A

Major Project Report

on

ENHANCEMENT OF INTRUSION DETECTION SYSTEM USING CRISP CONTROLLER

Submitted in Partial Fulfillment of
the Requirements for the Degree
of

Bachelor of Engineering

in

Computer Engineering

to

North Maharashtra University, Jalgaon

Submitted by

Vrushali A. Patil
Punam A. Patil
Vivek V. Bonde
Yogesh D. Kate

Under the Guidance of

Mr. Satpal D. Rajput

DEPARTMENT OF COMPUTER ENGINEERING
SSBT's COLLEGE OF ENGINEERING AND TECHNOLOGY,
BAMBHORI, JALGAON - 425 001 (MS)

2014 - 2015

SSBT's COLLEGE OF ENGINEERING AND TECHNOLOGY, BAMBHORI, JALGAON - 425 001 (MS)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the major project entitled *Enhancement of Intrusion Detection System using Crisp Controller*, submitted by

Vrushali A. Patil
Punam A. Patil
Vivek V. Bonde
Yogesh D. Kate

in partial fulfillment of the degree of *Bachelor of Engineering* in *Computer Engineering* has been satisfactorily carried out under my guidance as per the requirement of North Maharashtra University, Jalgaon.

Date: April 10, 2015
Place: Jalgaon

Mr. Satpal D. Rajput
Guide

Prof. Dr. Girish K. Patnaik
Head

Prof. Dr. K. S. Wani
Principal

Acknowledgements

We would like to express our deep gratitude and sincere thanks to all who helped us to complete this project work successfully.
Our sincere thanks to principal Prof.Dr.K.S.Wani, SSBT COET for having provided us facilities to complete our project work.
Our deep gratitude goes to Prof.Dr.G.K.Patnaik, head of the department, for granting us opportunity to conduct this project work.
We are sincerely thankful to Ass.Prof. Satpal D. Rajput, project guide, for his valuable suggestions and guidance at the time of need.
Gratitude to ever working and ever supporting staff of the Computer Department.
Great thanks to our friends, our project associates and all those who helped directly or indirectly for completion of this project.

Vrushali A. Patil
Punam A. Patil
Vivek V. Bonde
Yogesh D. Kate

Contents

List of Tables

List of Figures

Abstract

With the rapid expansion of computer networks during the past decade, security has become a crucial issue for computer systems. The detection of attacks against computer networks is becoming a harder problem to solve in the field of Network security. Intrusion Detection System (IDS) is an essential mechanism to protect computer systems from many attacks. The major work of intrusion detection systems is used to detect the anomaly and new attackers in the networks, even still various false alarms are caused in order to neglect this necessary feature. As the transmission of data over the internet increases the need to protect connected system also increases. Existing system present an anomaly-based intrusion detection system with the help of Fuzzy controller. But to improve the system performance, we are using Crisp controller in our proposed system. Crisp controller is used to create a detection model in the training phase and update this model in the test phase respectively. After that, system user verifies these decisions and Crisp controller tunes detection model using system users feedbacks. To improve the accuracy of detect the anomaly in the system.

Chapter 1

Introduction

The major work of intrusion detection systems is used to detect the anomaly and new attackers in the networks. The detection of attacks against computer networks is becoming a harder problem to solve in the field of Network security. Intrusion detection as defined by the Sys-Admin, Audit, Networking, and Security (SANS) Institute is the art of detecting inappropriate, inaccurate, or anomalous activity. Today, intrusion detection is one of the high priority and challenging tasks for network administrators and security professionals. More sophisticated security tools mean that the attackers come up with newer and more advanced penetration methods to defeat the installed security systems. Thus, there is a need to safeguard the networks from known vulnerabilities and at the same time take steps to detect new and unseen, but possible, system abuses by developing more reliable and efficient Intrusion Detection Systems. Any Intrusion Detection System has some inherent requirements. Its prime purpose is to detect as many attacks as possible with minimum number of false alarms, i.e., the system must be accurate in detecting attacks. However, an accurate system that cannot handle large amount of network traffic and is slow in decision making will not fulfil the purpose of an Intrusion Detection System. We desire a system that detects most of the attacks, gives very few false alarms, copes with large amount of data, and is fast enough to make real-time decisions.

The Introduction is followed by the Section 1.1 describing the background of Intrusion Detection. Section 1.2 describes the motivation for this project, Section 1.3 describes problem definition, Section 1.4 describes scope of project and Section 1.5 describes the organisation of project. Finally last section describes the summary of this chapter.

1.1 Background

Intrusion detection started in around 1980s after the influential paper from Anderson. Intrusion Detection Systems are classified as network based, host based, or application based depending on their mode of deployment and data used for analysis. Additionally, Intrusion Detection Systems can also be classified as signature based or anomaly based depending upon the attack detection method. The signature-based systems are trained by extracting specific patterns (or signatures) from previously known attacks while the anomaly-based systems learn from the normal data collected when there is no anomalous activity[2]. Another approach for detecting intrusions is to consider both the normal and the known anomalous patterns for training a system and then performing classification on the test data. Such a system incorporates the advantages of both the signature-based and the anomaly-based systems and is known as the Hybrid System. Hybrid systems can be very efficient, subject to the classification method used, and can also be used to label unseen or new instances as they assign one of the known classes to every test instance. This is possible because during training the system learns features from all the classes. The only concern with the hybrid method is the availability of labelled data. However, data requirement is also a concern for the signature- and the anomaly-based systems as they require completely anomalous and attack free data, respectively, which are not easy to ensure. This Intrusion Detection System was used in MANET using Fuzzy controller[1].

1.2 Motivation

Until now no work has been done regarding Intrusion Detection System using Crisp controller. All the recent Intrusion Detection was developed using Fuzzy controller, even though the Crisp controller is more accurate than Fuzzy controller. So this motivated us to opt for Crisp controller to make the IDS more accurate and precise. Also in recent systems KDD data sets were used, but NSL proved to be more efficient, so NSL is preferred over KDD data sets.

1.3 Problem Definition

anomaly based Intrusion Detection Systems protect computer networks against novel attack. In existing Fuzzy controller based system input dataset iteratively depends on learner system. So number of iteration increases and performance of Intrusion detection rate degrades. However, by considering attacker feature of input data set minimizes the number of iterations required by the Learner System. In the proposed Crisp controller based system

only attacker feature of Input data set is checked with the learner System, means normal behaviour of input data set is neglected.

1.4 Scope

Intruder can hack into the system, modify data, corrupt data, change the permissions of data, copy and spread data without your consent, IDS can prevent the Intrusions and all the actions Intruder can do.

1.5 Organisation of the Report

The Organisation of Report is given below-

CHAPTER 1,titled *Introduction*, presents introduction of the system is explained in detail with the background, motivation, problem definition and scope of the system with the objective.
CHAPTER 2,titled *System Analysis*, presents system analysis with literature survey and Feasibility study of the proposed system. Also gives the risk analysis for the system with project scheduling and effort allocation.
CHAPTER 3,titled *System Requirement Specification*, presents system requirement specification which elaborates the requirements necessary for the development of the system like hardware-software requirements and functional-nonfunctional requirements.
CHAPTER 4,titled *System Design*, presents system design which contains system architecture along with the data flow diagrams and UML diagrams of the proposed system.
CHAPTER 5,titled *Implementation*, presents implementation which contains implementation details of the proposed system along with system development flow.
CHAPTER 6,titled *System Testing*, presents system testing which contains test analysis which include various test cases along with its results.
CHAPTER 7,titled *Result and Analysis*, presents snapshots of the important processing and experimental results of the proposed system.
CHAPTER 8,titled *Conclusion and Future Work*, concludes the dissertation and provides directions for further work in this area.

1.6 Summary

The chapter of Introduction is solely given to give a mere description of project before probing in to actual project. The Section 1.1 described the background of Intrusion Detection. Section 1.2 describe about motivation. Section 1.3 described the scope and Section 1.4 problem definition of the proposed system. The next chapter describes System Analysis which is important in software development process.

Chapter 2

System Analysis

System Analysis is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and strength. Once these things are satisfied, then next steps is to determine which operating system and language can be used for developing the tool. Once the developer start building the tool the developers need lot of external support. This support can be obtained from seniors, from books or from internet. Before building the system the above consideration are taken into account for developing the proposed system.
In this chapter Section 2.1 describes about existing system. The next section will describe the proposed system that is advantages of Crisp controller and data set. Section 2.3 will focus on the projects feasibility on the economical, operational and technical grounds. The Section 2.4 will describe the risks associated with project, because its important to know about the risks in beforehand starting the project. Scheduling is most important attribute of a project after designing, so last two sections will be about project scheduling and effort allocation. The project scheduling gives us the timeline of the project and effort allocation gives us the idea about efforts taken by the each member towards the project individually. Finally last section describes the summary of this chapter.

2.1 Literature Survey

Every system or computer in world is prone to attacks. These attacks are of different types and from different attackers. The types of attacks can be from other viruses, worms or malicious programs, they can easily be stopped using other programs or taking up some measures. The real challenge is to protect our system from attackers. These attackers can steal data, harm system, create nuisance in programs, creates noise. The attackers are called as Intruders. So a system or network must be able to prevent attack from Intruders, recognise them, categories them and block them if they try to sneak into the system or network[1].
The current Intruder Detector System (IDS) operates on Fuzzy Logic or Fuzzy Control.

Fuzzy controller operates over a range of values from 0-1. It has provision to operate over the decimal values too. It operates over fuzzy data. The problem with the Fuzzy controller is that its vague and non-linear. It creates too much false alarms[6]. Also, it uses KDD data sets for operation, which duplicates the data. Suppose if we feed some data or behaviour to IDS for detection and prevention from Intruder, them that data sets entered multiplies itself. Due to this issue of storage arises. Suppose we entered 2000 behaviours then it will check Naive system for 2000 conditions, but next time it will multiply the set and the system would be checked for 4000 conditions, wasting time. So a Crisp controller NSL data sets are to be used in our project, the reason behind it will be explained in next section.

2.2 Proposed System

As we saw in last section that IDS with Fuzzy controller and KDD data set has so many issues, so we propose a system which use Crisp controller and NSL data sets. crisp logic is very appropriate for using on intrusion detection. In proposed system, existing fuzzy control system and KDD data set will be replaced by proposed crisp controlled system and NSL data set in which input of current network traffic is fed to the Crisp controller. System checked the behaviour of malicious attack and according to the training given to the system it removes the attack. Due to crisp logic it only tracks the true behaviour and neglect negative behaviour so performance of system is improved by proposed crisp logic.
Other approaches for detecting intrusion include the use of autonomous and probabilistic agents for intrusion detection. These methods are generally aimed at developing a distributed Intrusion Detection System. To overcome the weakness of a single Intrusion Detection System, a number of frameworks have been proposed, which describe the collaborative use of network-based and host based systems. Systems that employ both signatures based and behaviour based techniques are discussed.

The proposed system is expected to work in 5 stages:
1. Training
2. Pre-processing
3. Characterising system behaviour
4. Detection process
5. Generating Intrusion status report

In first stage training will be provided to server as per data set, its nothing but only feeding server with behaviour to be checked. Second stage is nothing but pre-processing, behaviours will be just categorised in this stage. Third stage will be characterising the system for Intruder or normal based on its behaviour. The first three stages will be called as Training

phase as the tasks of these stages are just to train the system and categorised data sets. This training is done by IDS engine with the help of Naive Bayes algorithm. The last two stages will be called as Testing or Detecting Intruder. Fourth stage will be detection process in which four steps will be carried out and the last stage will be generating Intrusion status report.

The data analysed by the Intrusion Detection System for classification often has a number of features that are highly correlated and complex relationships exist between them. When classifying network connections as either normal or as attack, a system may consider features such as logged in and number of file creations. When these features are analysed individually, they do not provide any information that can aid in detecting attacks. However, when these features are analysed together, they can provide meaningful information, which can be helpful for the classification task[1][2][3].

2.3 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Feasibility study is a process to check possibilities of system development. It is a method to check various different requirements and availability of financial and technical resources. Before starting the process various parameters must be checked like estimated finance is there or not? The man power to operate the system is there or not? The man power is trained or not?

All the above conditions must be satisfied to start the project. This is why in depth analysis of feasibility is carried out.

Three key considerations involved in the feasibility analysis are given follow.

2.3.1 Economical Feasibility

The proposed system doesn't require any additional hardware. The standard devices i.e. a laptop are the prerequisites. It runs on Windows operating system. And Implemented in Java 1.7 which is again available in free of cost in market. So the implementation will add no hardware or software cost, due to all above reasons, will result in low market price.

2.3.2 Operational Feasibility

Once the system is designed there must be trained and expert operator. If they are not trained they should give training according to the needs of the system. From the users perspective our system fully operational feasible as it just requires some knowledge of computer. Proposed system has high operational feasibility. It provides high level of security to the network communication by detecting Intruder in the network.

2.3.3 Technical Feasibility

The proposed system is technically feasible in aspect that technology used to develop the system is simple and adopted all over technical world. Proposed system is being developed by using well known and popular software tools and basic hardware. To develop this system or to execute it very few basic hardware and software components are required. The software components required for development are Java 1.7, Netbeans IDE. We are using Netbeans here because it is free and open source because of its powerful GUI builder, because of its Java Mobility Support and its profiling and debugging tools. Being developed in Java, this project will be platform independent. The hardware required to develop this project are simply basic computers and a basic device to connect computers in LAN while executing it.

2.4 Risk Analysis

Risk Analysis and management are a series of steps that help a software team to understand and manage uncertainty. The goal of risk assessment is to prioritize the risks so that attention and resources can be focused on the more risky items. Risk identification is the first step in risk assessment, which identifies all the different risks for a particular project. The Problems or risks that we commonly faced are listed below:

1.Estimation and Scheduling: The unique nature of individual software projects creates problems for developers in estimating and scheduling development time. We should refer existing project experience to overcome this problem.

2.Sudden growth in requirements: There can be a sudden growth in resources that we have not thought earlier while project planning. This sudden growth can also lead in being late for project completion.

3.Breakdown of specification: At the initial stage of integration or coding, we felt that

requirements and specifications are incomplete or insufficient. As coding got progressed, requirement of specification was fulfilled. These risks are project-dependent and identifying them is an exercise in envisioning what can go wrong. Methods that can aid risk identification include checklists of possible risks, surveys, meetings and brainstorming, and reviews of plans, processes, and work products.

2.5 Project Scheduling

Software project scheduling is an activity that distributes estimated effort across the planed project duration by allocating the effort to specific software engineering task. In this phase we are identifying all major software engineering activities and the product function to which they are applying.

As we have selected the linear sequential model for developing our project we divide the work according to the phases of this model. As we are four partners working on this project and having 8-9 months, we scheduled the project according to it. If the project has been developed according to the schedule, the project schedule defines the task and milestones that must be tracked and controlled as the project proceed.

Following table gives project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a cross; the position of cross reflects the duration of the activity.

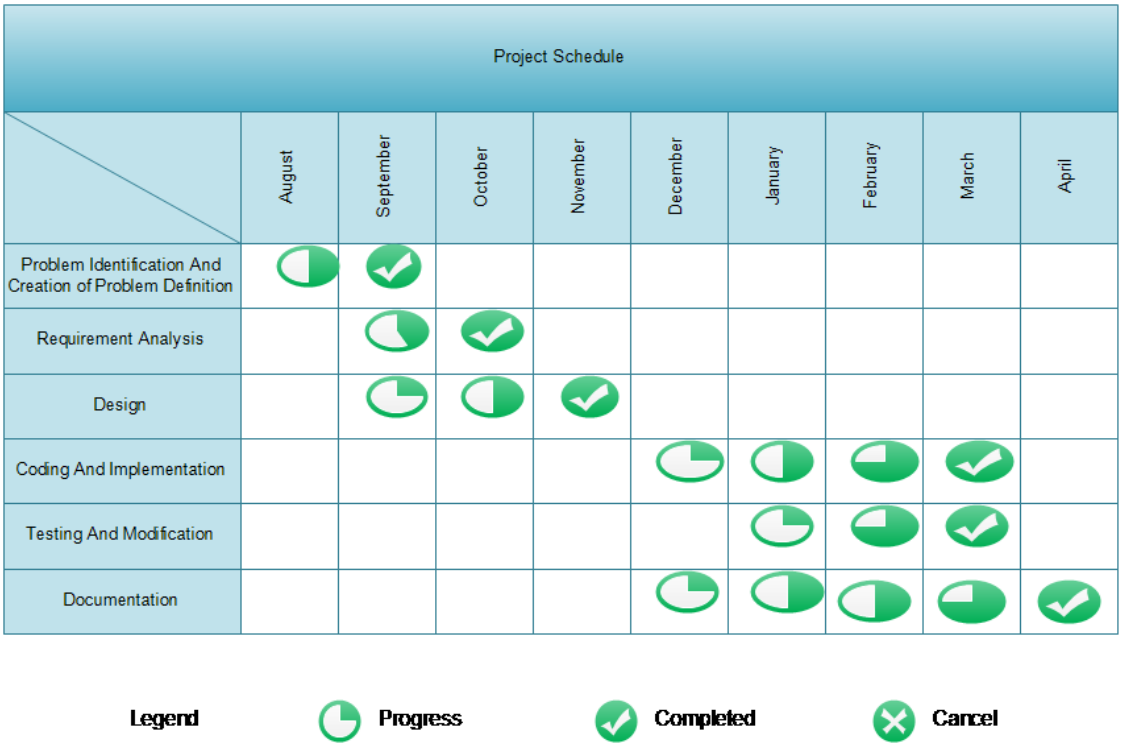

Figure 2.1: Gantt Chart

This table shows the row and column wise allocation for each phase in software developing process. We have spent most of our time for System Engineering or for Requirement Analysis. We considered system engineering a base for developing project and thought if this phase is gone correctly all other steps will be easier.

2.6 Effort Allocation

Activities	Vrushali Patil	Yogesh Kate	Vivek Bonde	Punam Patil
Project Planning	35%	25%	25%	15%
Literature Survey	30%	40%	20%	10%
Requirement Specification	25%	30%	30%	15%
Designing	35%	25%	20%	20%
Coding	30%	25%	35%	10%
Testing	20%	30%	40%	10%
Modification	30%	25%	30%	15%
Documentation	40%	25%	25%	10%

Table 2.1: Effort Allocation

2.7 Summary

In this chapter we took a detailed look on the various aspects of our project summarising system analysis, Literature survey described in Section 2.1, section 2.2 describes proposed system, various feasibility aspects in Section 2.3, and Section 2.4 and Section 2.5 describes the risk analysis and project scheduling respectively. The analysis suggest that proposed system will be more efficient than existing system and also its feasible to develop the proposed system. The next chapter describes System requirement specification.

Chapter 3

System Requirement Specification

In Software Requirement Specification is description of a software system to be developed, proposed system's hardware, software requirements with some functional and non functional requirement has been discussed.
The respected chapter describes software requirement specification. Section 3.1 describes the hardware requirements of the system. software requirements are described in Section3.2. Section 3.3 describes the functional requirements of the proposed system. Also, non- functional requirements are described in Section 3.4. the last section gives the summary of this chapter.

3.1 Hardware Requirements

1. System : I3, 2.4 Ghz
2. Hard Disk : 40 GB
3. Floppy Drive : 1.44 Mb
4. Monitor : 15 VGA Colour
7. Ram : 256 Mb

3.2 Software Requirements

1. Windows Professional
2. Java
3. RMI
4. JDBC
5. Swing
6. Netbeans IDE 7.4

3.3 Functional Requirements

As the network-computing environment increases in complexity, so do the functional requirements of IDSs. Common functional requirements of an IDS being deployed in current or near-term operational computing environments include the following:

1.The IDS must continuously monitor and report intrusions.

2.The IDS must supply enough information to repair the system, determine the extent of damage, and establish responsibility for the intrusion.

3. The IDS should be modular and configurable as each host and network segment will require their own tests and these tests will need to be continuously upgraded and eventually replaced with new tests.

4. Since the IDS is assigned the critical role of monitoring the security state of the network, the IDS itself is a primary target of attack. The IDS must be able to operate in a hostile computing environment and exhibit a high degree of fault-tolerance and allow for graceful degradation.

5. The IDS should be adaptive to network topology and configuration changes as computing elements are dynamically added and removed from the network.

6. anomaly detection systems should have a very low false alarm rate. Given the projected increase in network connectivity and traffic, simply decreasing the percentage of overall false alarms may not be sufficient as their absolute number may continue to rise.

7. The IDS should be able to learn from past experiences and improve its detection capabilities over time. A self-tuning ID will be able to learning from false alarms with the guidance of system administrators and eventually on its own.

8. The IDS shoul+d be able to be easily and frequently updated with attack signatures as new security advisories and security patches become available and new vulnerabilities and attacks are discovered.

3.4 Non-Functional Requirements

1. Understandability, usability, modifiability, inter-operability, reliability, portability, maintainability, scalability.

2. Configurability, customizable, adaptability, variability, volatility, traceability.

3.security, simplicity, clarity, ubiquity, integrity, modularity, nomadicity.

4.user-friendliness, robustness, timeliness, responsiveness, correctness, completeness, conciseness, cohesiveness.

3.5 Summary

This chapter describes the System Requirement Specification of the proposed system. Section 3.1 describes the Hardware Requirements which specify the hardware required for the system and Section 3.2 Software Requirements like Java software, netbeans etc... Also the Section 3.3 and 3.4 describes the functional and Non-Functional Requirements of the system. The next chapter gives details about system design.

Chapter 4

System Design

System Design is a meaningful engineering representation of something that is to be built. It can be traced to a customers requirements and at the same time assessed for quality against a set of predefined criteria for good design. In the software engineering context, design focuses on four major areas of concern: architecture, interfaces, and components.
In Section 4.1 shows system architecture, Section 4.2 gives data flow diagram and UML diagram are discussed in Section 4.2. In the last section, summary of this chapter is given.

4.1 System Architecture

The chapter of system architecture is important because it gives us the idea about the system which is planned. The various diagrams such as system architecture, data flow diagrams and UML diagrams serves the purpose.

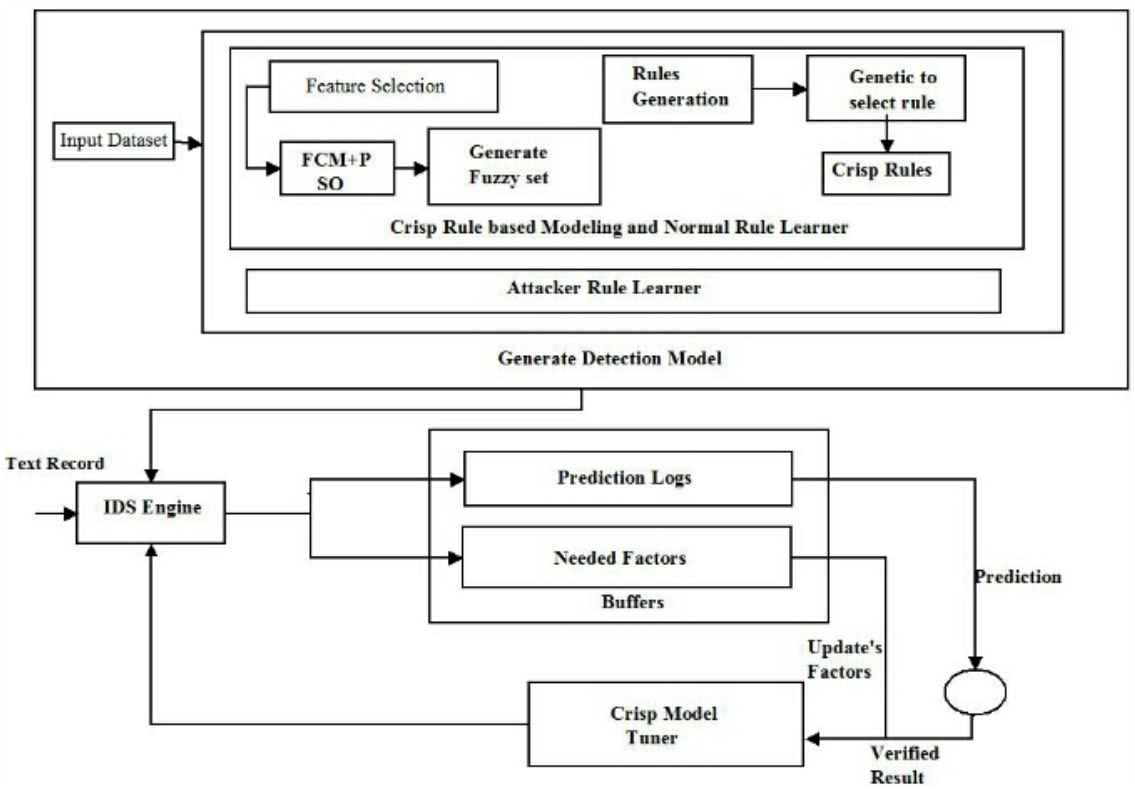

Figure 4.1: Architecture Diagram for anomaly Detection

4.2 Data Flow Diagram

Data Flow Diagram(Level 0) :

Figure 4.2: Data Flow Diagram- Level 0

Data Flow Diagram(Level 1) :

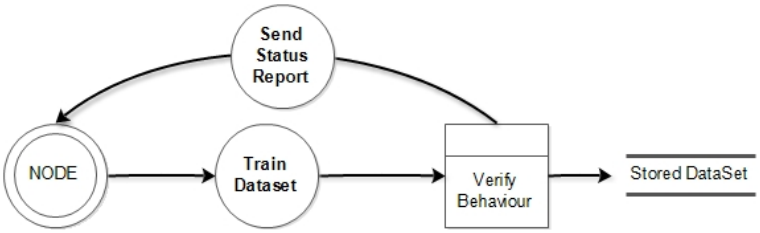

Figure 4.3: Data Flow Diagram- Level 1

Data Flow Diagram(Level 2) :

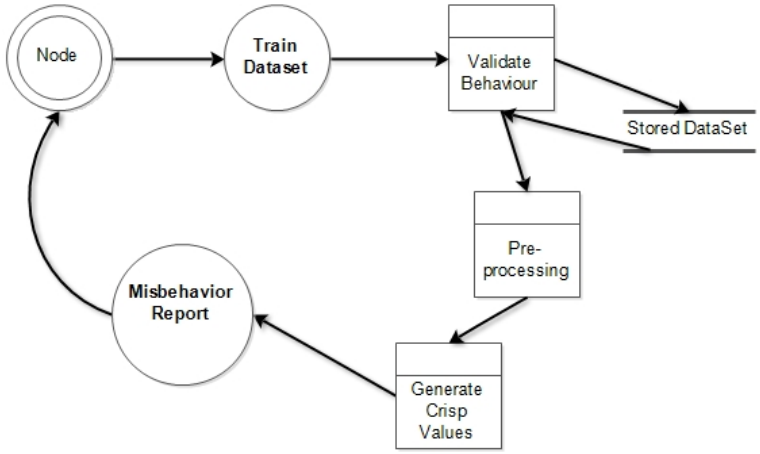

Figure 4.4: Data Flow Diagram- Level 2

Data Flow Diagram(Level 3) :

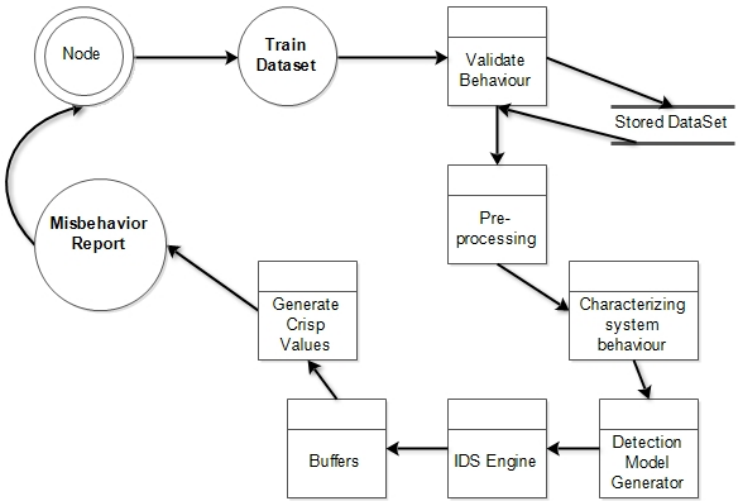

Figure 4.5: Data Flow Diagram- Level 3

4.3 UML Diagrams

4.3.1 Use Case Diagram

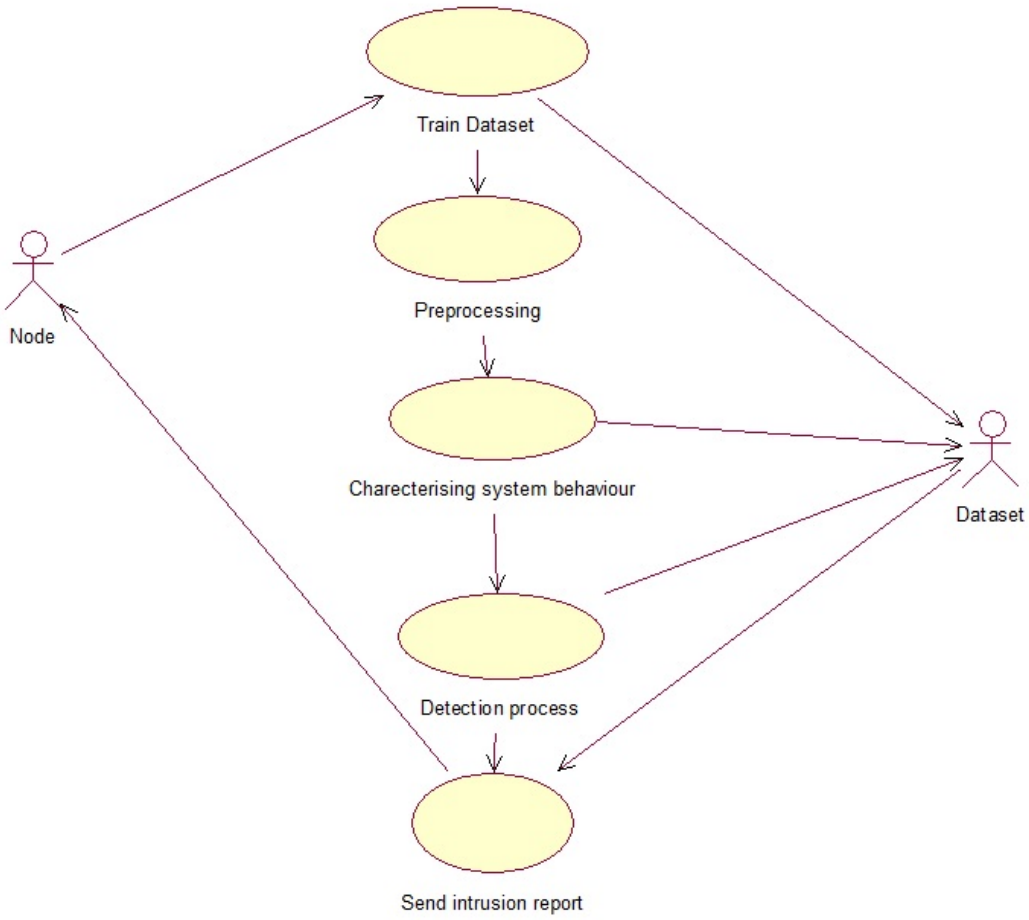

Figure 4.6: Use Case Diagram

4.3.2 Class Diagram

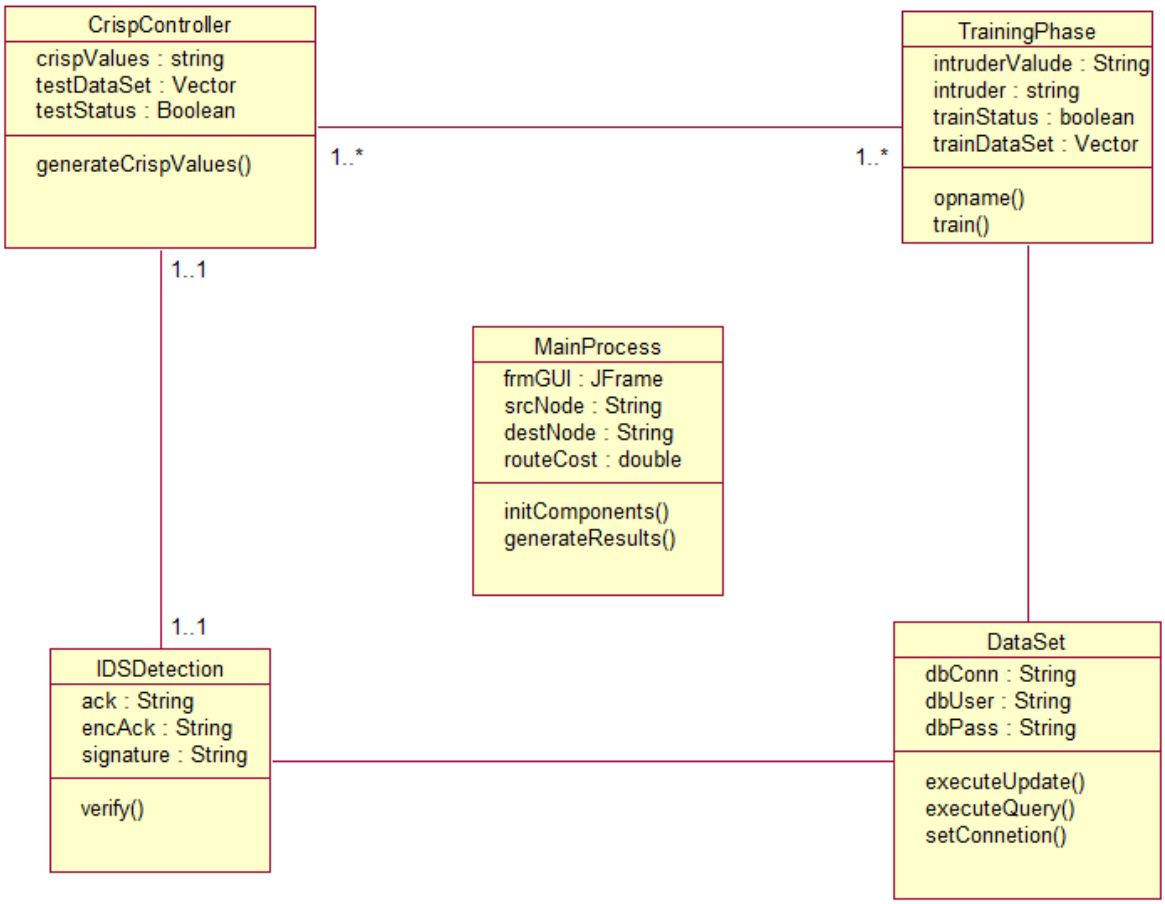

Figure 4.7: Class Diagram

4.3.3 Sequence Diagram

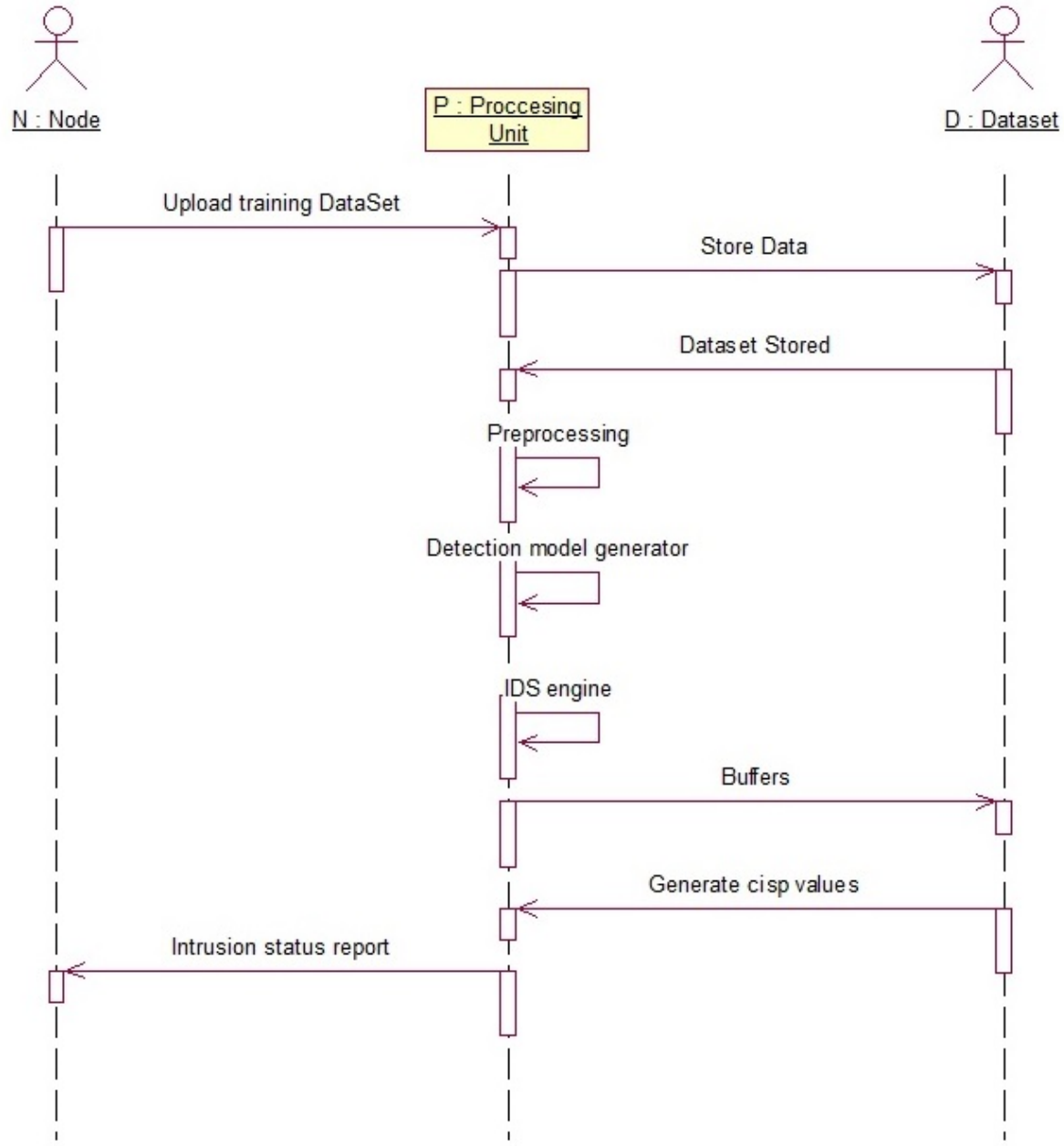

Figure 4.8: Sequence Diagram

4.3.4 Activity Diagram

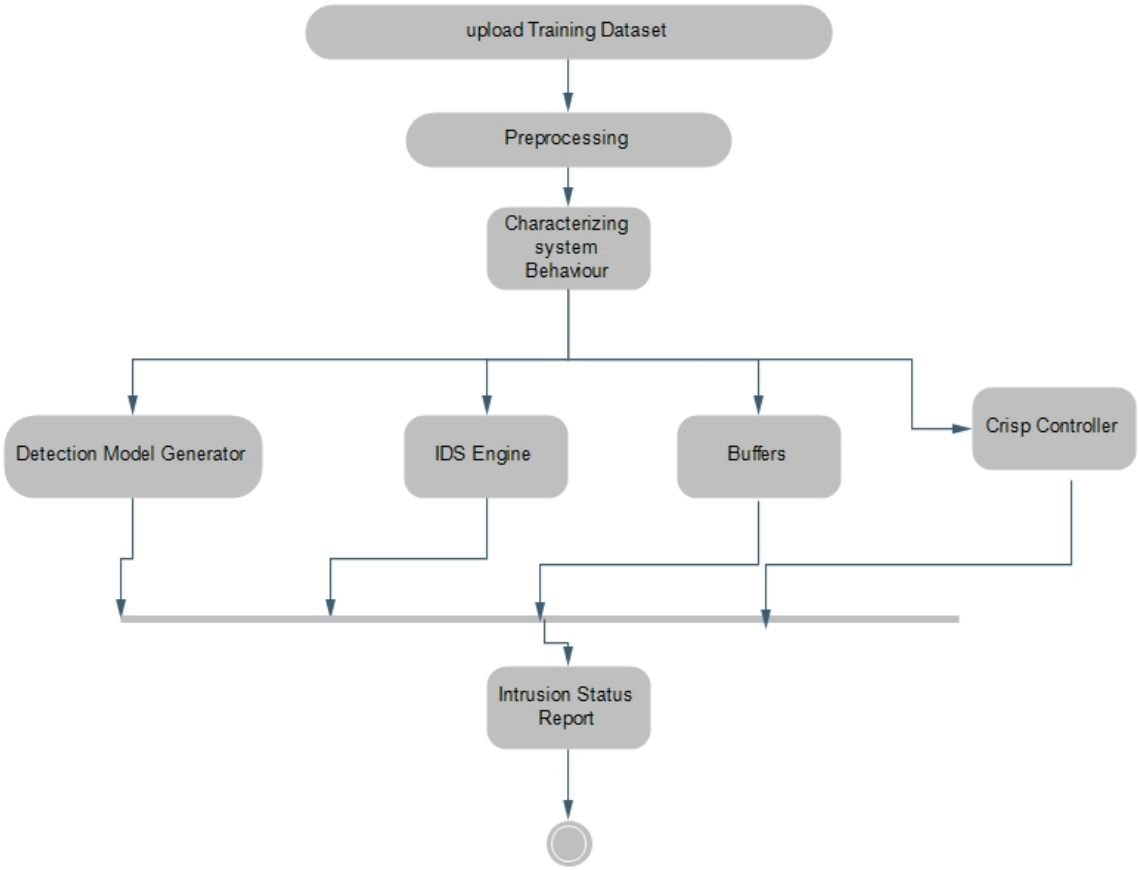

Figure 4.9: Activity Diagram

4.3.5 Component Diagram

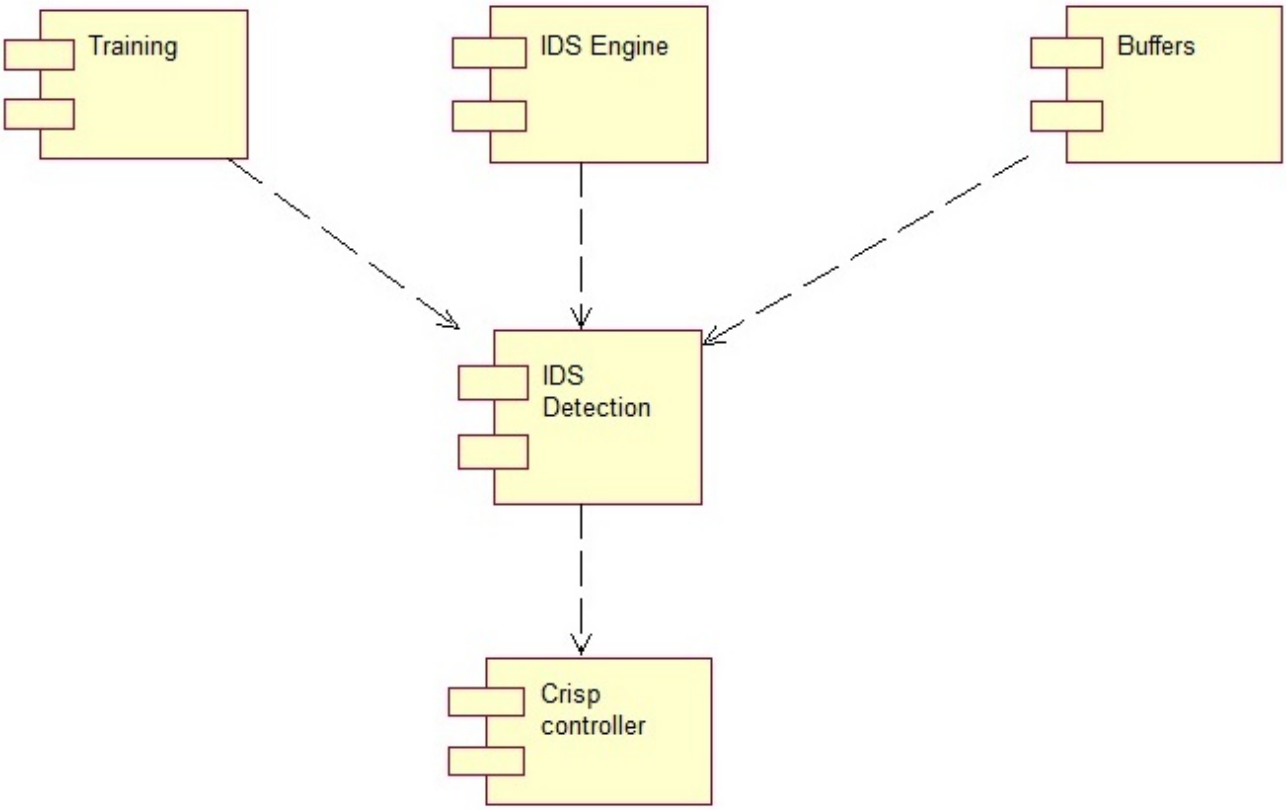

Figure 4.10: Component Diagram

4.3.6 Deployment Diagram

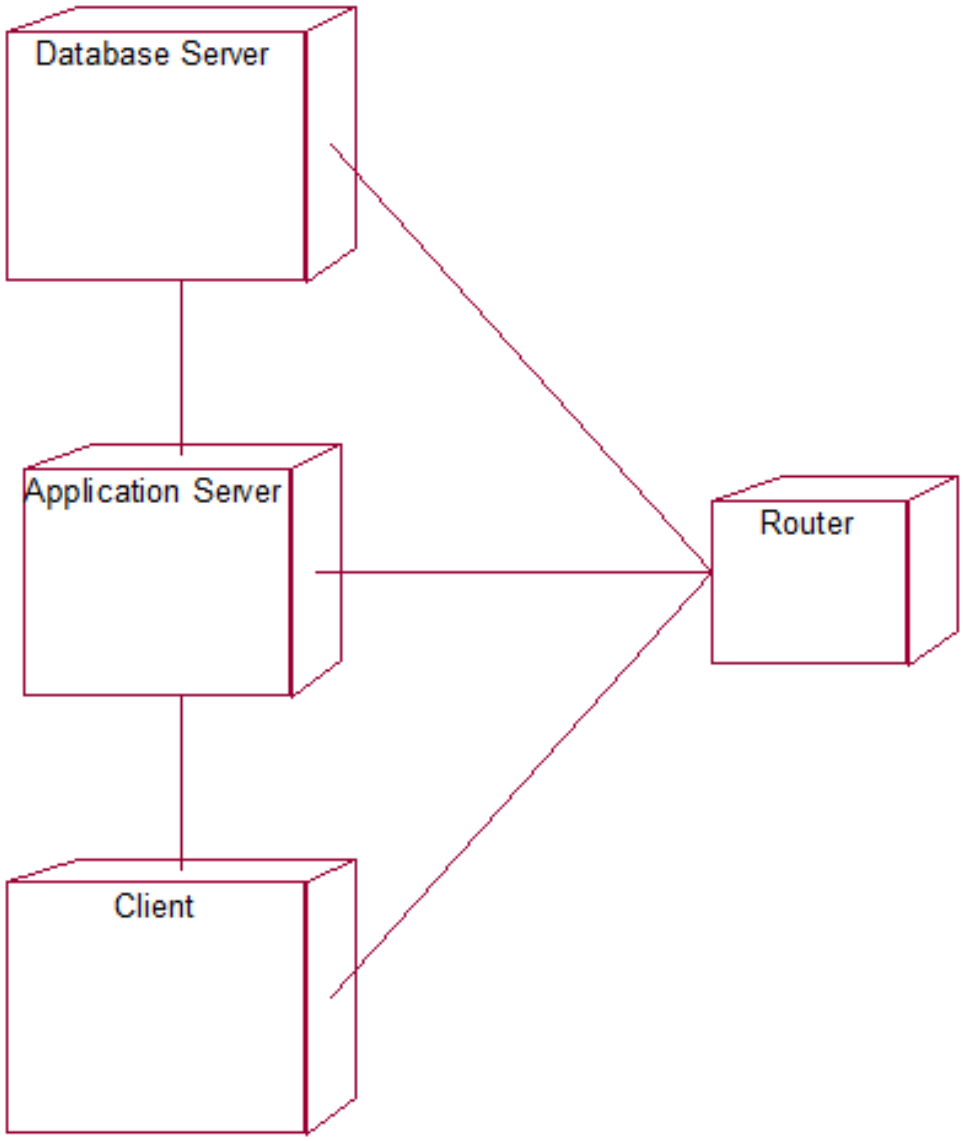

Figure 4.11: Deployment Diagram

4.4 Summary

The first section gives information about system architecture. The second is data flow diagrams. Then last section comprises of UML diagrams, that are Use case, Class, sequence, Activity, Component and deployment. The next chapter describes implementation of the system.

Chapter 5

Implementation

This chapter covers the various Implementation Details of the project including Implementation Environment and Flow of system in development.
Section 5.1 decribes the Implementation Details. Implementation Environment are explained in Section 5.2. Section 5.3 explains the Flow of system in development. In the last section, summary of the this chapter is given.

5.1 Implementation Details

To overcome the drawback of existing system, we are developing an Intrusion Detection System using Crisp controller which produces exact and better results as compared to systems based on Fuzzy Logic. It also has the capacity of self-updating. In proposed IDS existing Fuzzy Logic is replaced by Crisp controller to detect attackers based on their behaviour and update proposed KDD NSL database to remember the Intruder behaviour for next testing. System checked the behaviour of malicious attack and according to the training given to the system it removes the attacks. Due to use of Crisp controller, it only tracks the true behaviour and hence performance of the system increases.

Two main reasons for introducing crisp logic for intrusion detection are Intrusion detection involves many quantitative features. To categorize these quantitative features in order to establish high-level patterns, Crisp theory provides a reasonable and efficient way. And the Security itself is crisp. For quantitative features, there is no sharp delineation between normal operations and anomalies. Crisp episode rules allow one to create the high-level sequential patterns representing normal behaviour. With crisp spaces, crisp logic allows an object to belong to different classes at the same time. This concept is helpful when the difference between classes is not well defined. This is the case in the intrusion detection task, where the differences between the normal and abnormal classes are not well defined. Thus the intrusion detection problem (IDP) is a two-class classification problem: the goal is

to classify patterns of the system behaviour in two categories (normal and abnormal), using patterns of known attacks, which belongs to the abnormal class, and patterns of normal behaviour. In crisp logic, crisp sets define the linguistic notions, and membership functions define the truth-value of such linguistic expressions. Intrusion detection has been done in different phases which include training to the system, pre-processing and characterization of data sets, detecting Intruders in the network with the status report for recognizing Intruder in the network[3].

5.2 Implementation Environment

The project is windows based hence it requires any Windows distribution. In this project, the Windows 7 operating system is used. Project code is implemented in Java language and the software development platform used is Netbeans.

5.2.1 Why Java

Java is currently a leading programming language used by developers all over the world for developing new applications. Estimates suggest that in 2012, over 10 million users would be using Java through millions of devices capable of operating the Java Runtime Environment (JRE), which is necessary for running Java script on a device. The number of users engaged in using Java for development of solutions has continued to grow since the language was first introduced by Sun Microsystems in the year 1995.

The primary advantage of Java application development is that it is free and its syntax bears resemblance to various C-based programming languages, making it easier for developers to understand and implement. The language also features an extensive standard class library, most of which is quite well written. The Java developer kit includes tools for automatic memory management, while simultaneously providing a sturdy platform for behavioral transference from one address space to another. The language also provides superior performance as compared to many available languages, along with better portability and a simplified syntax as compared to the C++ language. Other advantages of Java software development include the non-committee driven language design, presence of the No Explicit Pointers tool and the availability of Explicit Interfaces to help the developer write error-free coding. Java application development also facilitates the availability of comprehensive documentation as well as numerous freely available sources for third-party libraries and codes. Developers also have a wide range of Java IDE choices, while remaining unaffected by the problem of a Fragile Binary Interface. Considering these benefits, it is no surprise that Java

assumed supreme popularity especially during the initial years following its introduction.

5.2.2 WEKA

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

5.2.3 NSL Datasets

The NSL-KDD data set suggested solving some of the inherent problems of the KDDCUP'99 data set. KDDCUP 99 is the mostly widely used data set for anomaly detection. But Tavallae et al conducted a statistical analysis on this data set and found two Important issues that greatly affected the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches. To solve these issues, they proposed a new data set, NSL-KDD, which consists of selected records of the complete KDD data set.

5.2.4 Naive Bayes Classifier

A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem (from Bayesian statistics) with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model". In simple terms, a Naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. Depending on the precise nature of the probability model, Naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for Naive Bayes models uses the method of maximum likelihood; in other words, one can work with the Naive Bayes model without believing in Bayesian probability or using any Bayesian methods. An advantage of the Naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix[4].

5.3 Flow of system development

The system development flow is given follow-

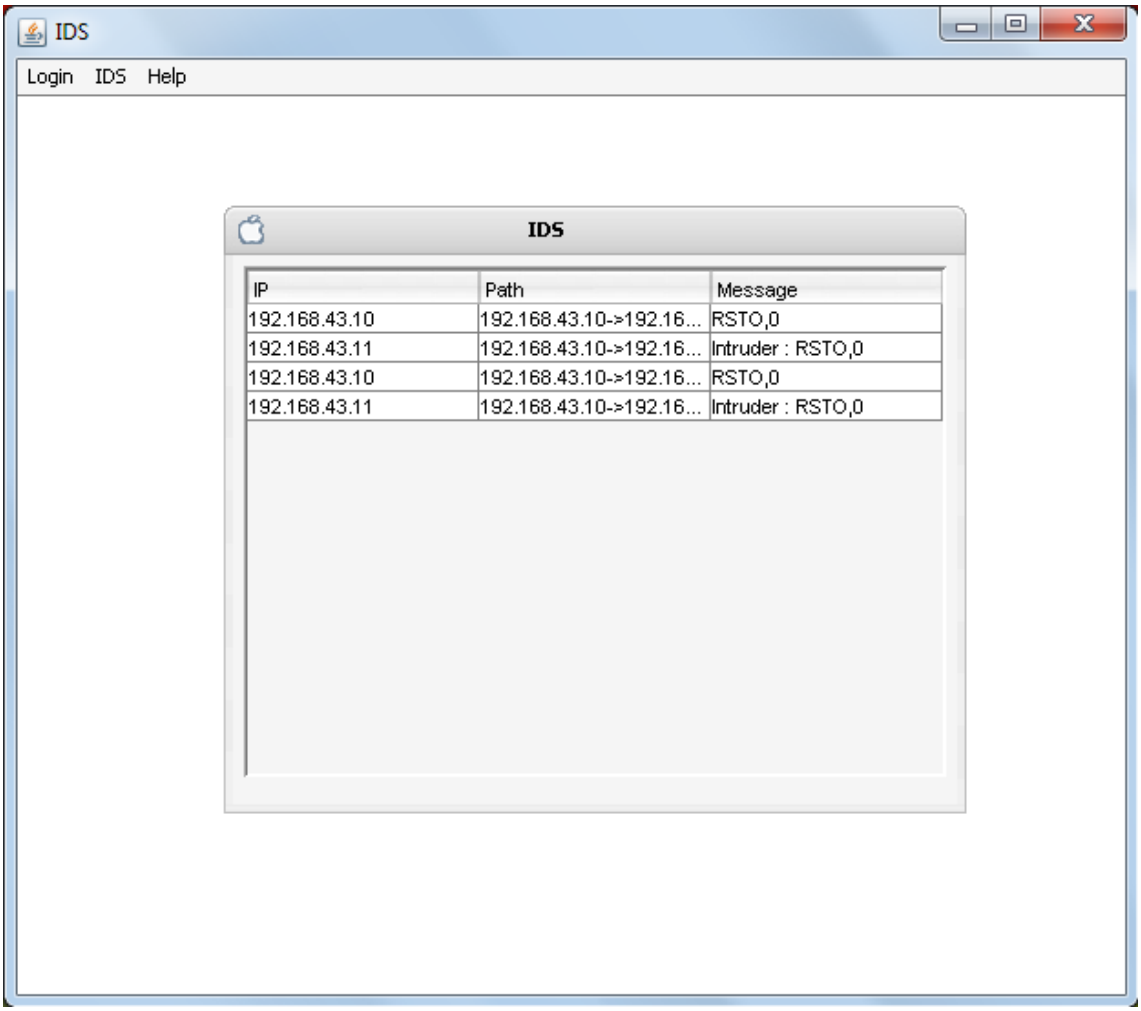

Figure 5.1: IDS using Crisp controller

5.3.1 Training Phase:

IDS's are trained and data sets are categorized. The training will be provided to server as per data set, its nothing but only feeding server with behaviour to be checked. It is nothing but learning phase for the system. Temporary data set stores NSL data set which is multidimensional array with output results. NSL set consist of features of data with their output values which further use for the classification and testing of the records. The figure 5.1 shows the IDS system using Crisp controller.

5.3.2 Testing Phase:

Depending on the behaviour of the system, Intruder (anomaly) system or normal system is characterized. Then detection process starts with detection model generator which generates the framework for the detection process. The IDS engine employs the detection model to

classify test samples. It monitors the work flow of the system. Prediction logs and compatibility of test samples with each rule are buffered. Buffers maintain all the information about Intruder and normal systems.
Given data sets are classified using Naive Bayes algorithm. Naive Bayes is probability based algorithm. The classified datasets gives standard mean and deviation values after testing phase. Where, mean is average of standard values of datasets and deviation is mean deviation of datasets. crisp logic is applied on the probabilistic values generated by Bayesian classifier to get resultant value for that particular record is Intruder or not. This procedure can be done in following steps-

1. Calculate individual Probability of every feature.
2. Calculate probability by grouping the features.
3. Calculate Final Probability of complete record finding values of anomaly System and Normal System.
4. Generate Output value from Maximum Probability value from step-3.

5.4 Summary

In this chapter, Section 5.1 explains Implementation details of the project like Implementation Environment in Section 5.2 and Flow of System development in Section 5.3. The next chapter describes the System Testing, How it is implemented and also the Test Cases and Test Results.

Chapter 6

System Testing

This chapter covers the System Testing to explain how the system is tested. It also includes the various Test Cases and their Results obtain during experimenting proposed system. Section 6.1 describes How to implement testing. Test cases and Test Results are explained in Section 6.2. The last section gives summary.

6.1 How to Implement Testing

Tests are the individual tests specified in a test plan document. Each test is typically described by

- An initial system state.
- A set of actions to be performed.
- The expected results of the test.

6.1.1 Testing As a Continuous Process

All testing follows a pre-planned process, which is agreed to. All tests consider not only a nominal system condition but also address anomalous and recovery aspects of the system. The system is tested in a stressed environment, nominally in excess of 150 percent of its rated capacities. All test products (test cases, data, tools, configuration, and criteria) are documented in a software description document. Every test shall be described in traceable procedures and have pass-fail criteria included.

6.1.2 Implementation

Test cases are planned in accordance to the test process and documented with detailed test descriptions. These test cases use cases based on projected operational mission scenarios. The testing process also includes stress / load testing for stability purpose (i.e., at 95% CPU use, system stability is still guaranteed). The test process thoroughly tests the interfaces and modules. Software testing includes a traceable white box testing, black box testing and other test processes verifying implemented software against design documentation and requirements specified.

6.1.3 Types of Testing Performed

In this section we perform various types of testing needed for the project. Testing allow to examine the project and its module by different ways, detecting the defects and remove them. Testing increases the efficiency of the product.

- *White Box Testing*

A level of white box test coverage is specified that is appropriate for the software being tested. The white box and other testing uses automated tools to instrument the software to measure test coverage. The program code is examined for defects. It based on design and implementation structures. It deals with internal logic and structure of the code. sometimes it needs changes in the program code, correcting defects, changes in GUI and etc...

- *Black Box Testing*

A black box test of integration builds includes functional, interface, error recovery, stress and out-of-bounds input testing. All black box software tests are traced to control requirements. In addition to static requirements, a black box of a fully integrated system against scenario sequences of events is designed to model field operation. Performance testing for systems is integrated as an integral part of the black box test process.

- *Other Testing*

Unit test comprises of a set tests performed by an individual program prior to the integration of the unit into large system. A program unit is usually the smallest free functioning part of the whole system. Module unit testing should be as exhaustive as possible to ensure that each representation handled by each module has been tested. All the units that makeup the system must be tested independently to ensure that they work as required. During unit

testing some errors were raised and all of them were recited and handled well. The result was quiet satisfactory and it worked well.

Integration testing is a system technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design. Bottom-up integration is the traditional strategy used to integrate the components of a software system into functioning whole. Bottom up integration consists of unit test followed by testing of the entire system. A sub-system consists of several modules that communicated with other defined interface . The system was done the integration testing. All the modules were tested for their compatibility with other modules .They test was almost successful. All the modules coexisted very well, with almost no bugs. All the modules were encapsulated very well so as to not hamper the execution of other modules.

After validation testing, software is completely assembled as a package, interfacing errors that have been uncovered and corrected and the final series of software test; the validation test begins. Steps taken during software design and testing can greatly improve the probability of successful integration in the larger system. System testing is actually a series of different tests whose primary purpose is to fully exercise the compute based system.

6.2 Test Cases and Test Results

Testing is a process of executing a program with the interest of Finding an error. A good test is one that has high probability of finding the yet undiscovered error. Testing should systematically uncover different classes of errors in a minimum amount of time with minimum amount of efforts. The test cases are tested manually and the are shown in tabular form as below.

TEST CASE ID	TEST CASE NAME	EXPECTED RESULT	RESULT	TEST RESULT PASS OR FAIL
Test 1	Login user	Logged in	User logged in successfully	Pass
Test 2	Broadcast message	Message broadcast	Message received	Pass
Test 3	Train dataset	Training Completed	Training Completed	Pass
Test 4	Testing	Generate test results	Generate test results	Pass
Test 5	Broadcast Intruder message	Intruder detected	Intruder detected	Pass
Test 6	Signature Keys match	Message broadcast	Message broadcast	Pass
Test 7	Signature keys not match	Intruder detected	Intruder detected	Pass

Table 6.1: Test cases

6.3 Summary

This chapter explains How to implement testing, Test cases and their Results. The next chapter describes the Result and Analysis for the project. In next chapter Results of the system is given.

Chapter 7

Results and Analysis

The chapter gives sample snapshots of important processing. And the experimental results of the proposed system which gives comparison between existing Fuzzy Logic and proposed crisp logic using intrusion detection system.

7.1 Sample Snapshot of important processing

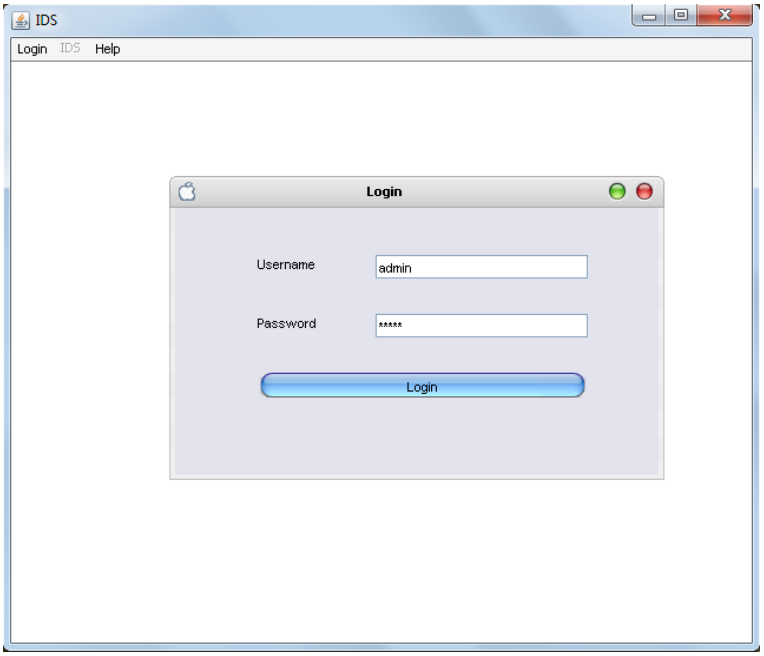

Figure 7.1: login Window

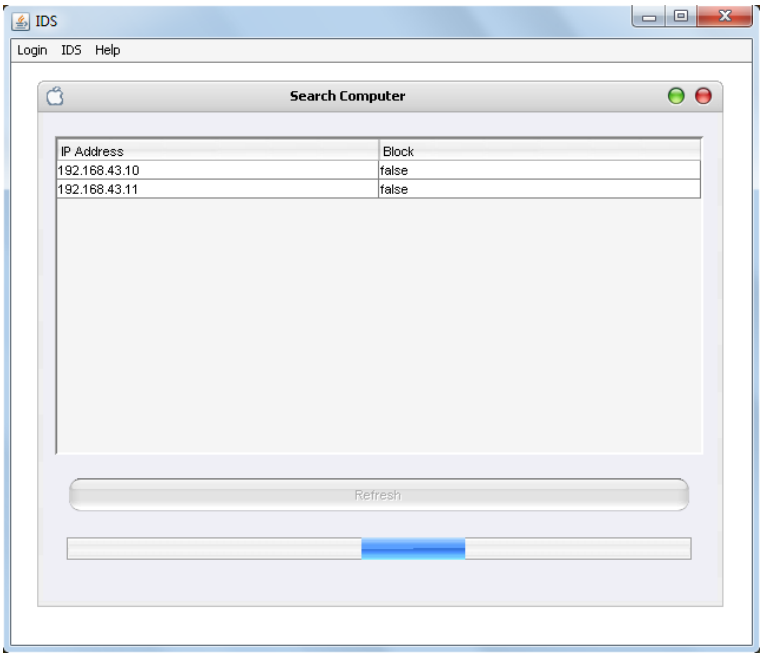

Figure 7.2: Systems in Network

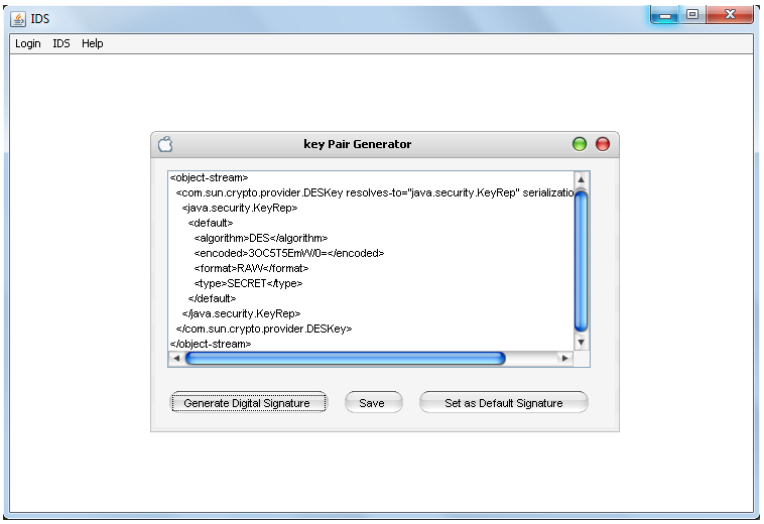

Figure 7.3: Digital Signature generation

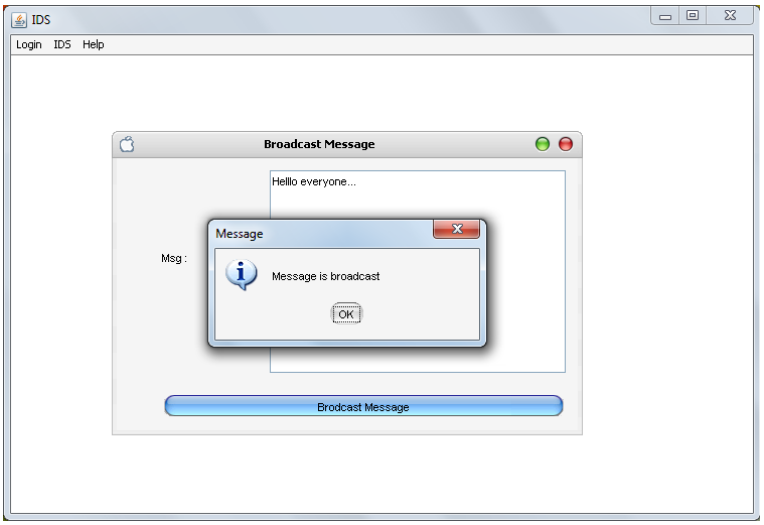

Figure 7.4: Broadcasting Message

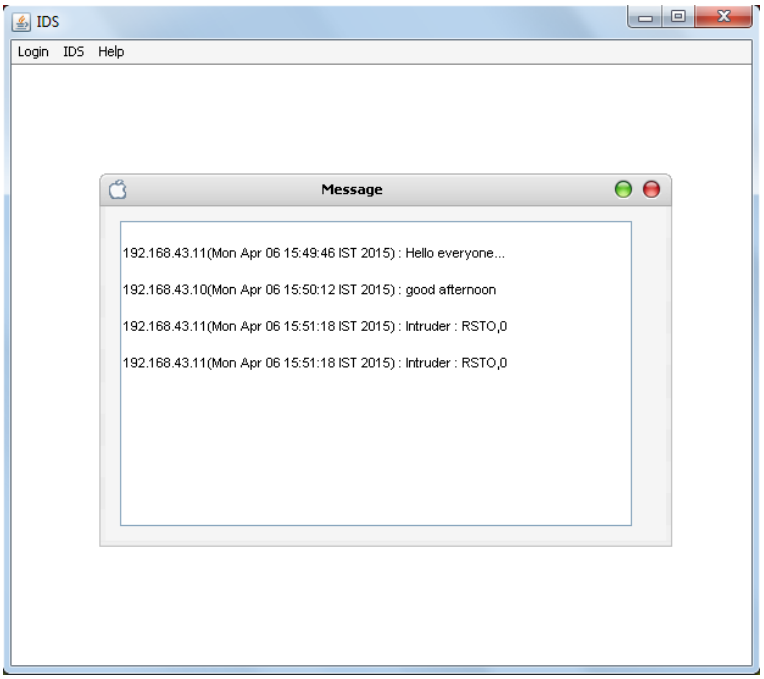

Figure 7.5: Income Message Window

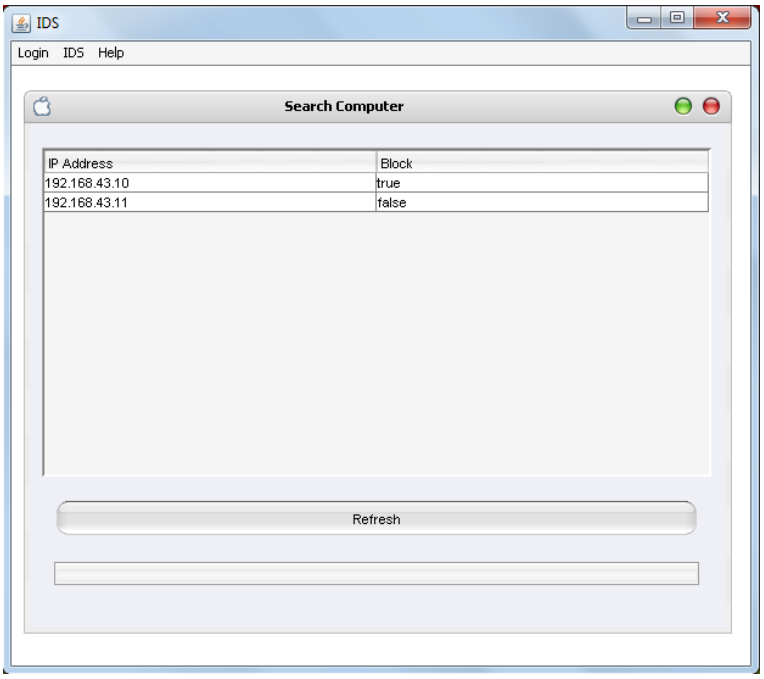

Figure 7.6: Intruder detected

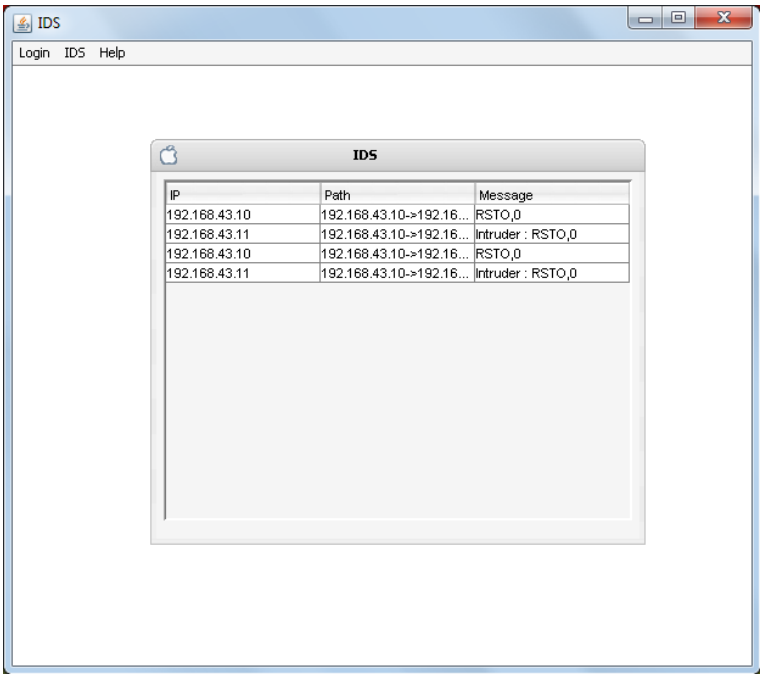

Figure 7.7: Intruder Detection Window

7.2 Results

As mentioned above, due to changes in normal behaviour of the network and appearance of new attacks, using the static model for Intrusion Detection Systems is not relevant. Here we have improved the performance of intrusion detection by updating the detection model substantially. This section describes the experimental result obtained for Proposed Crisp controller based Intrusion Detection System. To compare the results with Crisp based IDS and Fuzzy Based IDS as Shown in following Table[3]

Systems	**TP**	**TN**	**FP**	**FN**	**Accuracy**
Crisp controller	91	34	66	9	86
Fuzzy Logic	86	42	52	14	79

Table 7.1: Comparison between Crisp and Fuzzy based IDS

The accuracy of intruder detection is increased by some percentage using crisp controller. Following graph gives the comparison between crisp and fuzzy controller. Results are compared with different values such as TP, TN, FP, and FN.
True Positive means intruder present in the network and it is detected.
True Negative means intruder is present in the network and it is not detected.
False Positive means intruder is not present in the network and it is detected.
False Negative means intruder is not present in the network and it is not detected. Using

these values accuracy is calculated for both the system.

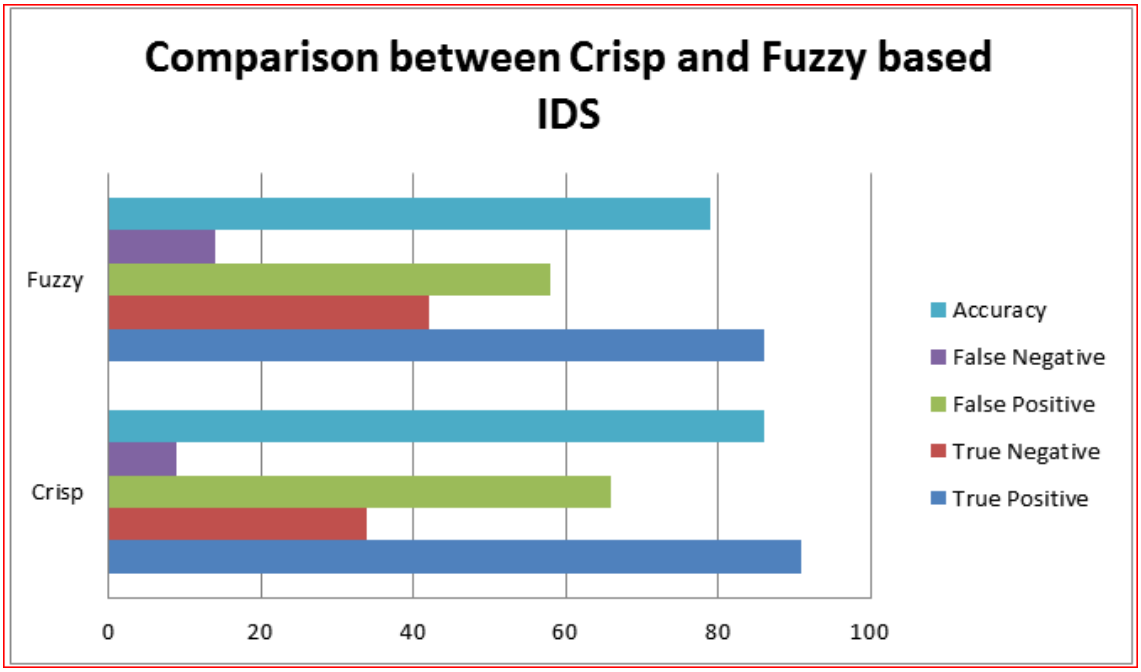

Figure 7.8: Comparison between Crisp and Fuzzy based IDS

Chapter 8

Conclusion and Future Scope

anomaly based Intrusion Detection Systems are provided in order to protect computer networks against novel attacks and improve Network security. These systems perform intrusion detection by comparing current network traffic with a behavioral model of normal network activity. As the pattern of network traffic changes over time, static models are not appropriate to monitor malicious activities. As the static models could be tuned with respect to changes in traffic pattern, adaptive models are used in this manner. In this, we have addressed the dual problem of Accuracy and Efficiency for building robust and efficient Intrusion Detection Systems. Our experimental results will show that Crisp controller are very effective and accurate in improving the attack detection rate.

To improve the accuracy for detect the anomaly in the Intrusion Detection System we extend our work with Genetic algorithm. Further improvements of Intrusion Detection Systems can be done by applying standard clustering algorithms. Combining knowledge from different security sensors into a standard rule base may also be used for improvement in this emerging field of research. FCC and Genetic Algorithms can be used to increase the efficiency of machine learning System.

Bibliography

[1] K.Kavitha, and S.Ranjitha Kumari,"Particle Swarm Optimization For Adaptive Anomaly-Based Intrusion Detection System Using Fuzzy Controller", International Journal of Computer Trends and Technology (IJCTT) volume 4 Issue10 Oct 2013.

[2] Kamal Kishore Prasad, and Samarjeet Borah,"Use of Genetic Algorithms in Intrusion Detection Systems: An Analysis",International Journal of Applied Research and Studies (iJARS) ISSN: 2278-9480 Volume 2, Issue 8, Aug-2013.

[3] Vrushali Patil, Vivek Bonde, Yogesh Kate and Punam Patil,"Enhanced Intrusion Detection System using Crisp Controller", International Journal of Scientic Engineering and Technology Research (IJSETR), ISSN 2319-8885 Vol.04,Issue.08,April-2015.

[4] S. Saravanan and Dr. R M. Chandrasekaran,"Intrusion Detection System using Bayesian Approach", International Journal of Engineering and Innovative Technology (IJEIT) Volume 4, Issue 7, January 2015.

[5] J.S.Narendra Kumar, T.Sudha Rani and M.Raja Babu,"Accuracy and Efficiency in Intrusion Detection System",IJCST, Volume-3, Issue 1, SPL 5, March 2012.

[6] Jonatan Gomez and Dipankar Dasgupta,"Evolving Fuzzy Classi

ers for In-trusion Detection", Proceedings of the 2002 IEEE Workshop on Information Assurance United States Military Academy, West Point, NY June 2001.

[7] Autonomous Agents for Intrusion Detection, http://www.cerias. purdue.edu/research/aa

d/, 2010.

[8] B.Amor, S.Benferhat, and Z. Elouedi, Naive Bayes vs. Decision Trees in In-trusion Detection Systems, Proc. ACM Symp. Applied Computing (SAC 04),pp. 420-424, 2004.

[9] 2. CRF++: Yet Another CRF Toolkit, http://crfpp.sourceforge.net/, 2010.

Index