrusion detection alert classifier for high speed networks. In *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*, pages 73–80, Aug 2013.

[21] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2000.

[22] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.*, 31(3):357–374, May 2012.

[23] Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. The 8 requirements of real-time stream processing. *SIGMOD Rec.*, 34(4):42–47, December 2005.

VITA

Sesha Sai Goutam Mylavarapu

Candidate for the Degree of

Master of Science

Thesis: REAL TIME HYBRID INTRUSION DETECTION SYSTEM USING APACHE STORM

Major Field: Computer Science

Biographical:

Education:

Received the Bachelor of Technology degree from Jawaharlal Nehru Technological University, Hyderabad, India, 2012, in Computer Science and Engineering.

Completed the requirements for the Masters degree with a major in Computer Science at Oklahoma State University in May, 2015.