1. INTRODUCTION

1.1 Overview

The system we had in mind while starting with the project was the Network Intrusion Detection System which will be useful to make Network attacks less and to prevent any harm to our system from the network.

The system would be useful to detect attacks and to notify the Network Administrator of the same. The major types detected would be DoS, Probe or Normal data.

1.2 Brief Description

➢ **Real-Time Network Intrusion Detection System:**

- Provides a wall of defense to confront the attacks of computer systems on Internet.
- Collects information from a network analyzes the information for symptoms of system breaches.
- Monitors network data and gives an alarm signal, when detects antagonistic activity on an open port.

➢ **False Alarm Rate:**

- Anomaly detection system has the potential to generate too many false alarms.
- 4 quadrants based on the result obtained by the NIDS:

  True Positive: attack data detected as attack data.

  True Negative: normal data detected as normal data.

  False Positive: normal data is detected as attack data.

  False Negative: attack data is detected as normal data.

➢ **Machine Learning Algorithms:**

- Used for classifying incoming network data into normal data and attack.
- E.g.
    - Decision Tree
    - Ripper Rule

- Back-Propagation Neural Network
- Radial Basis Function Neural Network
- Bayesian Network
- Näive Bayesian Etc.

1.3 Problem Definition

The title of the project is to "TO IMPLEMENT EFFECTIVE REAL-TIME NETWORK INTRUSION DETECTION SYSTEM TO REDUCE FALSE ALARM RATE AND IMPROVE EFFICIENCY, USING MACHINE LEARNING TECHNIQUE".

Our project is an attempt to provide a common platform for all the Intrusion Detection Systems. We aim to design IDS which can meet the demands of Reducing False Alarm Rate with higher detection rate. Many types of IDS already exist in the world, which provide assistance at different stages of the project development. But a problem commonly observed is the high False Alarm Rate. Our software should be able to assist the developers in this department greatly along with the individual stage support. This software is an attempt to achieve the Real Time Network Intrusion Detection System with a good efficiency.

1.4 Applying Software Engineering Approach

We generally tend to apply the waterfall model for Software Engineering of the project. However, we have used the AGILE model.

We started with the resource collection, the Design of the Project and then moved on to coding. While the coding commenced, we also simultaneously tested the system and improved our Coding and took feedback about it from the focused user group.

2. LITERATURE SURVEY

The section literature survey helps us to understand the fundamental concepts of the project. The need of the project is explained under Motivation and Goals. Various architecture and tools needed are explained later. Then, explained is the detailed study of the Applications and the existing system.

Literature survey is required prior to the design of any project for the following reasons.

1. To identify the gaps in the body of knowledge.

2. To identify relevant work.

3. To locate useful expertise.

4. To keep abreast of the Business Development Process.

A] STUDY OF RESEARCH PAPERS:

The papers we studied were:

1. "**Practical real-time intrusion detection using machine learning approaches" Phurivit Sangkatsanee, Naruemon Wattanapongsakorn, Chalermpol Charnsripinyo**

In this paper, they propose a real-time intrusion detection approach using a supervised machine learning technique. The approach is simple and efficient, and can be used with many machine learning techniques. Different well-known machine learning techniques are applied to evaluate the performance of IDS approach.

The experimental results show that the Decision Tree technique can outperform the other techniques. Therefore, they further developed a real-time intrusion detection

system (RT-IDS) using the Decision Tree technique to classify on-line network data as normal or attack data.

They have also identified 12 essential features of network data which are relevant to detecting network attacks using the information gain as feature selection criterions. The RT-IDS can distinguish normal network activities from main attack types (Probe and Denial of Service (DoS) with a detection rate higher than 98% within 2s. They have also developed a new post-processing procedure to reduce the false-alarm rate as well as increase the reliability and detection accuracy of the intrusion detection system.

Thus, the performance evaluation results showed that the proposed real-time IDS is efficient in terms of real-time detection speed and consumption of CPU and memory. It takes only 2 s to detect and classify the incoming network data. The classification results can be improved with an optional data post-processing procedure, which can additionally reduce the false alarm rate.

2. "**Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers"**
   Ming-Yang Su

This study proposed a method which can detect large-scale attacks, such as DoS attacks, in real-time by weighted KNN classifiers. The key factor for designing an anomaly-based NIDS is to select significant features for making decisions. Not only is excellent detection performance required, but real-time processing is also demanded for most NIDSs.

A good feature selection policy, which can choose significant and as few as possible features, plays a key role for any successful NIDS. The study proposed a genetic algorithm combined with KNN (k-nearest-neighbor) for feature selection and weighting. All initial 35 features in the training phase were weighted, and the top ones were selected to implement NIDSs for testing.

Many DoS attacks were applied to evaluate the systems. For known attacks, an overall accuracy rate as high as 97.42% was obtained, while only the top 19 features were considered. For unknown attacks, an overall accuracy rate of 78% was obtained using the top 28 features.

3. **"Improving the Accuracy of Intrusion Detection Systems by using the combination of Machine Learning Approaches"**
Hadi Sarvari, and Mohammad Mehdi Keikha.

This paper focuses on combination of machine learning approaches as to detect the system attacks. This article proposes a combinatory system. Primarily, it was concluded that the system accuracy increases by increasing the number of classes. Then it was revealed that this increase continues to a point at which it stops and remains constant.

They are given description about KDDCup99 Data set. One of the features of KDD 99 is that their samples are much fewer than those of other classes and since machine learning-based systems do not learn the features of these two classes, their detection accuracy are also much less than other two classes.

They have presented the Intrusion Detection Systems by Combination of Machine Learning system in that they have given results first of separately for NN, Decision Tree and SVM algorithm. After that, they gave results on the different combinations of algorithms. The U2R and R2L class samples are much fewer than other classes and therefore their detection percentage is less than of the others, too. This is called unbalanced data problem with input data of different classes.

They generated data for these two classes to solve the problem of the unbalanced data, so that they helped the system to recognize the features of these two classes. The addition of such a data may produce data not being in the input data. In fact, unknown features have been introduced to the system. As a whole, this system improved the recognition of R2L and U2R classes. KDD 99 contains 24 different known attacks in training data and 14 unknown attacks in testing data. After analyzing different

combinatory models, they reached their conclusion that the best model is one containing SVM, DT, 1NN, 2NN and 3NN.

4. "**Network intrusion detection and classification with decision tree and rule based approaches"**

T. Komviriyavut, P. Sangkatsanee, N. Wattanapongsakorn, C. Charnsripinyo,

In this paper, they present two network intrusion detection (IDS) techniques which are C4.S Decision Tree and Ripper rules to assess and test an online dataset (RLD09 dataset). The dataset was collected from actual environment and then preprocessed to have only 13 features which are much simpler than existing traditional dataset such as KDD99 with 41 features. Thus, the RLD09 dataset features can provide real-time detection speed with low memory and CPU consumption. The IDSs can classify the network data into classes which are normal data, Denial of Service (DoS) attack, and Probe (Port Scanning) attack. The IDS techniques give the detection rates higher than 98%. Furthermore, they can detect unknown or new attacks, where the C4.S Decision Tree detection rate is about the double of the Ripper rule detection rate. These tests can prove that our techniques are effective in detecting and classifying the new unknown attacks in the real environment.

5. "**A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach"**

M-Y. Su, G-J. Yu, C-Y. Lin,

This study proposed a real-time NIDS with incremental mining for fuzzy association rules. By consistently comparing the two rule sets, one mined from online packets and the other mined from training attack-free packets, the proposed system can render a decision every 2 seconds. Thus, compared with traditional static mining approaches, the proposed system can greatly improve efficiency from offline detection to real-time online detection.

Since the proposed system derives features from packet headers only, like the previous works based on fuzzy association rules, large-scale attack types are focused. Many DoS attacks were experimented in this study. Experiments were performed to demonstrate the excellent effectiveness and efficiency of the proposed system. The system may not cause false alarms because normal programs supposedly would not generate enough mal-formatted packets, or packets that violate normal network protocols.

.

3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Introduction

Internet is a global public network. With the growth of the Internet and its potential, there has been subsequent change in business model of organizations across the world. More and more people are getting connected to the Internet every day to take advantage of the new business model popularly known as e-Business. Internetwork connectivity has therefore become very critical aspect of today's e-business.

While an organization makes its information system available to harmless internet users, at the same time the information is available to the malicious users as well. Malicious users or hackers can get access to an organization’s internal systems in various reasons. The malicious users use different techniques to exploit the system vulnerabilities and compromise critical systems.

Therefore, there needs to be some kind of security to the organization’s private resources from the Internet. Different organizations across the world deploy firewalls to protect their private network from the Public network. But, when it comes to securing a Private network from the Internet using firewalls, no network can be hundred percent secured. This is because; the business requires some kind of access to be granted on the internal systems to Internet users. And therefore for assuring the complete security to the organization, an Intrusion Detection System is implemented.

An Intrusion detection system (IDS) is a security system that monitors computer systems and network traffic and analyzes that traffic for possible hostile attacks originating from outside the organization and also for system misuse or attacks originating from inside the organization.

3.1.1 Purpose

To implement an effective RT-NIDS for assuring the complete security against the various types of known, unknown attacks and adding up to the confidentiality, integrity, authentication, availability of the data and the systems

3.1.2 Intended audience and reading suggestions

The system is basically developed because of the threats that occur due to the exchange of data over the internet. So, the intended audience for the system would be any person or any organization that has to deal with the internet and is prone to the attacks due to the interaction over the internet, which compromises the data confidentiality, integrity and availability.

The system is so designed that any computer user would be able to understand and use it. No other additional information or no other knowledge regarding any particular technology is needed to use the system. The complexity or the internal details of the system are been hidden from the user keeping the golden rule of Software Engineering in mind.

3.1.3 Project Scope

Basic functionality of IDS can be differentiated with regard to the detection technique, misuse and anomaly detection (behavior)

1. **Misuse detection-**

   It is also known as signature detection technique which searches for well-known patterns Misuse detection aims to detect known attacks by characterizing the rules that govern these attacks. Thus, rules update is particularly important, and consequently, new definitions are frequently released by NIDS vendors. However the rapid emergence of new vulnerabilities makes misuse detection difficult to trust.

2. **Anomaly Detection-**

   It is also known as behavior detection technique which builds a model of the network behavior and attacks can be detected by measuring significant deviation of the current status against the behavior expected from the model.

   Anomaly detection is designed to capture any deviation from the profiles of normal behavior patterns. Anomaly detection is much more suitable than misuse detection to detect unknown or novel attacks, but it has the potential to generate too many false alarms. For an anomaly-based NIDS, the most difficult part is to present the normal profile, which depends on the policy of feature weighting and selection.

An IDS is a network based anomaly detector aimed to provide accurate and real-time enterprise intrusion detection and prevention solution to combat known attacks and reduce False Alarm Rate. An IDS is developed to provide a complete, better than existing and an open source solution to the rising no of insecure enterprise networks. Scope of our project is to analyze our IDS and provide our case study as a reference for further research and development of IDS.

This project aims at:

1. Achieving maximum intrusion detection rate
2. Achieve negligible false alarm rates
3. Increase efficiency of the system
4. Provide availability of resources for further research and development

The project outlines the following objectives:

1. Achieving expected detection and false alarm rates
2. Providing a user friendly menu for configuring and scaling the available options
3. Smooth running system with complete error and exception handling

3.1.4 Design and Implementation Constraints

Processing Power:-

IDS requires high speed processing machine, which is required to fulfill all the tasks such as capturing the real time traffic, processing and analyzing it, and providing the quick and accurate results.

Deployment Point:-

We will get the incoming data from the real-time sources using a packet sniffer like Wireshark or Capsa.

Detection/False Alarm Rate:-

Detection and false alarm rates depend on the choice of algorithm from user. With detections, come a number of false alarms as well. There will be further release of IDS in future that will improve these parameters in future. We also try to achieve better results using a combination of the different machine learning algorithms. By comparing the results obtained from various algorithms, we can get more accuracy with a very small number of false alarms.

Operating Platform:-

The target operating system of IDS is Linux and Windows. The solution should be developed such that it can smoothly run on several different distributions of Windows and Linux Operating Systems.

3.1.5 Assumptions and Dependencies:

We assume that the data be feed in the training and testing phase of the system is provided from a real-time packet sniffer. This data is then converted to specific format so that it can be processed and analyzed correctly and necessary and appropriate features from the data

packets can be extracted and further analysis on them can be done to classify them into the attack or normal category.

Dependency:

Packet Sniffer for Windows: Wireshark

3.2 System Features:

The proposed solution shall provide several services to its users. Major services provided by the IDS system are briefly discussed below.

3.2.1 Detecting Attacks

Packet Capturing Tools generate packets that have a packet format similar to the one required to process. The KDD99 data set is considered as the criteria for the packet format. The packets are converted into this specific format and then preprocessed. KDD has normal data and added to it are the four types of attacks:

1. DoS attack –

Attacker tries to prevent legitimate user from getting access to service eg. Neptune, Teardrop.

2. Probe –

Attacker tries to gain information about the target host. Eg Satan, Portsweep,etc

3. U2R-

Attacker has access to local machine and has super user privileges. Eg- buffer overflow, perl, etc

4. R2L-

Attacker does not have an account on the victim machine and tries to gain access eg guess-pswd, spy, etc.

The purpose of intrusion detection is to detect malicious activities from given datasets. So, how to express an illegal connection is essential. The selection of input features depends on attack type. There are majority of DoS attacks that frequently scan the hosts (or ports) using a larger time interval than 5 seconds, for example, one every minute.

So, host based and time-based features are used as Probe patterns. However, R2L and U2R, unlike most DoS and Probe attacks, don't have any "intrusion only" frequent sequential patterns. This is because DoS and Probe attacks involve many connections to some hosts in a very short period of time, the R2L and Probe attacks are embedded in the data portions of the packet and normally involve only a single connection. Therefore, content features were used as these attack features patterns. No matter what attack types, basic features are included in intrusion feature patterns.

3.2.2 Displaying Result

We will display our result in following formats:

1. Tabular format
2.Graphical Format
3. Percentile

In tabular format, the results will be displayed with different data such as total normal sample, attack sample, False alarm rate and types of attack detected. In graphical models results will be displayed in a graph whereas in percentile, results will simply be calculated in percentile and displayed.

3.3 External Interface Requirements:

3.3.1 User interfaces:

User Interface will be a GUI application developed in Java language satisfying all required elements to interact with system.

Graphical User Interface will provide following options:

- Display results in Table view
- Display results in Graphical view
- Different Graphical view options

The .csv or .arff files will be input to the program.

3.3.2 Hardware interfaces:

The solution makes extensive use of hardware devices.

- More than 1GB RAM for 10% dataset
- Core2Duo or faster processor
- Windows XP and above versions, Linux as OS

For faster and efficient work, the above requirements should be fulfilled

Interfaces for:

- Host:
  - Internal computer sources
  - Ex: OS audit and system logs
- Network
  - Packets via “sniffing”
- Target-based
  - Monitor object for changes e.g. Tripwire

3.3.3 Software interfaces:

- Python or Java as front end and backend
- Weka tool for testing our results

  – Sniffers

    - collect data from various sources such as log files, network packets
    - sends them to the analyzer

  – Analyzers

    - process data from sensors and determine if intrusion has occurred
    - may also provide guidance for the actions to take.

3.3.4 Communication interfaces:

The packets captured need to be converted into suitable format (according to KDD99 dataset) and communicated in a reliable and efficient way, so that efficient results canbe generated.

– Graphical User interface

- Select operations
- view the output and manage the behavior

3.4 Nonfunctional Requirements

3.4.1 Performance Requirements

The solution has to exhibit following performance requirements:

1. The system must have very high detection rates.
2. The system must have very low false alarm rates.

These requirements shall be achieved by using a combination of several algorithms. Another performance requirement is the detection of anomalies in real time. The active anomaly detection module is proposed for the same purpose.

3.4.2 Safety Requirements

There are no specific safety requirements associated with the purpose system. The IDS is composed of well-known and commonly used machine learning technique. This does not cause any safety hazards.

3.4.3 Security Requirements

Only authorized personnel are allowed to use the product and go through selection procedure. In case of forgotten passwords contact the developers. Similarly, changing the feature of the solution at runtime also requires password based authentication.

3.4.3 Software Quality Attributes

- **Reliability**

IDS should provide reliability to the user that the product will run stably with all. The features mentioned above available and executing perfectly. It should be tested and debugged completely. All expectation should be well handled.

- **Accuracy**

IDS should be able to reach the desired detection level. It should generate minimum false positive alerts with maximum detection rate.

- **Resources**

IDS should use minimal resources in term of memory, time and CPU.

- **User Friendliness**

IDS should have a graphical user interface with user friendly menu.

3.5 Other Requirements

3.5.1 Database Requirements

We have used our own database that was generated through attacks and captured through a packet sniffer.

3.5.2 Internationalization Requirements

The project requires internationalization among members of the group. There should be friendly environment among team members and efforts must be taken by all our team for best project prospects.

3.5.3 Legal Requirements

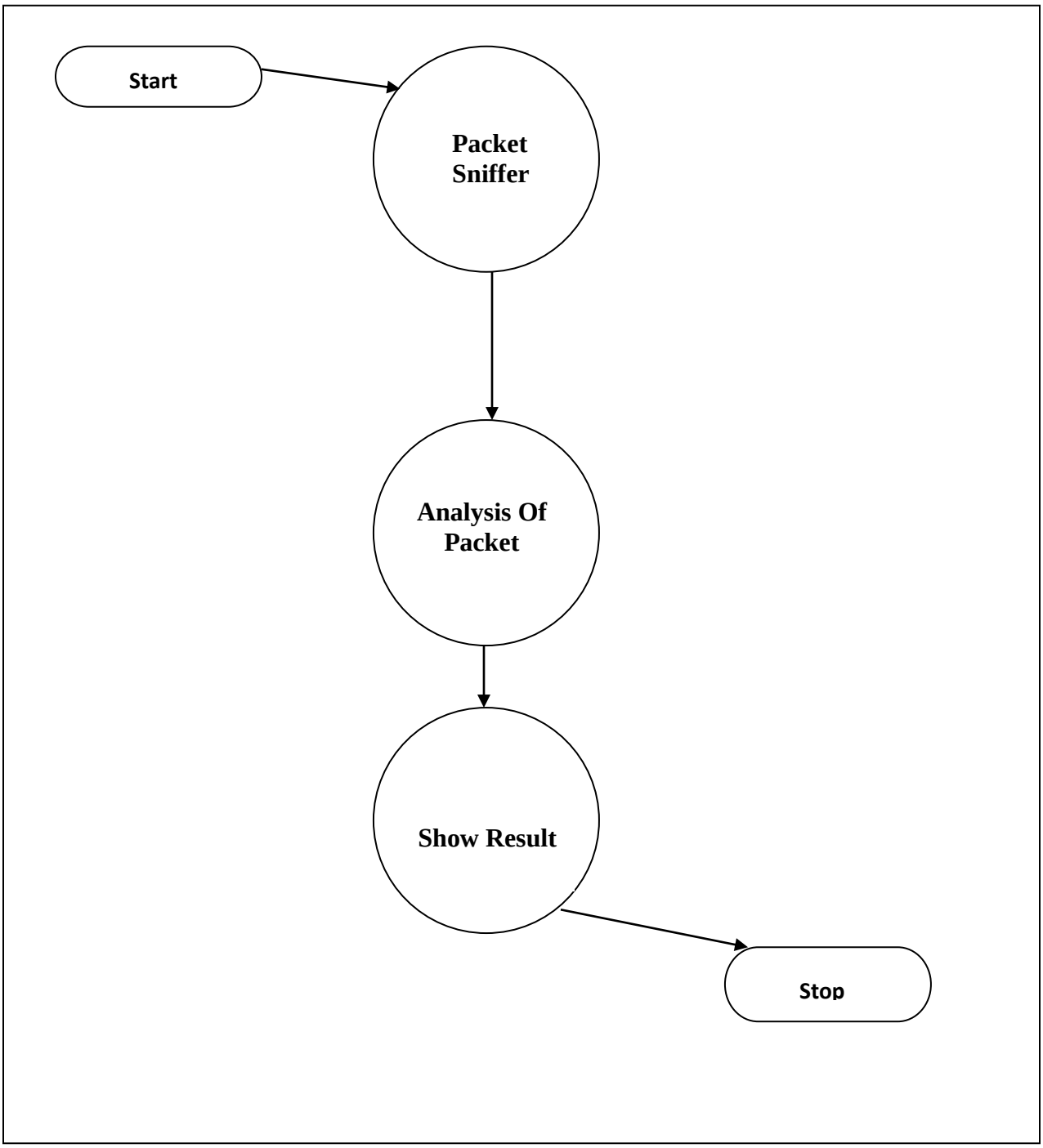
None.

3.5.4 Reuse Objectives for the project

The analyzing result can be used for further research and development

4. SYSTEM DESIGN

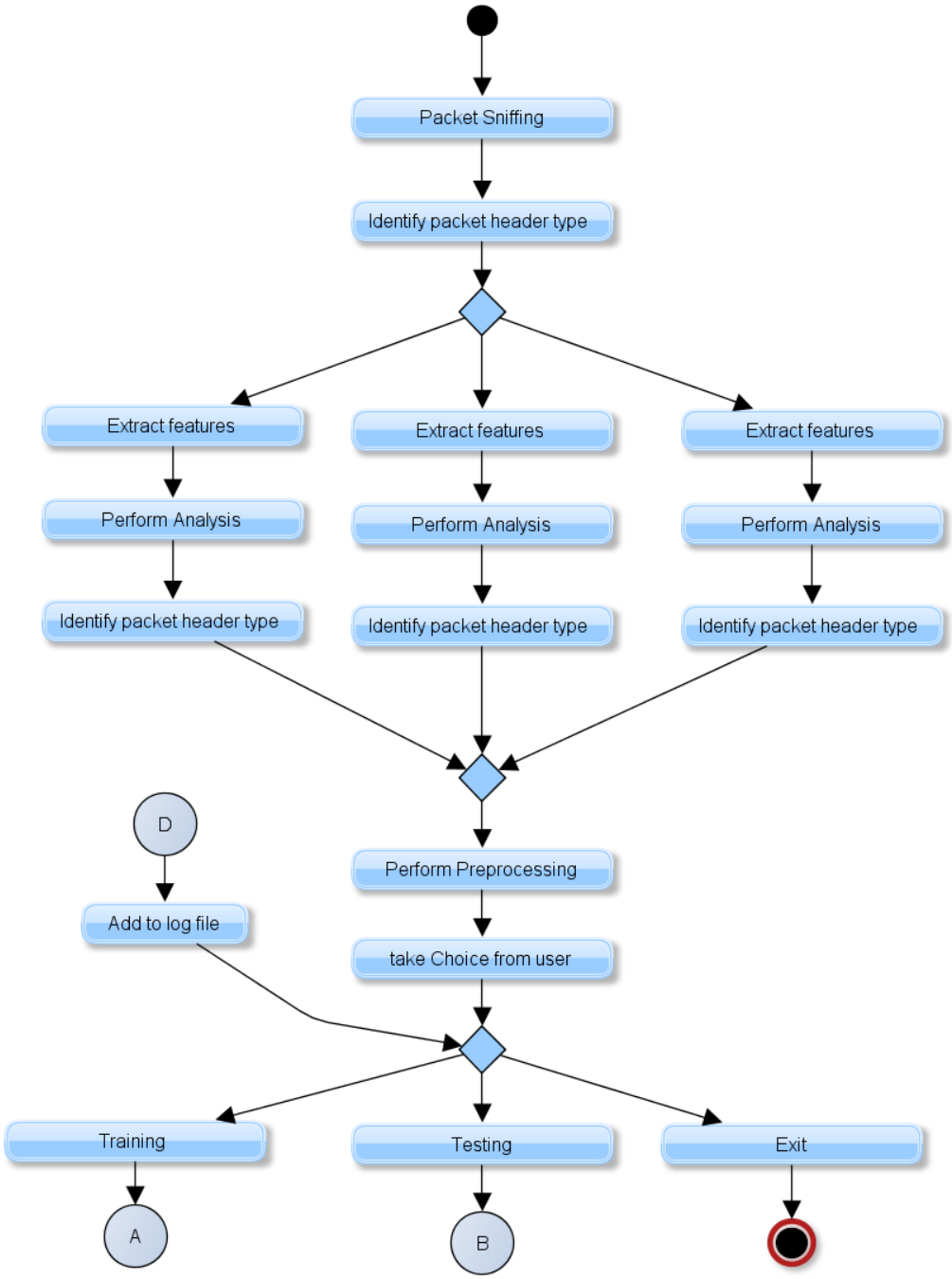
4.1 System Architecture



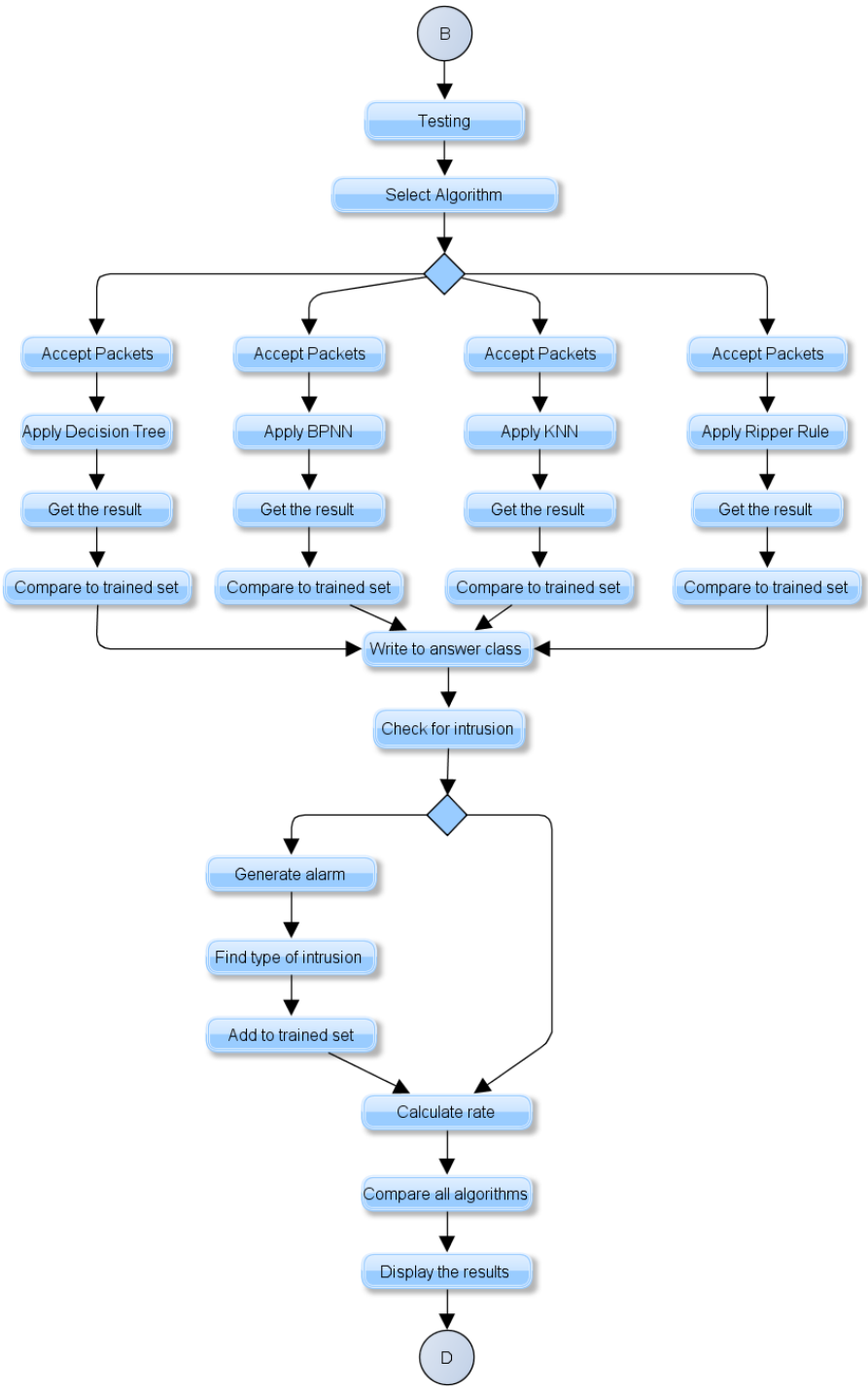
4.2 UML Diagrams

4.2.1 Activity Diagram

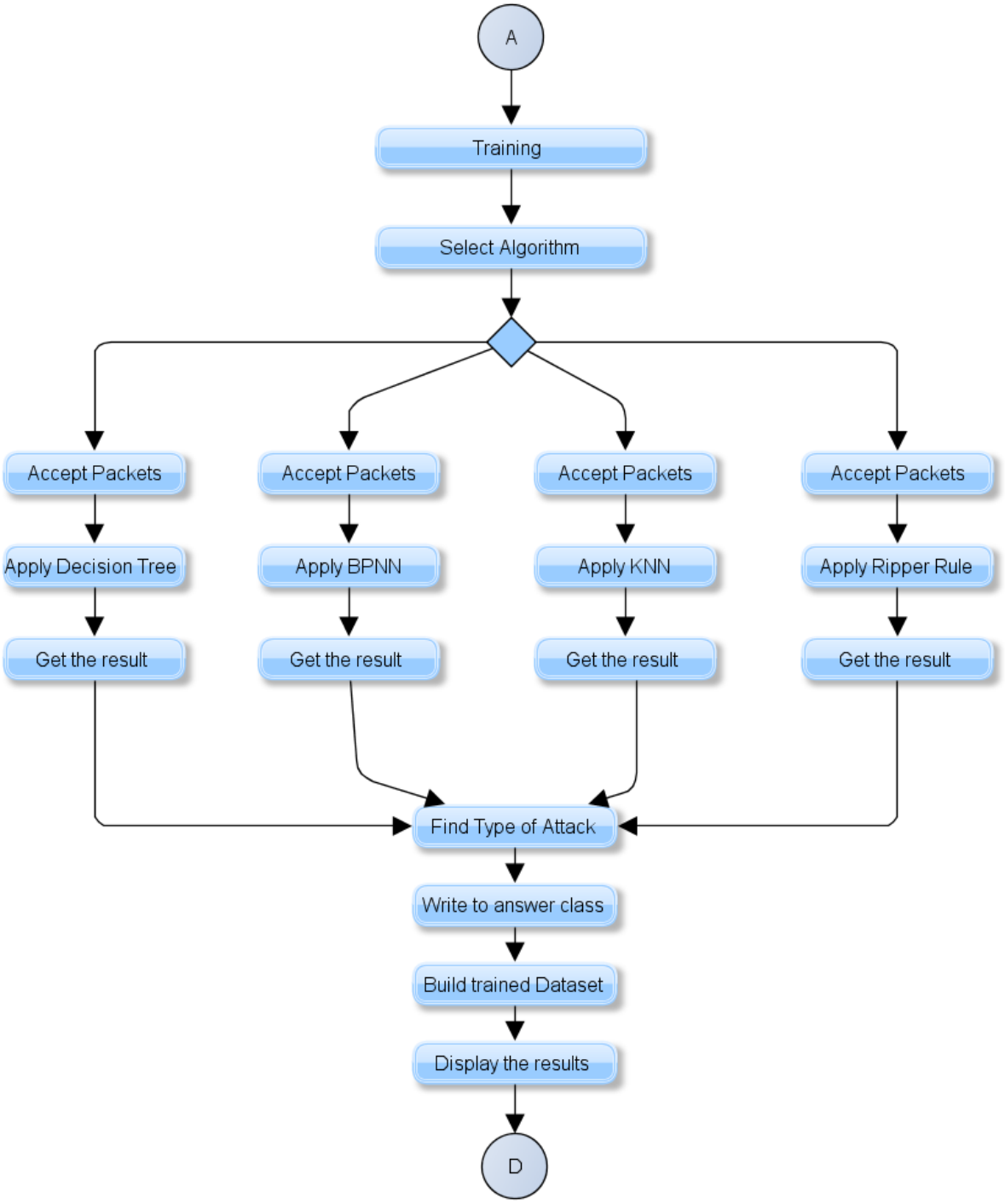
a. Pre-Processing



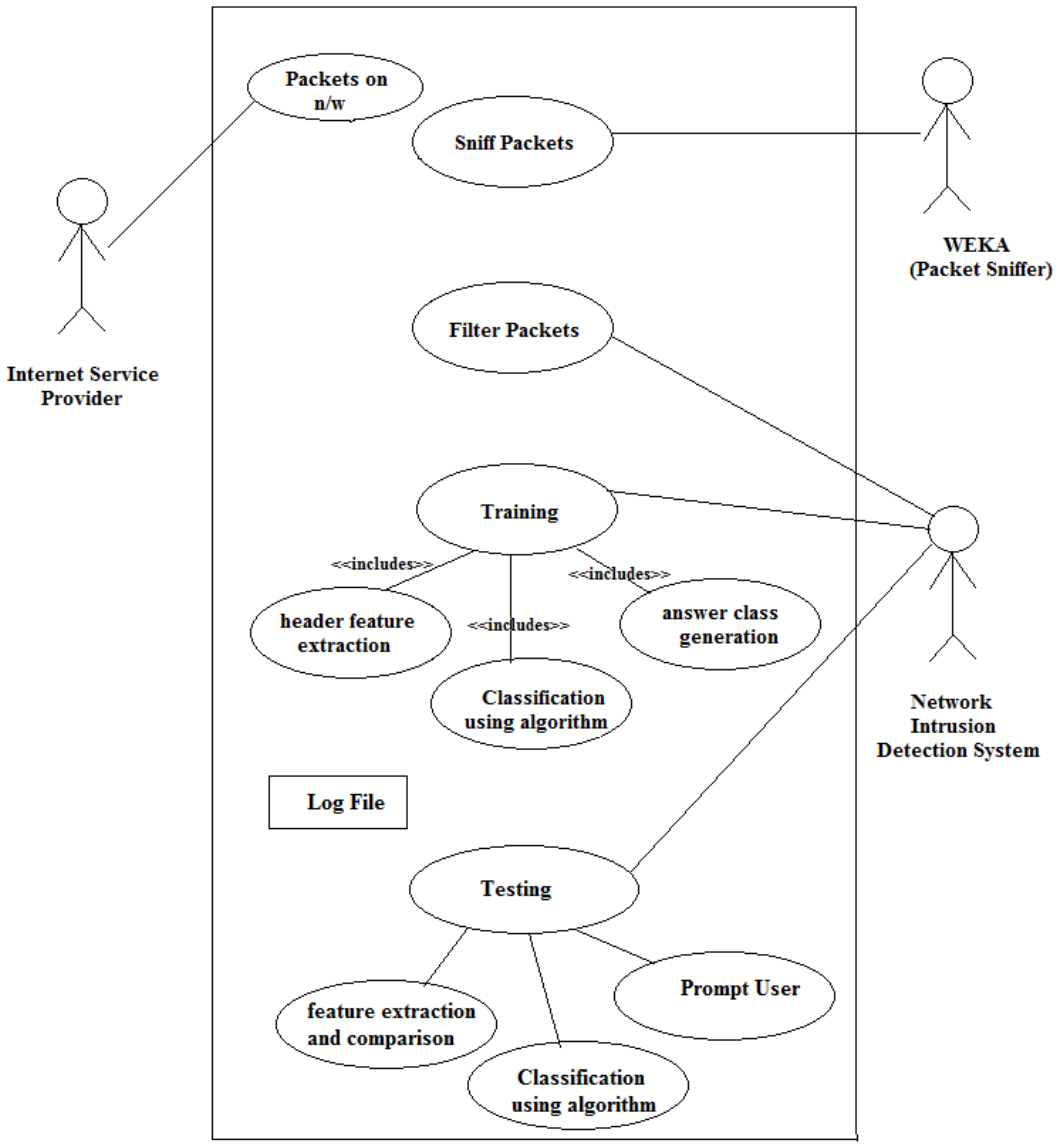
b. Training Phase:



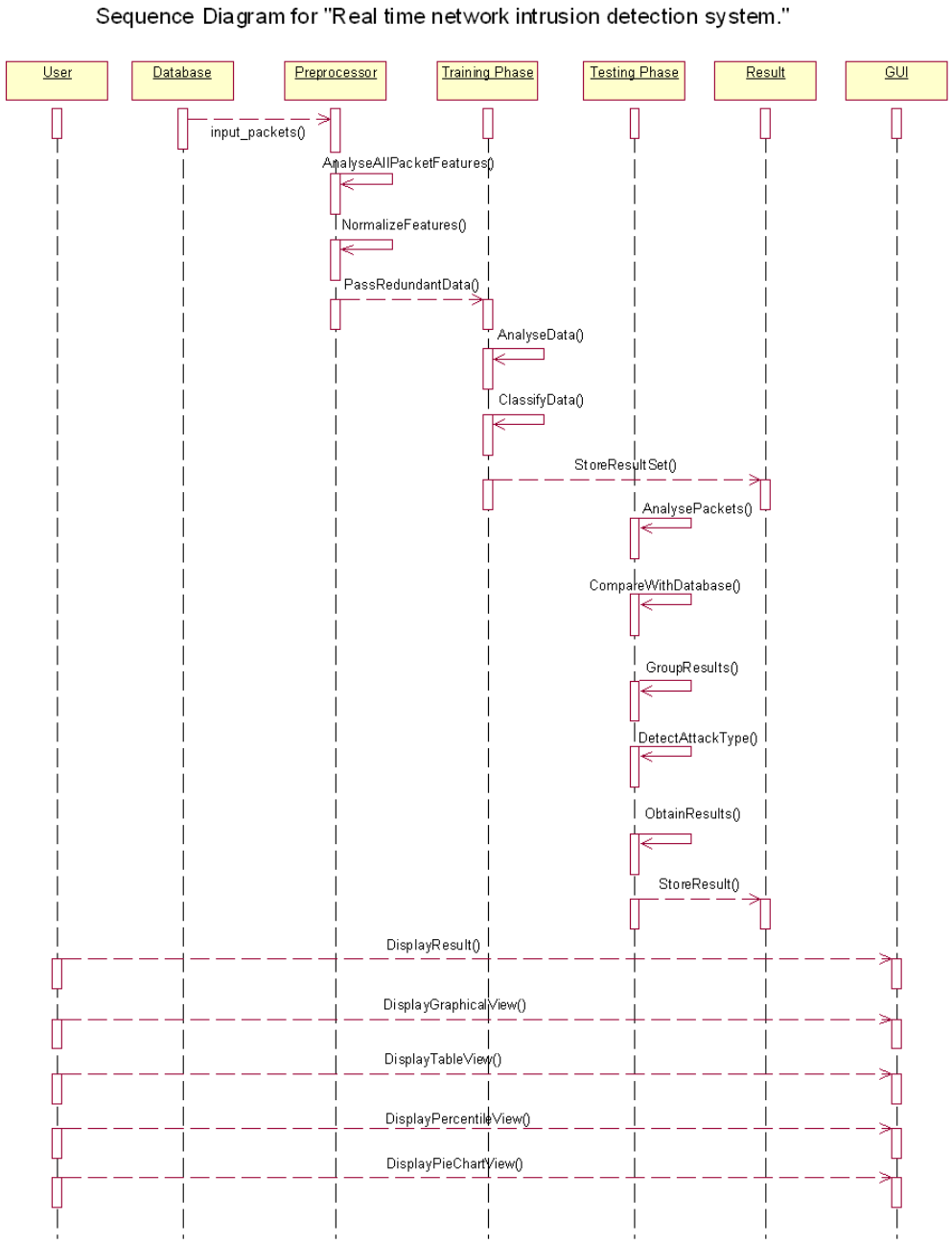
c. Testing Phase:



4.2.2 Use Case Diagram:



4.3 Sequence Diagram



5. TECHNICAL SPECIFICATION

5.1 Domain Area of Project

The domain of the project is Machine Learning and network security.

5.2 Technology Consideration

1. Windows XP/7/Vista
2. Java
3. Java Packages
    - 3.1 java.util.*;
    - 3.2 java.io.*;
    - 3.3 java.net.*;
    - 3.4 java.swing.*

5.2.1 Java

- **Simple**

Java was designed to be easy for the professional programmer to learn and use effectively. If you already understand the basic concepts of object-oriented programming, learning Java will be even easier. Best of all, if you are an experienced C++ programmer, moving to Java will require very little effort. Because Java inherits the C/C++ syntax and many of the object-oriented features of C++, most programmers have little trouble learning Java. Also, some of the more confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. Beyond its similarities with C/C++, Java has another attribute that makes it easy to learn: it makes an effort not to have *surprising* features. In Java, there are a small number of clearly defined ways to accomplish a given task.

- **Object-Oriented**

One feature of java is a clean, usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high performance non-objects.

- **Robust**

The multi-platformed environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. Thus, the ability to create robust programs was given a high priority in the design of Java.

- **Multithreaded**

Java supports multithreaded programming, which allows you to write programs that do many things simultaneously. The Java run-time system comes with an elegant yet sophisticated solution for multiprocess synchronization that enables you to construct smoothly running interactive systems.

- **Java's Magic: The Bytecode**

The key that allows Java to solve both the security and the portability problems just described is that the output of a Java compiler is not executable code. Rather, it is bytecode. *Bytecode* is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the *Java Virtual Machine* (*JVM*). That is, in its standard form, the JVM is an *interpreter for bytecode.*

6. PROJECT ESTIMATE,SCHEDULE AND TEAM STRUCTURE

6.1 Feasibility Study

The feasibility study of our project "To Reduce False Alarm Rate [FAR] for Intrusion Detection System [IDS] by machine learning algorithm" involves the following constraints:

6.1.1 Technical Feasibility

The technology and the resources used to develop the system involve the following components.

1. Hardware Components

- Computer system

2. Software Components

- Java, Windows XP/7/Vista

The technical specifications of the system imply that the system is very well adaptable to the available technology and does not demand any more technically advanced, unavailable or infeasible equipment.

6.1.2 Economical Feasibility

The resources required by the system are available easily in the market at reasonable cost. The various costs incurred in the system are as given below:

Component	Cost
Personal computer(Intel Core2Duo) CPU 2 5 GHz , 1 GB RAM)	Rs. 25,000
Java 1.6.0	Rs. 0

Viewing the specifications, the system was found to be economically feasible for the concerned user. The above specified are onetime costs.

6.1.3 Time Feasibility

There were several studies carried out during the project development. This task was done successfully. Updates were kept, the weekly report of advancement during project time. Various issues and ideas are discussed in the meeting with the College Staff and the Project Guide and we strive to get better solutions.

6.1.4 Operational Feasibility

Modern PCs are becoming easier to handle. Also the system is providing both types of input i.e. keyboard and mouse. The users who are comfortable with using PCs find this application operationally much more feasible.

6.2 Resources

6.2.1 Human Resources

The total human resources required for the development of the project is four. The testing resources required to test the project is four.

6.2.2 System Requirements

1. Hardware Requirements

"To Reduce False Alarm Rate [FAR] for Intrusion Detection System [IDS] by machine learning algorithm" for Linux has following minimum hardware requirements.

- GHz CPU 2 GB RAM
- 80 GB Hard Drive
- Keyboard, mouse(optical mouse controls work the best)
- Color Monitor screen resolution 1024x768

- **2. Software Requirements**

  "To Reduce False Alarm Rate [FAR] for Intrusion Detection System [IDS] by machine learning algorithm" has the following software requirements.

- Windows XP/Vista/7

- JDK 1.6.0

6.3 Scheduling

Project Start Date	26-July-2011
Project End Date	21-April-2012
Project Duration	9 months

Table 6.2 Project Duration

Sr. No.	Milestone Name	Milestone Description Information relating to the deliverable at this milestone	Timeline No. of weeks for the current milestone	Remarks Comments such as the weight age of the milestone in percentage of the total project etc.
1	Design of Project Overview Construction	Shows the participants in the project and an overall design.	11	15%
2	Developing and testing of modules	Preparing and designing the User Interface, Command, Syntax and Semantic Recognition, Command Implementation, and additional utilities	16	50%

3	Final Integration and Project testing	Testing of project and specification	2	50%
4	Documentation and Project report and Presentation		2	50%

Table 6.3 Phases of Project

6.4 Risk Management

Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project. A risk is a potential problem-it might happen it might not. But, regardless of the outcome, It's a really good idea to identify it, access its probability of occurrences, estimate its impact, and establish a contingency plan should the problem actually occur.

For any project developed, it is empirical for the team to monitor and manage risks. Risks are the unexpected delays and hindrances that are faced by the software and need to be actively managed for rapid development. In the words of Tom Glib, "if you don't actively attack risks, they will actively attack you."

The key functions of software risk management are to identify, address and eliminate sources of risk before they become threats to successful completion of a software project. An effective strategy must address three issues:

- Risk avoidance
- Risk monitoring
- Risk management and contingency planning

If a software team adopts a proactive approach to risk, avoidance is the best strategy. This is achieved by developing a plan for risk mitigation. As the project proceeds, risk monitoring activities commence and the project manager monitors factors that may provide indication of whether the risk is becoming more or less likely. Risk management and contingency planning assumes that mitigation efforts have failed and the risk has become a reality.

The RMMM plan tackles risks through risk assessment and risk control. Risk assessment involves risk identification, risk analysis and risk prioritization; while risk control involves risk management planning, risk resolution and risk monitoring.

6.4.1 Risk Table

The risks are categorized on the basis of their occurrence and the impact that would have, if they do occur. Their impact is rated as follows:-

1. Catastrophic.

2. Critical.

3. Marginal.

4. Negligible.

Risk	**Category**	**Probability**	**Impact**
1.Size estimate may be low	Project	50%	Marginal
2. Stricter completion Schedule	Business	10%	Critical
3. Sophistication of the end users application program	Business	40%	Marginal
4. Less reuse than planned	Technical	25%	Marginal
5.Poor Quality documentation	Business	30%	Critical
6. Debugging phase may take more time than expected	Technical	30%	Marginal
7. Poor comments in the code	Technical	20%	Negligible
8. Lack of technical support	Technical	35%	Critical

on unforeseen environment			
9. Increase on the workload on the developers and hence divided attention on project	Personal	40%	Critical
10. Schedule might slip due to inexperienced persons	Personal	40%	Critical
11. Hardware Risks	Support	40%	Critical
12. Environmental Risks	Performance	20%	Marginal
13. Security Risks	Security	20%	Catastrophic

Table 6.4 Risk Table

6.4.2 RMMM (Risk Mitigation, Monitoring and Management) Plan

- **Risk 1:**
  - Size estimate may be high
  - **Mitigation:** keep on modularizing the software and estimate size of individual modules.
  - **Monitor:** module to be written.
  - **Management:** increase members in the more time consuming modules.

- **Risk 2:**
  - Strict completion schedule.
  - **Mitigation:** schedule the project in such a fashion that major modules are completed first so that they get enough time for testing and debugging.
  - **Monitor:** keep track of the schedule slips.
  - **Management:** put in extra hours to make up for the lost hours.

- **Risk 3:**

- Less reuse than planned
- **Mitigation:** use OOPS concepts such as classes, objects, functions etc.
- **Monitor:** modules should not be written containing similar functionality
- **Management:** revise all the modules and take necessary steps to make it reusable.

- **Risk 4:**
  - Sophistication of the end users application program
  - **Mitigation:** complex queries and function are to be implemented.
  - **Monitor:** every minor issue relating the query is concerned
  - **Management:** customer should be clearly notified about the limitations of the functionalities.

- **Risk 5:**
  - Debugging phase may take more time than expected
  - **Mitigation:** use debugging tools if available.
  - **Monitor:** Ensure that debugging tools are properly used or not.
  - **Management:** increase the no of hours to be worked for each member

- **Risk 6:**
  - Poor comments in the code:
  - **Mitigation:** commenting standards are better studied and expressed
  - **Monitor:** review of the code with special attention given to comments will determine if they are up to standard.
  - **Management:** time must be made available to make comments up to the standards.

6.4.3 Team Structure

Sr. No.	Work	Team members
1.	GUI	Mrinal, Sampada, Anjum, Jyoti
2.	Algorithms	Sampada, Mrinal, Jyoti, Anjum
3.	Documentation	Mrinal, Anjum, Jyoti, Sampada

Table 6.5 Team Structure

7. SOFTWARE IMPLEMENTATION

7.1 Introduction

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. We implemented our project through machine learning techniques. Machine learning algorithms used are K-NN and Decision Tree. During implementation proposed CPU time, speed of execution of modules, handling database and data structure is a tough task. Selecting proper data structure is very important for executing programs. Choosing the most appropriate language for implementation, whether it is platform dependent or not and handling all various types of files for execution should be kept in mind. Now are project deals is done in Windows which is OS independent and pure object oriented.

7.2 Database

Attack was sent from an intruder machine to our machines and the data that was sent was captured through Wire shark Packet Sniffer. The data was stored according to the type of attack that was generated and according to the type of packets used to train the algorithm.

The same database was used for testing purposes. Once the training was done, the testing is done through newer data that is captured through the packet sniffer and used as a raw database.

A connection is a sequence of TCP packages which starts and ends at a specific time with the flow of data form source IP address to the destination IP. 41 features were defined for each connection in this collection, which are divided into four major categories namely: primary features of TCP protocol, time–based and host-based traffic features. Every connection has a label which determines whether it is normal or it is one of the defined attacks. The present attacks are divided into two categories as follows: DOS and PROBE. Note that NORMAL is a normal connection.

7.3 Important Modules and Algorithms

7.3.1 k-NN Algorithm

7.3.1.1 Introduction

The k-nearest neighbor's algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. k-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). The main idea is to choose heuristically optimal k nearest neighbors by cross validation technique. If k = 1, then the object is simply assigned to the class of its nearest neighbor.

The basic example of k-NN is as shown in Figure1 which depicts a 3-Nearest Neighbor Classifier on two-class problem in a two-dimensional feature space. In this example the decision for triangle1 is straightforward – all three of its nearest neighbors are of class O so it is classified as an O. The situation for triangle0 is a bit more complicated at it has two neighbors of class star and one of class O. This can be resolved by simple majority voting or by distance weighted voting.

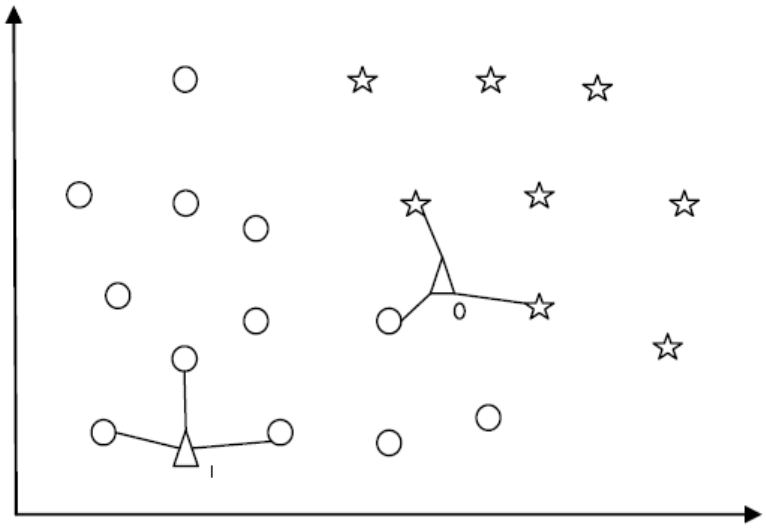


Fig 7.1 Representation of 3NN classifier on a two-class problem

7.3.1.2 Generalization:

There are total 22 types of sub-attacks given in KDD'99 Data Set, classified into 4 attack types. So, we need generalize our training data-entries. After generalizing, all data-entries will have their attack types attach to them, according to table shown below and all entries are stored back into training database.

Sr. No.	Probe	DoS
#1	ipsweep	Back
#2	Nmap	Land
#3	Portsweep	Neptune
#4	satan	Pod
#5	-	Smurf
#6	-	Teardrop
#7	-	-
#8	-	-

Table 7.1 Attack Types Classification

7.3.1.3 Filter:

Filter is used to delete duplicate values from training data. It is coded into module filterdb and these filtered entries are stored back into training data.

7.3.2 Training:

In training phase, user has to set following parameters:

- Split/don't split knndb format (Feature Space)
- k-NN Classifier

- Weighting Function

- Error Acceptance Value (EAV)

7.3.2.1 Don't split /Split knndb format (Feature Space)

This parameter selects the storage structure of feature space. If user selects 'Don't split knndb format' then whole feature space is stored into one single file, and 'k' nearest neighbors are being searched into this file.

If user selects 'Split knndb format' then whole feature space is divided according to its results, that is Normal, Dos, Probe, U2r, R2l, and they are stored into different files. And 'k' nearest neighbors are being searched into all files, so we get maximum 'km' neighbors where 'm' is the number of categories.

Mathematically,

$$\text{knndb\_format} = \begin{cases} 1, & \text{if user selects 'Don'tsplit knndb'} \\ 2, & \text{if user selects 'Split knndb'} \end{cases}$$

7.3.2.2 k-NN Classifier:

This parameter gets value of 'k' from user. User can select any positive integer value ranging from 1 to 9.

Mathematically,

$$k = \begin{cases} n, & if\ k > n \\ k, & Otherwise \end{cases}$$

7.3.2.3 Weighting Function:

This parameter selects the weighting function from user. It has following weighting formulae available:

1. Absolute Distance

2. Euclidian Distance

3. 3-Norm Manhattan Distance

4. 4-Norm Manhattan Distance

Mathematically,

$$Weighting\ Function = \begin{cases} 1, & if\ 'Absolute\ Distance' \\ 2, & if\ 'EuclideanDistance' \\ 3, & if\ '3-Norm\ Manhattan\ Distance' \\ 4, & if\ '4-Norm\ Manhattan\ Distance' \end{cases}$$

7.3.2.4 Error Acceptance Value (EAV):

This parameter selects the error acceptance value from user. Error acceptance value gets maximum error value, which system should achieve. It accepts positive integer value from user. After setting these parameters it asks user to select training file or enter training file path. This file should be in correct format. After entering file path, system checks following conditions in sequence:

1. Check availability of file.
2. If file is available then check its format.
3. If format is correct then checks format of every data-entry.

If any error occurred then it prints proper error message and asks user to select file again. Perform generalization and filter on training data. Generalization changes the packet types from sub-attacks to attack and stores the data-entries back into training data. Filter delete duplicate entries from training data and store it back into training data. After setting all parameters and generalization and filtering training algorithm is executed

7.3.3 Training Algorithm:

Steps

1) Dataset was studied.
2) The features of the packet were recorded.
3) Considered the deviation of feature values with the values of normal packets.
4) Classified the data into normal Probe and DoS
5) Stored the trained dataset into log file for future used.

7.3.3 Testing Algorithm:

Steps

1) New Dataset was studied.
2) The features of the packet were recorded.
3) Considered the deviation of feature values with the values of training dataset.
4) Classified the data into normal Probe and DoS depending on training.
5) Stored the trained dataset into log file for future used.

7.3.2 Decision Tree Algorithm

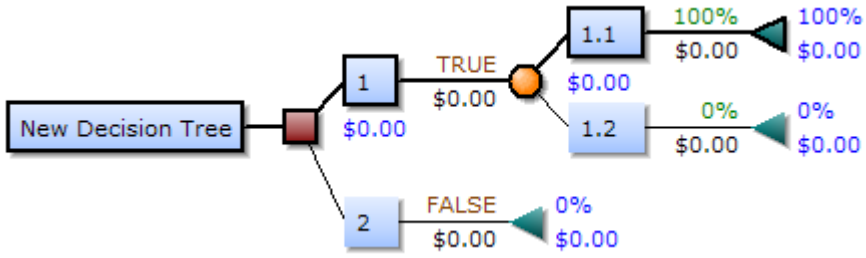
7.3.2.1 Introduction

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal. If in practice decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probability model as a best choice model or online selection model algorithm. Another use of decision trees is as a descriptive means for calculating conditional probabilities

In decision analysis, a "decision tree" — and the closely related influence diagram — is used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

A decision tree consists of 3 types of nodes:-

1. Decision nodes - commonly represented by squares
2. Chance nodes - represented by circles
3. End nodes - represented by triangles



7.3.2.1 Information Gain

Information gain measures the expected reduction in entropy, or uncertainty

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- Values(A) is the set of all possible values for attribute A, and Sv the subset of S for which attribute A has value v Sv = {s in S | A(s) = v}.
- the first term in the equation for *Gain* is just the entropy of the original collection *S*
- the second term is the expected value of the entropy after S is partitioned using attribute A

It is simply the expected reduction in entropy caused by partitioning the examples according to this attribute. It is the number of bits saved when encoding the target value of an arbitrary member of *S*, by knowing the value of attribute *A*.

8. SOFTWARE TESTING

8.1 Introduction

Software Testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.

Software is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails. Most physical systems fail in a fixed (and reasonably small) set of ways. By contrast, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible**.**

8.2 Test Specification

8.2.1 Introduction

Software Testing can be defined as: Testing is an activity that helps in finding out bugs/defects/errors in a software system under development, in order to provide a bug free and reliable system/solution.

8.2.2 Test Plan

Case ID	Test case name	Inputs	Expected output	Actual Output	Remark
1	Input training data	Real Time Dataset	Correct data	Correct data	Yes
2	Input training data	Real Time Dataset	Correct data	Incorrect Feature Values	No
3	Input	Real Time	Correct data	Dataset not in	No

	training data	Dataset		correct format	
4	Input training data	Other than Dataset	Correct data	Wrong Dataset	No
5	Selection of Algorithm	Algorithm	Algorithm Selected	Algorithm Selected	Yes

Table 8.1 Test Plan

8.2.3 Functional Testing

Functional testing is performed on the project management software. The functional testing allows us to test the various functionality of the software.

Black box testing takes an external perspective of the test object to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid input and determines the correct output. There is no knowledge of the test object's internal structure. The software is tested manually.

8.2.4 Function Point Analysis

Function point analysis for project management software which provides a mechanism that both software developers and users could utilize to define functional requirements. It determines a best way to gain an understanding of the user's needs.

- **Data Functions**

Internal Logical files:

The internal logical files allow the user to maintain the data in the database that is provided by the user from a user interface of from the external by system (ILF).

External Interface files:

The External Interface files are one in the data will be stored in the other system and the user is not responsible for maintaining the data in the database (EIF).

- **Transactional Functions**

External Inputs

External Input is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application (EI).

External Outputs

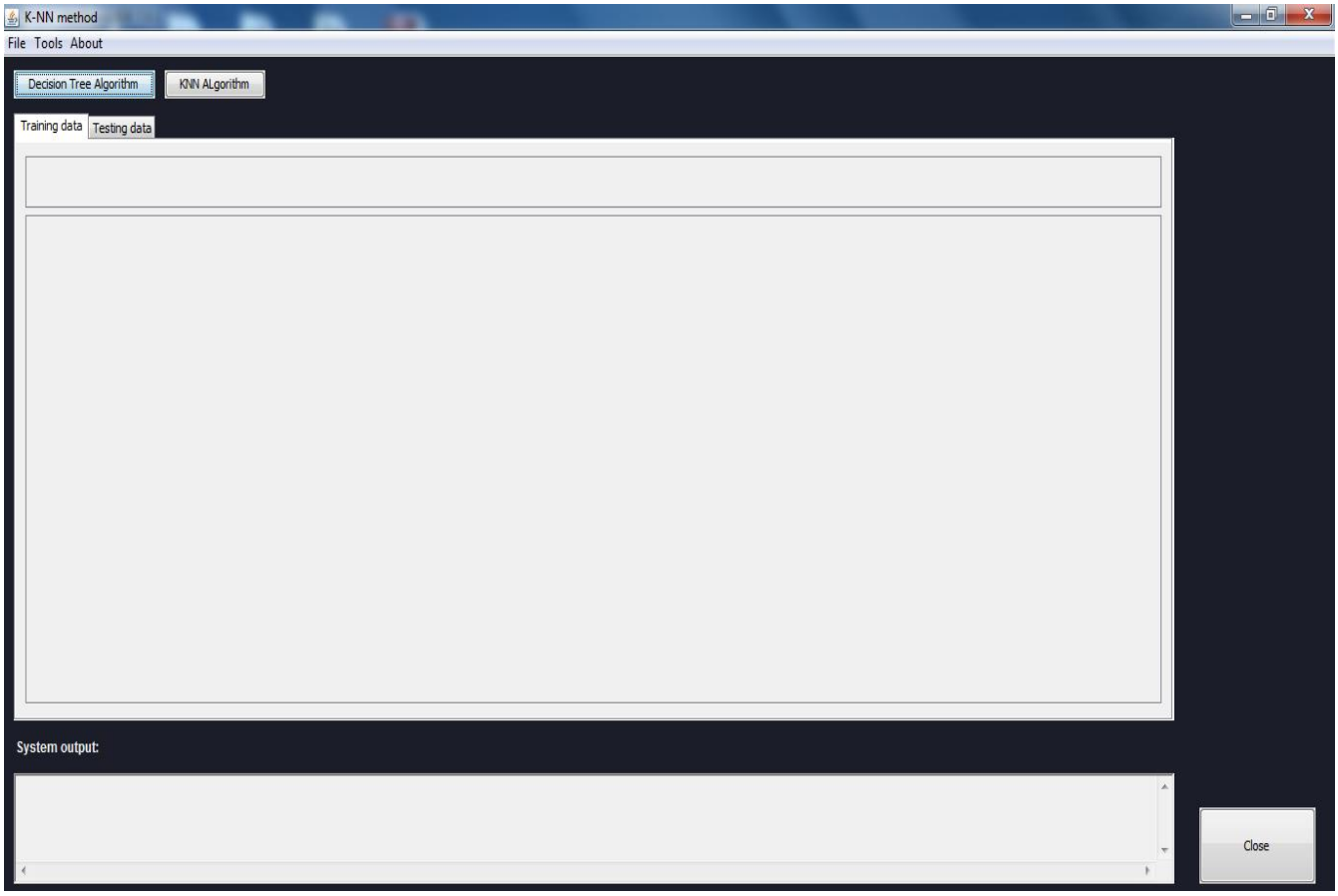
External Output is an elementary process in which derived data passes across the boundary from inside to outside (EO).

External Inquiries

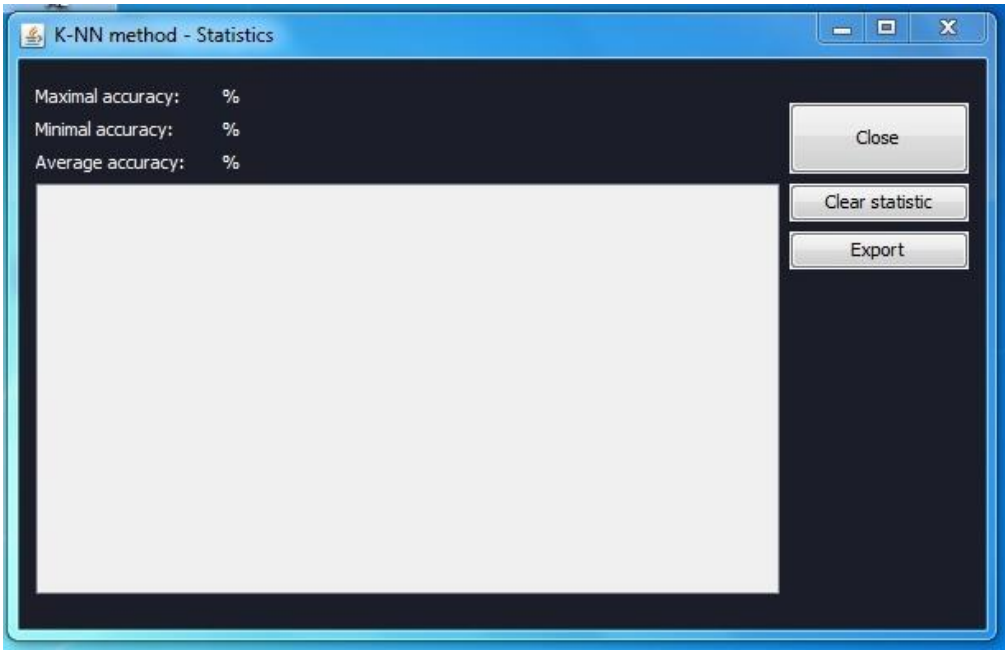
External Inquiries is an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files (EQ).

9. RESULTS

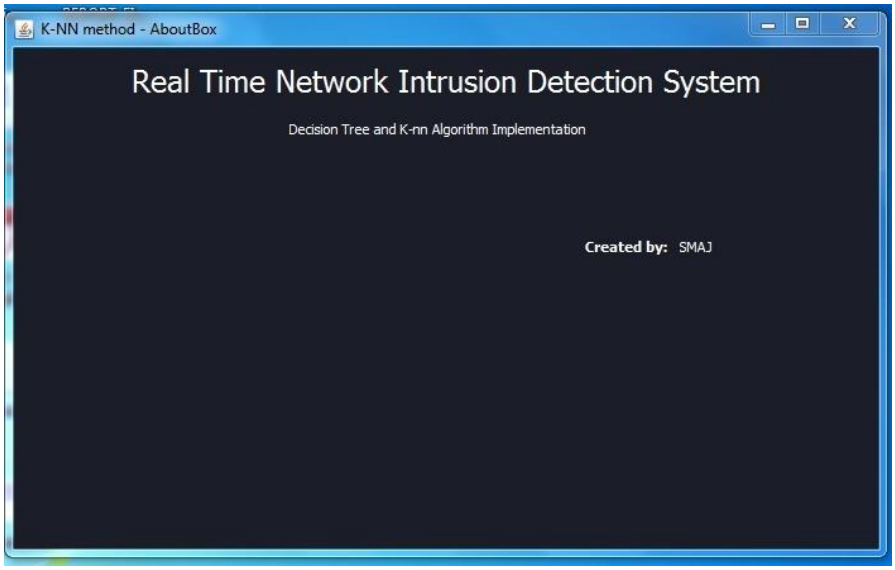
1. Main Window:



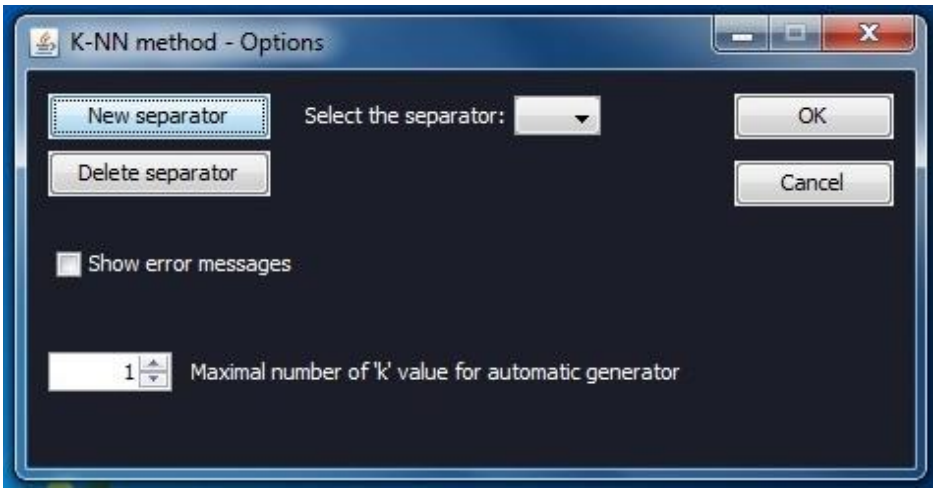
2. Statistics Window:



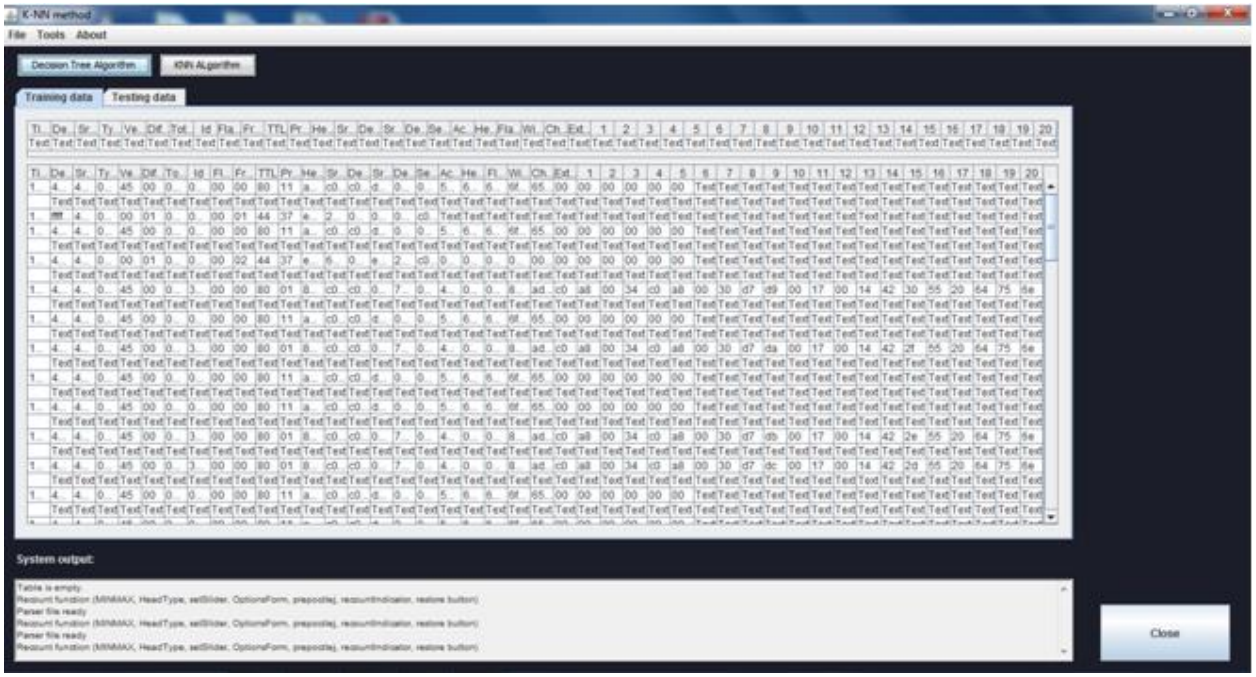
3. About –Help Box:



4. Options Pane:



5. Training/Testing File:



10. DEPLOYMENT AND MAINTENANCE

10.1 Installation and Un-Installation

- Wireshark would have to be installed and un-installed in the regular way.
- Java Setup also shall require the same technique.
- Our software would run through an executable jar file. Just double click on it and run. This makes it very user-friendly

10.2 User Help

10.2.1 Getting Started

Compile and Run the GUI main program. The welcome page will appear with java at the back. The welcome shows the Project title with option “NEXT“ and “EXIT“.

10.2.2 Training/Testing

This page contains two main button “TRAINING” and “TESTING” which will lead you to the training phase and testing phase respectively. There is “BACK” and “EXIT” button on each page.

10.2.3 Training and Testing Algorithm

The files to be trained and tested can be imported. Similarly, the tested and trained files can be exported and saved to be used later on.

10.2.4 Result

This page displays result in tabular view. To determine whether the packet is normal or intrusion packet.

10.2.5 Exit

User Can exit the system at any point of time.

11. CONCLUSION AND FUTURE SCOPE

11.1 Conclusion:

In our project, we use different machine learning algorithms to model Intrusion Detection System (IDS) to improve detection rate and reduce False Alarm Rate (FAR). With the help of k-NN and Decision Tree, we have developed our algorithms and studied their performance. The performance is classified in terms of accuracy, detection rate, False Alarm Rate (FAR) and accuracy for four categories of attack and normal data. Our aim is to reduce False Alarm Rate (FAR) and increase the detection rate compared to results of IEEE papers we had selected in our literature survey, in which we have succeeded.

11.2 Future Scope

Future works of our project are:

- Use our system for real time application in networking.
- Use the system for different datasets format.
- Use the system for detecting new attacks.
- Use more Machine Learning Algorithms to improve accuracy.

12. REFERENCES

[1] C. Jirapummin, N. Wattanapongsakorn, J. Kanthamanon, Hybrid neural networks for intrusion detection system, in: The International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC),Thailand, 2002, pp. 928–931.

[2] W. Lee, S. Stolfo, K. Mok, Mining in a data-flow environment: experience in network intrusion detection, in: The 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '99), San Diego, 1999.

[3] Z. Pan, S. Chen, G. Hu, D. Zhang, Hybrid neural network and C4.5 for misuse detection, in: The 2nd International Conference on Machine Learning and Cybernetics, China, 2003, pp.2463–2467.

[4] M. Moradi, M. Zulkernine, A neural network based system for intrusion detection and classification of attacks, in: The IEEE International Conference on Advances in Intelligent Systems Theory and Applications, Luxembourg, 2004,pp. 148–153.

[5] N. Ngamwitthayanon, N. Wattanapongsakorn, C. Charnsripinyo, D.W.Coit, Multi-stage network-based intrusion detection system using back propagation neural networks, in: Asian International Workshop on Advanced Reliability Modeling (AIWARM), Taiwan, 2008, pp. 609–619.

[6] A. Abraham, R. Jain, Soft computing models for network intrusion detection systems, in: Knowledge Discovery, Computational Intelligence, vol. 4, Heidelberg, 2005, pp. 191–207.

[7] N. Liao, S. Tian, T. Wang, Network forensics based on fuzzy logic and expert system, Computer Communications 32 (2009) 1881–1892.

[8] S. Pukkawanna, V. Visoottiviseth, P. Pongpaibool, Lightweight detection of DoS attacks, in: The IEEE International Conference on Networks (ICON), Australia, 2007, pp 77–82.

[9] C-M. Chen, Y-L. Chen, H-C. Lin, An efficient network intrusion detection, Computer Communications 33 (2010) 477–484.

[10] K. Labib, R. Vemuri, NSOM: a real-time network-based intrusion detection system using self-organizing maps, Networks and Security (2002).

[11] R.S. Puttini, Z. Marrakchi, L. Me, A Bayesian classification model for real-time intrusion detection, in: The 22nd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering. AIP Conference Proceedings, vol. 659, 2003, pp. 150–162.

[12] M. Amini, A. Jalili, H. Reza Shahriari, RT-UNNID: a practical solution to realtime network-based intrusion detection using unsupervised neural networks, Computer & Security 25 (2005) 459–468.

[13] M-Y. Su, G-J. Yu, C-Y. Lin, A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach, Computers and Security 28 (2009) 301–309.

[14] Z. Li, Y. Gao, Y. Chen, HiFIND: a high-speed flow-level intrusion detection approach with DoS resiliency, Computer Networks 54 (2010) 1282–1299.

[15] S. Chakrabarti, M. Chakraborty, I. Mukhopadhyay, Study of snort-based IDS, in:International Conference and Workshop on Emerging Trends in Technology (ICWET), Mumbai, India, 2010, pp. 43–47.

[16] M. Panda, M.R. Patra, Semi-Naı¨ve Bayesian method for network intrusion detection system, Neural information processing, Lecture Notes in Computer Science (Springer Link) 5863 (2009) 614–621.

[17] P. Sangkatsanee, N. Wattanapongsakorn, C. Charnsripinyo, Network intrusion detection with artificial neural network, decision tree and rule based approaches, in: The International Joint Conference on Computer Science and Software Engineering, Thailand, 2009.

[18] P.N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Pearson Addison Wesley, 2005.

[19] Weka 3.6.0 tools [Online]. <http://www.cs.waikato.ac.nz/ml/weka/>.

[20] Jpcap library [Online]. <http://jpcap.sourceforge.net/>.

[21] Mark A. Kon1,Leszek Plaskota: Neural Networks, Radial Basis Functions, and Complexity

[22] Adrian G. Bors: Introduction to Radial Basis Function(RBF) Neural Networks

[23] Ming-Yang Su, Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers

[24] Phurivit Sangkatsanee, Naruemon Wattanapongsakorn, Chalermpol Charnsripinyo, Practical real-time intrusion detection using machine learning approaches

[25] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, Wei-Yang Lin Intrusion detection by machine learning: A review

Appendix A : Project workstation selection, installations along with setup and installation report preparations.

1.1 Operating Environment

The target operating system of IDS is Linux and Windows. The solution should be developed such that it can smoothly run on several different distributions of Linux and Windows Operating Systems.

1.2 Design and Implementation Constraints

Processing Power:

IDS requires high speed processing machine is required to fulfill all the tasks.

Deployment Point:

We will get data from the real-time sources like a packet sniffer.

Detection/False Alarm Rate:

Detection and false alarm rates depend on the choice of algorithm from user. With detections, come a number of false alarms as well. There will be further release of IDS in future that will improve these parameters in future.

Operating Platform:

IDS will work well for Windows and Linux.

1.3 Assumptions of Minimum Requirements:

- Intel Pentium IV
- Hard Disk Space : 40 GB
- Monitor : 14” Inches

- Keyboard : 104 Keys
- Internal Memory : 256 MB RAM
- Mouse : Optical Mouse

1.4 Dependencies:

Packet Sniffer for Windows: Wireshark

1.5 Tools:

We use the following tools for the project:

a. Wireshark (Ethereal):

We use this tool for capturing real time network data packets. It is:

- Network protocol analyzer
- Open Source
- Easy to install, use, manage

b. Tools for generation of attacks

The attacks that we studied will have to be practically implemented to be able to study the characteristics well and also to train our IDS against such attacks in future.

The tools for generation of these attacks are as listed below:

For DoS attacks:

- LOIC
- Commands at the command prompt

For Probe attacks:

- IP Sweep
- PortSweep

c. Java:

- JDK: 1.6.0_02 or higher

- Netbeans 6.5.1 Editor

d. Weka:

- Machine learning tool
- Open Source
- Easy to install, use, manage
- Already implemented algorithms to cross check the system results
- Large number of acceptable file formats

2. Programming of the project functions, interfaces and GUI (if any) as per 1st Term term-work submission using corrective actions recommended in Term-I assessment of Term-work.

2.1 Programming of the project functions:

The major project functions are:

2.1.1 Packet Capturing:

a. Packet sniffing:

- Packet Sniffing is used to detect packets that are incoming to host
- The open source tool i.e. Wireshark (Ethereal) – the network protocol analyzer is used to capture the real-time traffic across the network.

b. Extract packet features:

- Out of all the features that are there in the packet header, only some suitable features are needed.
- These features have to be extracted so that they can be further normalized and processed.

2.1.2 Training the System:

a. Train the i/p data with known answer classes

- As we are using machine learning approach, we need to train the system for the normal behavior of the system.
- So that, it can report an alarm whenever any deviations from it are noticed.
- The system is fed with the input data with known answer classes and hence it gets trained for it.

b. Test the input data

c. Algorithms:

The following predictive classification algorithms are been implemented for the training of the data:

- Decision Tree
- K-Nearest neighbor
- Adaboost
- Back-Propagation neural networks

2.1.3 Testing the System:

a. Use unknown data:

- After the system is trained, it can be fed with the newer type of input data/unknown data which is without any answer class.

b. Test the data and detect the intrusion, if any:

- The data is tested and if intrusions are found, they are reported using the experience of the system.

c. Compare the results of all algorithms:

- The results obtained from all the algorithms are collected and compared.

2.1.4 False Alarm Detection and Reduction

a. Analyze performance of system:

The performance of the system is analyzed in the post-processing phase.

b. Compare results of algorithms for the false alarms

The results obtained from various algorithms are compared and the False Alarm Rates are calculated for each one.

c. Reduce the false alarm rate

By training the system for newer type of data, the accuracy of the system increases and also the False Alarm Rate gets reduced**.**

2.2 Interfaces:

2.2.1 User interfaces:

User Interface will be a GUI application developed in Java language satisfying all required elements to interact with system.

2.2.2 Hardware interfaces:

- Host:
  - Internal computer sources
  - Ex: OS audit and system logs
- Network
  - Packets via “sniffing”
- Target-based
  - Monitor object for changes e.g. Tripwire

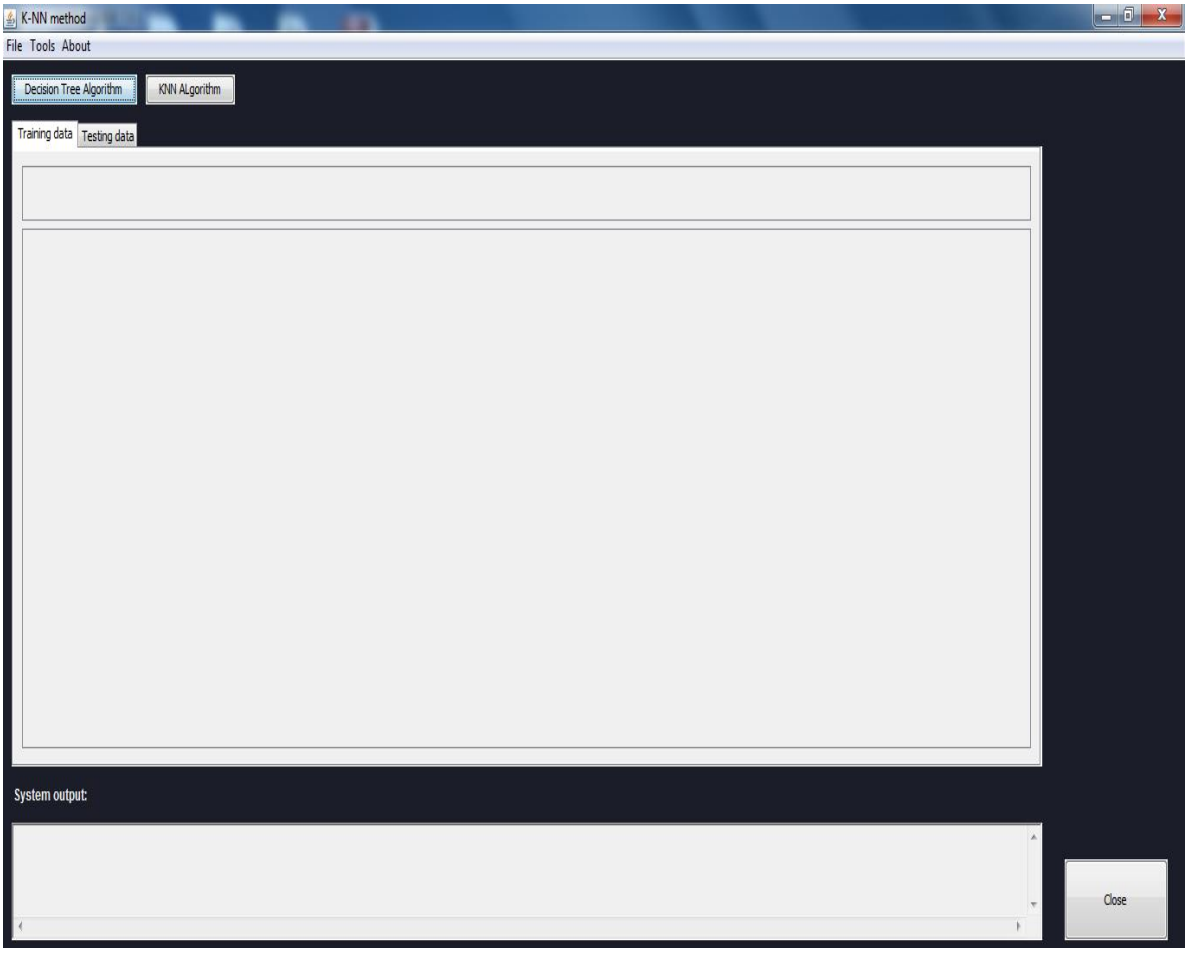
2.2.3 Software interfaces:

- Sniffers
  - collect data from various sources such as log files, network packets
  - sends them to the analyzer
- Analyzers
  - process data from sensors and determine if intrusion has occurred
  - may also provide guidance for the actions to take.

2.2.4 Communication interfaces:

- Graphical User interface
    - Select operations
    - view the output and manage the behavior

2.3GUI



3. Test tool selection and testing of various test cases for the project performed and generate various testing result charts, graphs etc. including reliability testing.

3.1 The testing of the project is done manually. No testing tool is used for the same. The test-cases are very clear and thus they can be tested for very easily even when we test them manually.

3.2 Test-cases:

Test Case No.	Unit to test	Test Data	Steps to be executed	Constraint	Expected Result	Pass/Fail	Comment
1	Dataset	Data in format of wireshark packet file	Input dataset and click button	Only 10% data taken	Data read successful	Pass	Working properly
2	Dataset	Data in format of wireshark packet file	Input dataset and click button	Only 10% data taken	Data read unsuccessful	Pass	Working properly
3	Training phase	Training Data (File)	Start training phase and click button	Part of 10% data taken	Training phase completed successfully	Pass	Working properly
4	Testing phase	Testing Data (File)	Start testing phase and click button	Remaining data of 10% taken	Testing phase completed successfully	Pass	Working properly

3.3 Testing has to be done to improve performance of the system and the following factors take care of the performance:

3.3.1 Obtain highest detection rate

True Positive: attack data detected as attack data.

True Negative: normal data detected as normal data**.**

3.3.2 Reduce FAR

False Positive: normal data is detected as attack data.

False Negative: attack data is detected as normal data

3.3 Reliability Testing:

Reliability is one of the most important elements of test quality. It has to do with the consistency, or reproducibility, of an examinee's performance on the test. If a test yields inconsistent scores, it may be unethical to take any substantive actions on the basis of the test. There are several methods for computing test reliability including test-retest reliability, parallel forms reliability, decision consistency, internal consistency, and interrater reliability. For many criterion-referenced tests decision consistency is often an appropriate choice

The system is checked for reliability by using it under different conditions, by feeding it with various types of data. Also, the amount of data i/p to the system is also varied and the system is tested. The system is proved to be reliable, as the MTTF (Mean Time to Fail) is very high and the MTTR (Mean Time to Recover) is very low.