ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

A COMBINED REASONING SYSTEM FOR KNOWLEDGE BASED NETWORK INTRUSION DETECTION

BY

MESERET ASSEFA ADAMU

JUNE 2016

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

A COMBINED REASONING SYSTEM FOR KNOWLEDGE BASED NETWORK INTRUSION DETECTION

A THESIS SUBMITTED TO SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY

MESERET ASSEFA ADAMU

JUNE 2016

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

A COMBINED REASONING SYSTEM FOR KNOWLEDGE BASED NETWORK INTRUSION DETECTION

BY

MESERET ASSEFA ADAMU

Name and Signature of Members of the Examining Board

Name	**Title**	**Signature**	**Date**
_____	Chairperson	_____	_____
Dr. Million Meshesha	Advisor	_____	_____
Dr. Workshet Lamenew	Examiner	_____	_____
Dr. Wondwossen Mulugeta	Examiner	_____	_____

DECLARATION

I, the undersigned, declare that this thesis is my original work and has not been presented as a partial degree requirement for a degree in any other university and that all sources used for the thesis have been duly acknowledged.

Meseret Assefa Adamu

2016

The thesis has been submitted for examination with my approval as University Advisor

---

Dr. Million Meshesha

2016

DEDICATION

It is a lifetime opportunity for me to dedicate this thesis work to my father Ato Assefa Adamu, my mother W/r Tewabech Gella and my brother Eshetu Assefa for their love of education, guidance, motivation and scarification for me to reach at this level. Dear Father, may God rest your soul in peace. Your love and motivation is always with me. Dear Mother, I know all the ups and downs you passed through, your pray and strength makes me strong always. Long live for you! Dear Esha, I have no words to say about your effort for the family. You are my role model, motivation and teacher. It is a great opportunity for me to have such a blessed brother. Thanks God and stay blessed!

Acknowledgment

Before all, I praise the almighty **GOD** and his mother **ST. MARY** for making everything the way it is. Next, I am so glad to express my warm gratitude to my advisor **Dr. Million Meshesha** for being with me in all the ups and downs of this work. He is honestly the GREATEST teacher and advisor I have ever known. Thank you so much for showing and guiding me to an interesting research direction and recognizing my potential in addition to the care, support and valuable comments you gave me. What you did is beyond acknowledgment.

Special thanks to Dr. Solomon, Dr. Martha, Ato Getachew J., Dr. Dereje, Dr. Tibebe, Ato Ermias, Meseret, Adey, Kalkidan and to all staff member of School of Information Science.

Many thanks to **Dawit Hassen**, for providing me all the resources, the precious advices and supporting ideas you gave me.

I would like to thank to all staff members of Addis Ababa University ICT office for their support on my thesis work, especially **Minilik Tesfaye,** Thank you for your support and stay blessed.

Dr. Ahmed Hasen, Daniel Mamo, Abeba Adinew, Hailemariam Negussie, Eyob, Mahlet, Almaz, Messay, Genet, Ahmed Z., Dr. Desalegn, Aklilu, Nuredin, Tirsit and all staff members of IES, Please accept my acknowledgment.

My sincere thanks goes to my spiritual father, Melake Brehanat Abba Petros Solomon, your praise, love of education, precious advices and motivation always push me forward. Your spirit will be with me forever.

I would like to offer special thanks to all my family members specially **Eshetu Assefa**, **Melaku Mamo**, and **Yeshiber Assefa** your motivation and follow-up gives me strength.

My friends, Fire(shuluka), Kore, Abrish(Mushiraw), Sol, Gize(Girma), Zewongel, Meheret, Marshet, Miki,Yeshi and others for your encouragement, love and support with all you have, I have nothing except THANK YOU.

I am also happy to say thank you for my elementary, high school and University friends and teachers.

****

〈〈አሐደ ቀለመ ዘነገረከ ይኩን ከመአቡከ፤ወክልዔተ ቀለማተ ዘነገረከ ይኩን ከመ ፈጣሪከ።።〉〉

አንድ ቀለም የነገረህ እንደ አባትህ ይሁን፤ሁለት ቀለማትን የነገረህም እንደ ፈጣሪህ ይሁን።

****

Meseret Assefa

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CBNIDS: Case Based Network Intrusion Detection System

CBR: Case Based Reasoning

CLI: Command Line Interface

CRN: Case Retrieval Nets

DARPA: Defense Advanced Research Project Agency

DOS: Denial of Service

EM: Expectation-Maximization

FN: False Negative

FP: False Posistive

GUI: Graphical User Interface

HIDS: Host Based Intrusion Detection System

ICT: Information Communication Technology

IDPS: Intrusion Detection and Prevention Systems

IDS: Intrusion Detection System

IE: Inference Engine

IPS: Intrusion Prevention System

JRE Java Runtime Environment

KBS: Knowledge Based System

KDD: Knowledge Discovery in database

K-NN: K-Nearest Neighbor

LAN: Local Area Network

MIT: Massachusetts Institute of Technology

NIDS: Network Based Intrusion Detection System

RBR: Rule Based Reasoning

TN: True Negative

TP: True Positive

Abstract

Nowadays, the Internet plays a vital role in incessant communication; its effectiveness however can diminish owing to effects called intrusions. Intrusion is an activity that adversely affects the targeted system. There are different ways of detecting and preventing intruders in the network. Knowledge Based System (KBS) is the widely used one with rule-based reasoning or case-based reasoning.

In this study, a combination of rule based and case based reasoning for network intrusion detection is proposed. To this end, knowledge is extracted using data mining from sampled KDDcup'99 intrusion data set. Both descriptive and predictive models are created using K-means clustering and JRip rule induction. Descriptive model is used to design case-based reasoning and predictive model to construct rule-based reasoning. The method of combination used is a conditional combination model, which has a controller in between RBR and CBR. The controller is developed by Java eclipse programming language. In the combined system, it is the RBR that first treat the new query for recommending a solution. Otherwise, the query is automatically forwarded to the CBR system where the case retrieval module identifies the most related solution using case similarity measure. The combination of rule-based and case-based reasoning methods has shown a substantial improvement with regards to performance over the individual reasoning methods. The combined system scores 93.33% overall performance and achieves 90.5% accuracy with an average Precision and Recall of 90% and 91% respectively. The user acceptance testing also resulted 88% this is a very good acceptance. This shows the system has registered a promising result to come up with an applicable system. But, further exploration has to be done to refine the knowledge base and boost the advantages of combining CBR with RBR.

Keywords: - *Intrusion detection, knowledge based system, combined Intrusion detection, combination of CBR and RBR, knowledge-based intrusion detection, combined reasoning system.*

CHAPTER ONE

INTRODUCTION

1.1. Background

In the last few years, the networking revolution has finally come of age. More than ever before, we see that the Internet is changing computing, as we know it [1]. The possibilities and opportunities are limitless; unfortunately, so too are the risks and chances of intrusions. It is very important that the security mechanisms of a system are designed so as to prevent unauthorized access to system resources and data.

Computer Security is the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources such as hardware, software, firmware, information/data, and telecommunications [1].

These days most organizations are dependent on the Internet to communicate with people and systems to provide news, online shopping, email, credit card detail and personal information [2]. Due to the rapid growth in technology and widespread use of the Internet, a lot of problems have been faced to secure the system's critical information within or across the networks because there are millions of people attempting to attack on systems to extract critical information [2] [3]. To defend against various cyber-attacks and computer viruses, lots of computer security techniques have been intensively studied in the area of cryptography, firewalls, anomaly and intrusion detection. However, completely preventing breaches of security appear, at present, unrealistic. We can, however, try to detect these intrusion attempts so that action may be taken to repair the damage later. This field of research is called Intrusion Detection [1].

Among the Intrusion detection techniques, network intrusion detection has been considered to be one of the most promising methods for defending complex and dynamic intrusion behaviors [8]. As firewalls and anti-viruses are not enough to provide full protection to the system, organizations have to implement the Intrusion Detection System (IDS) to protect their critical information and computational resources against various types of attacks [3].

Usman et al. [3], define intrusion as, an attempted act of using computer system resources without privileges, causing incidental damage. Intrusion Detection means any mechanisms which detect the intrusive behavior. IDS monitor network traffic and its suspicious behavior against security. According to Sumit et al. [4], nowadays IDS perform signature-based monitoring of the cyber infrastructure to identify any malicious activities and generate an alert when such activity is detected. IDS monitor network traffic and its suspicious behavior against security. If it detects any threat then alerts to the system or network administrator about intrusion. IDS are a set of techniques and methods that are used to detect suspicious activities both at the network and host levels [3].

Intrusion Detection System can be classified into three categories [5]. The first is Host-Based IDS that evaluates information found on a single or multiple host systems, including contents of operating systems, system and application files. The second is Network-Based IDS, which evaluates information captured from network communications, analyzing the stream of packets traveling across the network. Packets are captured through a set of sensors. The final one is Vulnerability-Assessment IDS, which detects vulnerabilities on internal networks and firewalls [6]. According to Fayyad et al. [6], there are two primary models to analyze events and detect attacks: Misuse detection and Anomaly Detection Models. Misuse detection is an IDS that detect intrusions by looking for activity that corresponds to known signatures of intrusions or vulnerabilities. On the other hand, Anomaly Detection Model is an IDS that detect intrusions by searching abnormal network traffic. Data mining based methods are another paradigm for building intrusion detection system. The main advantage of these methods is that they leverage the generalization ability of data mining method and in order to detect new and unknown attacks. A data mining based IDS uses machine learning and data mining algorithms on a large set of system audit data to build detection models [8]. Hyt et al. [8], noted that intrusion detection techniques using data mining have attracted more and more research interests in recent years.

Data mining is the task of discovering interesting patterns from large amounts of data, where the data can be stored in databases, data warehouses, or other information repositories [7]. According to Fayyad et al. [6], data mining is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable hidden patterns in data. Data mining has two major tasks predictive modeling and descriptive modeling. As an important application area of data mining, data mining meliorates the great burden of analyzing huge volumes of audit data and realizing performance optimization of detection. Many data mining systems are great in

deriving useful statistics and patterns from huge amounts of data, but they are not very smart in interpreting these results, which is crucial for turning them into interesting, understandable and actionable knowledge [9].

A knowledge Based System (KBS) is a computer system which represents knowledge about a specific problem domain and this knowledge can solve the problems from the problem domain [2] [11]. KBS is a sub-field of Artificial Intelligence (AI) that works on knowledge base for effective decision making by imitating the behavior of human expert within a well-defined, narrow domain of knowledge [10]. KBS has come across a variety of approaches based on the knowledge representation methods and the reasoning strategies applied during implementation. Rule based reasoning (RBR) and case based reasoning (CBR) are two popular approaches used in knowledge based systems [12]. According to Prentzas [12], the main advantages of RBR are compact representation of general knowledge, naturalness of representation, modularity and provision of explanations. However RBR, has some drawback such as knowledge acquisition bottleneck, brittleness of rules, inference efficiency problems, difficulty in maintenance of large rule bases, problem solving experience is not exploited and interpretation problems. On the other hand, Case Based Reasoning has an inability to express general knowledge. Cases, by nature, express specialized knowledge. So, they cannot express general knowledge. This is a disadvantage compared to rule-based systems [12].

As noted in [12], the combination of (two or more) different problem solving and knowledge representation methods is a very active research area in Artificial Intelligence. The aim is to create combined formalisms that benefit from the strength of each method. It is generally, believed that complex problems are easier to solve with hybrid or integrated approaches. The effectiveness of various hybrid or integrated approaches has been demonstrated in a number of application areas. One of the popular types of combination involves the combination of rule-based reasoning (RBR) and case-based reasoning (CBR).

The motivation for this research is as stated by Prentzas [12], the efforts to combine symbolic rules and cases have yielded advanced knowledge representation formalisms. The effectiveness of those approaches stems from the fact that rules and cases are complementary in representing application domains and solving problems. Rules represent general knowledge of the domain, whereas cases represent specific knowledge. Rule-based systems solve problems from scratch, while case-based systems use pre-stored situations to deal with similar new instances.

Further, Azeb [28] on her work combine rule based reasoning with an existing case based reasoning for oil Drilage Company. So, on her finding she stated that "the combination of case based reasoning and rule based reasoning improve the accuracy of the decisions of the company."

Therefore, the combination of both approaches turns out to be natural and useful. The complementarities of the advantages and disadvantages of the two knowledge representation and reasoning approaches intensify the usefulness of their combination. Using the complementarities nature of the two reasoning systems for knowledge based network intrusion detection is the main motivation for this research.

1.2. Statement of the problem

Due to an increase in connectivity via the Internet, and the vast spectrum of financial possibilities that are opening up, more and more systems are subject to attack by intruders. These subversion attempts try to exploit flaws in the operating system as well as in application programs and have resulted in spectacular incidents. Some specific examples of intrusions that concern system administrators include [3]:

- Unauthorized modifications of system files so as to facilitate illegal access to either system or user information;
- Unauthorized access or modification of user files or information;
- Unauthorized modifications of tables or other system information in network components (e.g. modifications of router tables in an internet to deny use of the network);
- Unauthorized use of computing resources (perhaps through the creation of unauthorized accounts or perhaps through the unauthorized use of existing accounts).

Intrusion detection systems are hardware and software systems that monitor events occurred on computers and computer networks in order to analyze security problems. The number and severity of these attacks has been increasing continuously. Consequently, IDSs have become an integral part of the security infrastructure of organizations [5]. Intrusions to computer networks are called as ‘‘attacks" and these attacks threaten the security of networks by violating privacy, integrity and accessibility mechanisms. Attacks can be originated from users who login to the computer using the Internet trying to gain super user or administrator rights and other users who misuse the rights they have. IDSs automate monitoring and analyzing the attacks [3] [4] [5].

There are a number of researches that have been done on knowledge based IDS and Data Mining independently [4] [5], but integrating data mining and knowledge based system are a new research area. Abdulkerim [7], attempted to integrate data mining with knowledge based system. He conducted data mining experiment using JRip decision tree algorithm and design the knowledge based by using rule based knowledge representation method. Dawit [13], further conducted a research on the same area by using Case Based Reasoning method. According to Prentzas [12], the rule based approach to the development of knowledge based system has drawback such as; it is not possible to draw conclusions from rules when there are missing values in the input data. For a specific rule, a certain number of condition values must be known in order to evaluate the logical function connecting its conditions. In addition, rules do not perform well in cases of unexpected input values or combinations of them. On the other hand, Case Based Reasoning has an Inability to express general knowledge. Cases, by nature, express specialized knowledge. So, they cannot express general knowledge. This is a disadvantage compared to rule-based systems [7].

Data mining (DM) based methods are another paradigm for building intrusion detection systems. The main advantage of these methods is that they leverage the generalization ability of data mining methods and to detect new and unknown attacks. A data mining-based IDS uses machine learning and data mining algorithms on a large set of system audit data to build detection models. These models have been proven very effective [8]. These algorithms are generally classified as either misuse detection or anomaly detection. Misuse detection algorithms learn how to classify normal and attack data from a set of training data which contains both labeled attack and normal data [8]. Anomaly detection algorithms learn a model of normal activity by training on a set of normal data. Anomaly detection algorithms then classify as an attack activity that diverges from this normal pattern based on the assumption that attacks have much different patterns than normal activity. In this way new unknown attacks can be detected [10].

According to Huy [8] and Aikaterini [11], a DM intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system. According to Tigabu [17], data mining has been known to aid the process of Intrusion Detection and the ways in which the various techniques have been applied to enhance the security of the

network. Generating patterns and knowledge is vital for IDSs to differentiate standard behaviors from strange behaviors.

There are also different works on combining rule based and case based reasoning to improve the performance of their systems [12]. But they are not integrated with data mining. The attempts simply integrate the two reasoning‘s by acquiring knowledge from the domain experts. But they are not exploiting the potential of Data Mining for extracting hidden knowledge.

Therefore, this study focuses exploring a way to automatically use a knowledge acquired using descriptive and predictive mining models for network intrusion detection systems that combines case based reasoning and rule-based reasoning.

To this end, this study explores and finds answers to the following research questions:-

- How to acquire rules and cases from the dataset using data mining models?
- How to select the best combination model that performs better in detecting intruders?
- How to combine rule-based reasoning and case-base reasoning for constructing combined reasoning system for knowledge based NID?
- To what extent does the combined reasoning system for knowledge based network intrusion detection identify unknown attacks and minimize false alarm rates?

1.3. Objective of the study

1.3.1 General Objective

The general objectives of this study is to explore, design and develop a knowledge based network intrusion detection system that combines case-based reasoning and rule-based reasoning which automatically use a knowledge acquired using descriptive and predictive mining models.

1.3.2 Specific Objectives

The specific objectives of this study are the following:-

- To apply different DM algorithms and select suitable data mining classification and clustering algorithms for constructing descriptive and predictive models.
- To understand methods, approaches, techniques and tools from literatures.
- To acquire cases and rules from descriptive and predictive models.

- To identify and select the best combination model of RBR and CBR.
- To propose framework for conditional combination of RBR and CBR.
- To develop a prototype for the combined reasoning system.
- To evaluate the performance of the prototype and its user acceptance.

1.4. Scope and limitations of the Study

This research is basically the integration of the work of Abdulkerim [7] and Dawit [13] in the area of NIDS by combining data mining and knowledge base system. On top of their work both Abdulkerim [7] and Dawit[13] conduct their research by integrating different data mining models (predictive and descriptive respectively) with different KBS approaches (rule based and case based respectively). Therefore, this study conducted on the combination of the two authors work.

For integrating data mining with a combined approach for network intrusion detection, hidden knowledge is automatically acquired from intrusion dataset collected from KDDcup'99. The KDDcup '99 dataset was created by processing the tcpdump portions of the 1998 DARPA intrusion detection system evaluation dataset, created by Lincoln Lab under contract to DARPA. For knowledge acquisition based on data mining technique, different techniques is used in order to identify the natural categories of instances of network intrusion. The acquired knowledge using data mining technique is used to develop a combined network intrusion detection system.

Due to shortage of time and privacy concern of an organizations, this study did not include real life network, and advisory facility for the network administrator not included.

1.5. Significance of the study

The general objective of this study is to integrate data mining with knowledge based system in order to develop a combined network intrusion detection system by the combining rule-based reasoning (RBR) and case-based reasoning (CBR). According to Prentzas [12], efforts to combine symbolic rules and cases have yielded advanced knowledge representation formalisms. The effectiveness of those approaches stems from the fact that rules and cases are complementary in representing application domains and solving problems. Rules represent

general knowledge of the domain, whereas cases specific knowledge. Rule-based systems solve problems from scratch, while case-based systems use pre-stored situations to deal with similar new instances. Therefore, the combination of both approaches turns out to be natural and useful. The complementarities of the advantages and disadvantages of the two intelligent methods intensify the usefulness of their combination.

To this end, as a hypothesis to be tested, this research will provides a combined reasoning system for knowledge based network intrusion detection that will perform better than what has been done before. So, the system will help network administrators at different sectors to take action in accordance of the detected network attack, and the study will be able to enhance intelligence of the knowledge based intrusion detection system and provide idea on the integral of data mining with combined system.

1.6. Methodology of the Study

Methodology is the systematic, theoretical analysis of the methods applied to a field of study [10]. Kumar [10] noted that methodology comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge.

Methods described in the methodology, defines the means or approaches of data collection, algorithm selection, tool and technique selection [10]. To this end, the researchers reviewed different related literatures such as books, journal articles, Conference proceeding papers, and the Internet in order to have detailed understanding on the present research.

1.6.1. Study design

This study is an experimental and qualitative research and in this study, Knowledge Discovery in Databases (KDD) method is followed. In order to make knowledge extraction as much correct as possible (i.e. in order to keep the correctness of the knowledge as it is kept at the source) different techniques would be applied [13]. According to Mihaela et al. [21], among these techniques, data mining techniques and, more general, KDD techniques became the most used in the recent years. KDD is the process of extracting and refining useful knowledge from large data. According to Fayyad et al. [6], KDD is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. KDD refers to the overall process of discovering useful knowledge from data, and data mining refers to a particular step in

this process. Data mining is the application of specific algorithms for extracting patterns from data. Steps in the KDD process, such as data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining, is essential to ensure that useful knowledge is derived from the data [3].

In an attempt to acquire knowledge using KDD, the following seven steps are included in this study; data selection, data preprocessing, data transformation, data mining, evaluation, and interpretation and use of knowledge [13], as shown in figure 1.1.

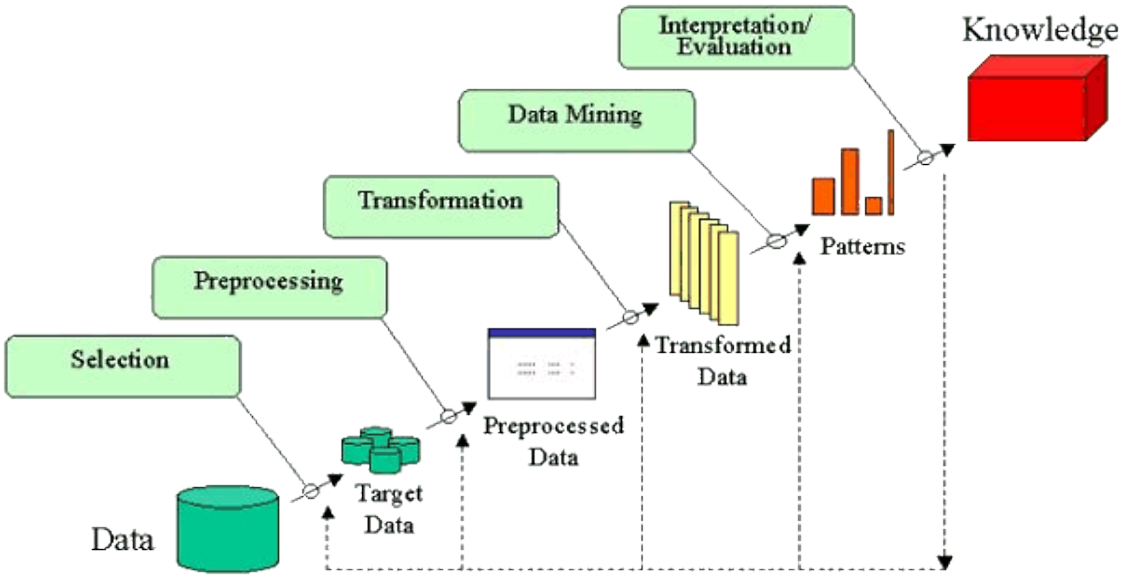

Figure1.1. an overview of steps of KDD (knowledge discovery in database)[17]

1.6.2. Data selection

Data selection is where data relevant to the analysis task are retrieved from the data [13]. According to Abdulkerim[7], Tigabu[17] and Dawit[13], the KDD Cup 1999 intrusion detection contest data (KDD cup 99 Intrusion detection data sets) has been used in this study. This data was prepared by the 1998 DARPA (Defense Advanced research Project Agency) Intrusion Detection Evaluation program by MIT Lincoln Labs (MIT Lincoln Laboratory). The KDDcup'99 data set from Lincoln lab available at ACM Knowledge Discovery web site. Considering the time it takes to process the KDDcup'99 data set and coming up patterns for this study.

1.6.3. Data preprocessing

The data set is preprocessed before the actual mining task is undertaken on the data set. According to Tigabu [17], data preprocessing is an important issue for data mining, as real-world data tend to be incomplete, noisy and inconsistent. Data preprocessing includes data cleaning, data integration, data transformation and data reduction.

KDDcup'99 is about 4 gigabytes of compressed raw (binary) tcpdump data of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with about 100 bytes. The KDDcup'99 data set is labeled with exact one specific attack type i.e., either normal or an attack. Deviations from 'normal behavior', are considered as attacks. Attacks were further classified into four sections represented into DOS (deny of service), Probe (information gathering), U2R (user to root) and U2L (remote to local).

DataPreparator tool is used for data preprocessing. DataPreparator is a free software tool designed to assist with common tasks of data preparation (or data preprocessing) in data analysis and data mining. DataPreparator can assist exploring and preparing data in various ways prior to data analysis or data mining. It includes operators for cleaning, discretization, numeration, scaling, attribute selection, missing values, outliers, statistics, visualization, balancing, sampling, row selection, and several other tasks. DataPreparator is written in Java and requires Java Runtime Environment (JRE) to be installed on a given machine [13].

1.6.4. Data Transformation

The data transformation step of this study includes finding useful features to represent the data, depending on the goal of the task, and using dimensionality reduction or transformation methods to reduce the effective number of variables and instances from the dataset (KDD cup 99) under consideration [17].

1.6.5. Data Mining models

Data mining is the process used to extract knowledge in databases. Data mining process is very much useful for extracting and identifying useful information and subsequent knowledge from databases [8]. Data mining satisfy its main goal by identifying novel valid, potentially useful, and easily understandable correlations and patterns present in existing data. This goal of data

mining can be satisfied by modeling it as either Predictive or Descriptive nature [9].The Predictive model works by making a prediction about values of data, which uses known results found from different datasets. Classification algorithm such as JRip is used for constructing predictive models. The Descriptive model mostly identifies patterns or relationships in datasets. It serves as an easy way to explore the properties of the data examined earlier and not to predict new properties. In this study clustering algorithms such as K-means clustering is used for constructing descriptive models.

The aim of data mining in this research is to finding cases and rules from the KDD cup99 dataset, predictive and descriptive data mining techniques used and the algorithm for both predictive and descriptive that perform better is selected.

In order to get hidden knowledge from the data set and compare the performance of cluster, WEKA 3.7.9 is used. WEKA (Waikato Environment for Knowledge Analysis) is chosen since it is proven to be powerful for data mining and used by many researchers for mining task and the researcher is familiar with the tool. WEKA is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. WEKA is free software available under the GNU General Public License. The WEKA workbench contains a collection of visualization tools and algorithms for data analysis and descriptive modeling, together with 12 graphical user interfaces for easy access to this functionality. The software has Header section and Data section and Supported attributes numeric, nominal, string and date data type.

1.6.6. Testing and Evaluation procedure

The KBS is evaluated based on system performance testing method and user acceptance testing. System performance is evaluated using different techniques such as, Precision, Recall and F-measure.

Precision is the number of class classified correctly over the total number of instances classified as class . Recall, on the other hand measures the number of correctly classified examples relative to the total number of positive examples. F-measure is the harmonic mean of precision and recall.

For user acceptance testing, Addis Ababa University ICT senior Network Administrators are engaged by administering an evaluation sheet containing a series of questions (appendix 3).

1.6.7. Use and combination of knowledge

Case based reasoning and rule based reasoning techniques are two alternative ways of problem solving in intelligent systems. Their knowledge representation and reasoning methods are naturally alternatives [15].

Approaches that combine rules and cases based on component dominance are categorized in to rule dominant, case dominant and balanced approach [12]. According to [12] in rule-dominant approaches, the rule-based component prevails in the inference process, whereas the case-based component plays a complementary role. These approaches usually focus on the rule-based component and invoke the case-based component only when necessary (usually in specialized or exceptional situations). In case-dominant approaches, the case-based component plays a more important role and the rule-based component is less significant (supportive). This scheme is useful, for instance, when the case library contains a limited number of cases. In balanced approaches, the role of the combination components is balanced that is, none of the combined components plays a supportive role.

Therefore, in this research rule-dominant combination approach is used, where the RBR comes first to accept the incoming query and the solution given by the RBR is taken as a correct solution, otherwise the CBR take the turn and diagnose the type of intrusion. The reasons why we select this combination approach is because rules are easy to represent, find exact matches for problems to return solution, easy for machines to process.

1.6.8. Tools and Approaches

Weka

For the purpose of data mining for this study the researcher used Weka. Weka (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. Weka is free software available under the GNU General Public License. The Weka workbench contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality. The software has Header section and Data section and Supported attributes numeric, nominal, string and date data type.

The Main Features of weka 3.7.9:-

- 49 data preprocessing tools
- 76 classification/regression algorithms
- 8 clustering algorithms
- 3 algorithms for finding association rules
- 15 attribute/subset evaluators + 10 search algorithms for feature selection

CBR-Based Framework jCOLIBRI

jCOLIBRI is an object-oriented framework for developing CBR systems It offers an easier development process that is based on the reuse of past designs and implementations. jCOLIBRI formalizes the CBR knowledge using domain independent CBR ontology (CBROnto), which is mapped over the classes of the framework, a knowledge-level description of the CBR tasks and a library of reusable Problem Solving Methods (PSMs) [39]. jCOLIBRI is selected for this study based in several aspects: availability (open source framework), implementation (the Java implementation implies to a great extend usability, extensibility and user acceptance), GUI (the provided graphical tools facilitate the system design). Another justification for this study is connected with the fact that jCOLIBRI affords the opportunity to apply Case Retrieval Nets (CRN) model as Case Base structure [13].

jCOLIBRI is just a white-box tool that permits programmer users to have total control of the internal details of the software. This framework represents the bottom layer of platform and lacks of any kind of visual interface. On the other hand, the top layer includes the graphical development tools to aid users in the development of CBR systems. These tools are enclosed in the COLIBRI Studio IDE and generates applications that use the components provided by jCOLIBRI.

COLIBRI Studio includes the following tools: a Case Designer that enables the definition of the structure of your cases; a tool to configure the storage of cases in textual files; a tool to select the in-memory organization of the case; a tool to define the similarity between cases; and several tools to define the behaviour of the system. Moreover COLIBRI Studio integrates several wizards that ease the design process [2] [13].

JRip Rules Classifiers

JRip (RIPPER) is one of the basic and most popular algorithms. Classes are examined in

increasing size and an initial set of rules for the class is generated using incremental reduced error JRip (RIPPER) proceeds by treating all the examples of a particular judgment in the training data as a class, and finding a set of rules that cover all the members of that class. Thereafter it proceeds to the next class and does t he same, repeating this until all classes have been covered [7].

Java eclipse is also used to develop a user interface integration of CBR and RBR also done by using this software.

1.7. Organization of the Thesis

This thesis is structured into six chapters. The first chapter discusses background to the study problem area, statement of the problem, objective of the study, scope of the study, research methodology and application of the study.

The second chapter related literatures reviewed and discussed about DM and knowledge discovery technology, DM Processes, DM models, application of DM, Knowledge base system and knowledge representation in general and in particular in the area of case based and rule based reasoning, combination of case based and rule based reasoning and intrusion detection and Intrusion detection and prevention principles.

The third chapter deals with understanding the methods and approaches of the system. The proposed system architecture and the description of the system architecture are explained.

The forth chapter deals with understanding the data and dataset preparation, Data transformation and feature selection and evaluation metrics. The fifth chapter provides a comprehensive implementation of data mining to find rules and cases from KDD cup99. This includes the supervised and unsupervised approach. The sixth chapter is the main chapter of the thesis where combination of case based reasoning and rule based reasoning took place. Evaluation of the developed system is also discussed under this chapter. The last/seventh chapter presents the conclusions and recommendations of the study.

Chapter Two

Literature Review

In spite of the wide growth of information technology, security has remained one challenging area for computer and networks. The numbers of hacking and intrusion incidents are increasing year on year as technology rolls out [1]. Security threat comes not only from external intruders but also from internal users in the form of misuse. A firewall [2] simply blocks openings into our network/system, but cannot distinguish between good or bad activity. Therefore, if there is a need to allow an opening to a system (like a web-server), then a firewall which is fixed-rule based cannot protect against intrusion. In contrast, Intrusion Detection Systems (IDS) can monitor for hostile activity on these openings [1] [2] [7] [13].

2.1. Network Intrusion Detection (NID)

With advancements in network technologies, Internet services have been undergoing constant growth in network traffic, accompanied by an increasing number of anomalies such as Denial of Service (DoS) attacks, port scans, worms, virus exploits and misconfigurations. These anomalies represent a large fraction of the Internet traffic that is unwanted and prevent legitimate users from accessing network resources in an optimal manner [22]. Therefore, detecting and diagnosing these threats are crucial tasks for network operators to ensure that the Internet resources remain available. Because legitimate traffic must be able to travel efficiently, quickly and accurately identifying anomalies in network traffic is important, requires development of good detection techniques. Anomalies are patterns of interest to network defenders, who want to extract them from vast amount of network traffic data.

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices [17].

2.1.1. Categories of Network Intrusion Detection

There are several ways to categorize the types of IDS as shown in figure 2.1, depending on the kind of activities, transactions and traffics or systems differs. Instruction Detection system [24]

can be categorized into Network based Intrusion Detection System (NIDS) and Host based Intrusion Detection System (HIDS).

Based on the different approaches to event analysis, IDS can also be distinguished between Signature based detection and Anomaly detection.

Each type of IDS has its own merits and demerits [24].

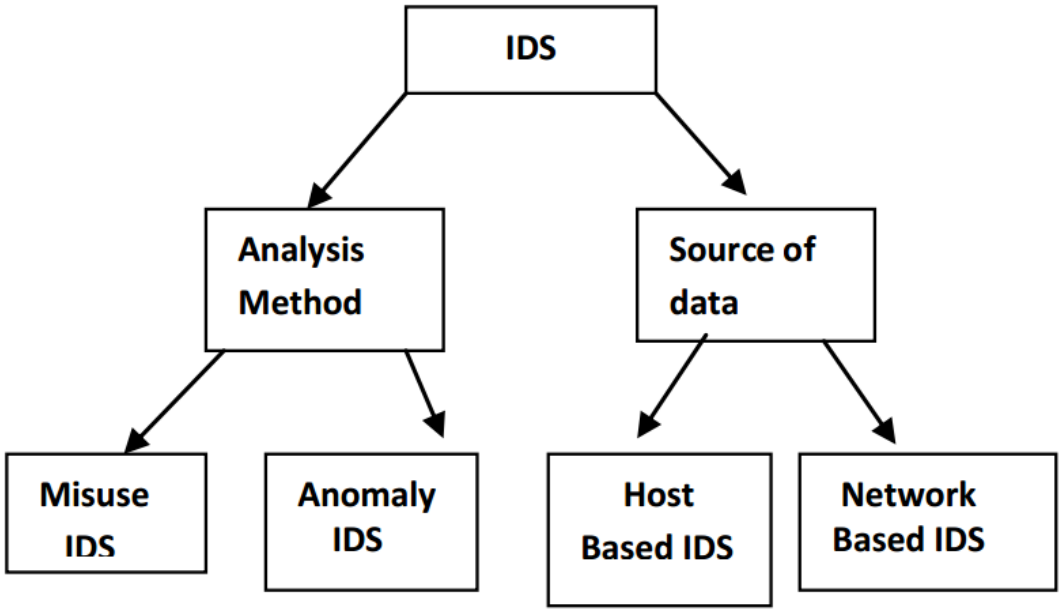

Fig. 2.1 Taxonomy of IDS [24]

Network **based vs Host based intrusion detection systems**

According to [7], [13] and [22] Network based IDS are best suited for alert generation of intrusion from outside the perimeter of the enterprise. NIDS are inserted at various points on Local Area Network (LAN) and observe packets traffic on the network information is assembled into packets and transmitted on LAN or Internet. NIDS are more worth when they are placed outside the firewalls, thereby alerting personals of incoming packets that might get avoided to the firewall. Some NIDS allow taking input of custom signatures taken from user security policy which permits limited detection security policy violation. This limitation is due to packets traffic information that does not work well today in switched and encrypted environments, where packets analysis is weak in detecting, attacking or originating from authorized network users.

NIDS make use of raw network packets as the data source. The IDS typically use a network adapter in licentious mode that listens and analyses all traffic in real-time as it travels across the network.

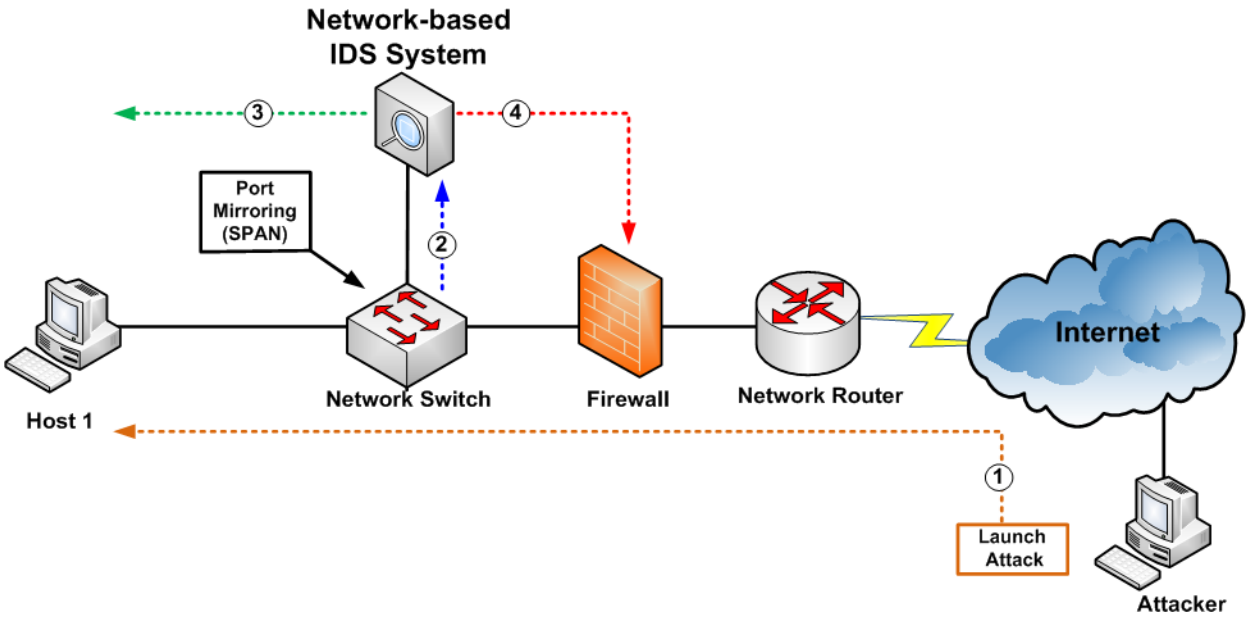

Fig 2.2 Network based Intrusion detection system [23]

In general, a NIDS (as shown in Figure 2.1 [23]) Network Based IDS are to detect malicious activities within the network and packets that are able to pass firewall which means it doesn't crosses the firewall. NIDS are watches the traffic in real time and monitors traffic patterns, packet header, port number etc in order to detect security violation and threat. These types of IDS are valuable for detecting network based attacks like bandwidth based DDoS. Based on the IDS configuration, they react in real time to intervene the attacks before they are been successful. Most common responses are to drop the packets in order to avoid assembling of payload. There are other set of responses like disconnect sessions, reconfigure firewalls, quarantine intruder in honey-pots or padded cells.

According to [24], although there are still many issues in the research of NIDS, the following two issues appear to be the most challenging. The first challenge is the high false alarm rate in the anomaly NIDS, which has formed a hurdle for practical applications; therefore, reducing this high false alarm rate has become an intensive ongoing research topic. The second challenge is the detection of attacks that is likely to generate little network traffic and attacks originating from

inside the protected network. There are some harmful network attacks that do not generate significant network traffic.

Jaiganesh [24] further discussed about the advantages and disadvantages of NIDSs as follows:

Advantages of IDS

- Large networks can be monitored by deploying a few devices with a good network design.
- Ongoing network operations will not be disturbed by deploying NIDS, since they are passive devices.
- NIDSs are not vulnerable to direct attack, and may not be known by attackers.

Disadvantages

- NIDS may fail to recognize attack, when network volume becomes overwhelming.
- Since many switches have limited or no monitoring port capability, some networks are incapable of providing all the data for analysis by a NIDS.
- NIDS cannot analyze encrypted packets, making some of the traffic invisible to the process and reducing the effectiveness of NIDS.
- Attacks involving fragmented or malformed packets cannot be detected easily.

Host based intrusion detection system (HIDS) monitors incoming and outgoing activity on a particular system in the network. Specifically, it monitors the dynamic behavior and the state of the computer system [24]. The administrator will be notified once an intrusion has been detected. An NIDS is usually used alongside a HIDS in order to identify any activities that HIDS overlooked. Host-based IDS places monitoring sensors also known as agents on network resources nodes to monitor audit logs which are generated by network operating system or application program [22]. As per [22] and [24], audit logs contain records for events and activities taking place at individual Network resources. It is done because these HIDS can detect attacks that cannot be seen by NIDS such as Intrusion and can be misused by trusted insider. Host based systems utilize signature rule base which is derived from site-specific security policy. Host based can overcome the problems associated with Network based IDS immediately after alarming the security personnel who can locate the source provided by site security policy. HIDS

also verifies if any attack is unsuccessful, either because of immediate response to alarm or any other reason.

According to [23] HIDS can be represented by the following figure.

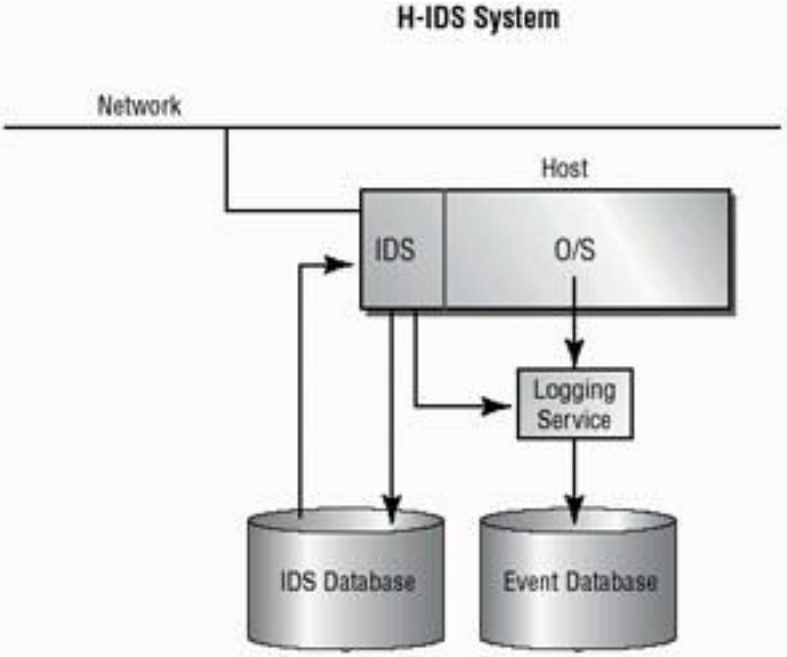

Fig 2.3 Host based Intrusion detection system [23]

Signature Based Detection Vs Anomaly based intrusion detection system

The major category of IDS is known as Misuse Detection which is otherwise known as signature-based detection, since alarms are generated based on specific attack signatures [7] [13] and [25]. This type of signature attacks includes specific traffic or activity that is based on known intrusive activity. In general, Misuse Detection System assumes the abnormal behavior and activity which has a simple to define model. The primary advantage is its simplicity of adding known attacks to the model [24].

IDS can use signature-based detection, relying on known traffic data to analyze potentially unwanted traffic [13]. It has advantages and disadvantages. One of the advantages of Signature-based approaches result in fewer false alarms because they can be very specific about what it is and they are looking for. Because the IDS is looking for something known, a lot of

information regarding what the misuse is, the potential impact, and how to respond can be provided. This knowledge is extremely important in understanding what is occurring and effectively responding [25].

Signature based detection has also disadvantages; Signature-based approaches can only detect misuse for which a signature exists [13 and 24]. For a signature to exist, the form of misuse must be known about beforehand so it can be researched and programmatically identified. This means any new form of misuse will not be detected by a signature based system until it is identified, analyzed, and then incorporated into the product. Depending on the circumstances, this could be hours, days, weeks, or even months [13, 25 and 26].

Anomaly Based Detection on the other hand is based on defining the network behavior. Behavior of the network is the predefined one, when it is accepted or else it triggers the event in the anomaly detection. The accepted behavior of the network is prepared or learned by the specifications of the network administrators [26].

According to [27] the important phase in defining the network behavior is the IDS engine that is capable to cut through the various protocols at all levels. The engine is able to process the protocols and understand the goal. Though this protocol analysis is computationally expensive, the benefits it generates like increasing the rule set helps in less false positive alarms.

According to [13] the major drawback of Anomaly Detection is defining its rule set. The efficiency of the system depends on implementation and testing on all protocols that are available. Rule Defining Process is also affected by various protocols used by several vendors.

Intrusions can be detected by observing deviations from the expected behaviors of the system monitored. These “normal” behaviors can either correspond to some observations made in the past or to some forecasts made by various techniques. Everything that does not correspond to this “normal” pattern will be flagged as anomalous.

Therefore, the core process of anomaly detection is not to learn what is anomalous but to learn what is normal or expected [7] [13] [27]. The key advantage of anomaly-based detection over the other techniques is the ability to detect new attacks, attacks for which no signature or known protocol violation exists [25].

2.2. Components of Intrusion Detection System

As cited by Dawit [13], Intrusion Detection Systems are generally made up of the following main types of components:-

- **Sensors**—these are deployed in a network or on a device to collect data. They take input from various sources, including network packets, log files, and system call traces. Input is collected, organized, and then forwarded to one or more analyzers.
- **Analyzers**—Analyzers in an IDS collect data forwarded by sensors and then determine if an intrusion has actually occurred. Output from the analyzers should include evidence supporting the intrusion report. The analyzers may also provide recommendations and guidance on mitigation steps.
- **User interface**—the user interface of the IDS provides the end user a view and way to interact with the system. Through the interface the user can control and configure the system. Many user interfaces can generate reports as well.
- **Honeypot**—in fully deployed IDS, some administrators may choose to install a "honeypot," essentially a system component set up as bait or decoy for intruders. Honeypots can be used as early warning systems of an attack, decoys from critical systems, and data collection sources for attack analyses. Many IDS vendors maintain honeypots for research purposes, and to develop new intrusion signatures. Note that a honeypot should only be deployed when the organization has the resources to maintain it. A honeypot left unmanaged may become a significant liability because attackers may use a compromised honeypot to attack other systems.

2.3. Types of Attack

An IDS is an effective security tool which helps the users and administrators to prevent unauthorized access to network resources. There are various kinds of attacks that are most common in IDS. They are probe attacks, denial-of-service attacks (DoS), remote to local (R2L) and user to root (U2R) attacks [13] [22].

2.3.1. Probe Attacks

According to Jaiganesh [24] Probe attacks are aimed at acquiring information about the target network from a source that is often external to the network. The basic connection level features

such as the “duration of connection” and “source bytes” are significant while features like “number of files creations” and “number of files accessed” are not expected to provide information for detecting probes.

2.3.2. Denial-of-Service Attacks

The Denial-of-Service attack (DoS) attacks are meant to force the target to stop the service that is provided by flooding it with illegitimate requests [25]. Hence, for the DoS attack to be detected that the traffic features such as the “percentage of connections having same destination host and same service” and packet level features such as the “source bytes” and “percentage of packets with errors” are significant. Detecting the DoS attacks is not being important to know whether a user is “logged in or not.” DoS is an attempt to make a computer resource unavailable to its intended users in computer security. Typically the targets are high-profile web servers and the attack attempts to make the hosted web pages unavailable on the internet. This is a cyber crime, which violates the Internet proper usage policy as indicated by the Internet Architecture Board (IAB) [22] [24].

As stated by Jaiganesh [24], there are two main types of DoS attacks: Flooding and Flaw exploitations. Flooding attacks can be simply implemented. A DoS attack by just using the ping command is a typical example that results in sending the victim more number of ping packets. Moreover, when the attacker has access to greater bandwidth than the victim then this will easily and quickly grab the victim.

DoS attacks [24] have two general forms:

- To force the victim computers to reset or consume its resources such that it can no longer provide its intended service.
- To obstruct the communication media between the intended users and the victim so that they can no longer communicate adequately.

A DoS attack is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service. Examples include [25] [26]:

- Flooding a network, thereby preventing legitimate network traffic.
- Disrupting service to a specific system or person.

- Attacks can be directed at any network device, including attacks on routing devices, web, electronic mail, or Domain Name System Servers.
- Consumption of computational resources, such as bandwidth, disk space, or CPU time.
- Obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.
- Disruption of state information, such as unsolicited resetting of TCP sessions.

2.3.3. Remote to Login (R2L) Attacks

The R2L Attacks are one of the most difficult ones to detect, as they involve the network level and the host level features. Therefore, selecting both the network level features such as the "duration of connection" and "service requested" and the host level features such as the "number of failed login attempts" among others for detecting R2L attacks [13] [22] [24].

2.3.4. User to Root (U2R) Attacks

The U2R Attacks involve the meaningful details that are very difficult to capture at an early stage. Applications of these attacks are Content Based and Target Based. Then for U2R Attacks features such as "number of file creations" and "number of shell prompts invoked," are selected, while features such as "protocol" and "source bytes" are ignored [13] [22] [24].

The different types of attacks and their categories discussed above summarized in table bellow [26]:

Attack type	Category
apacha2, back, land, mailbomb, netpune, pod, processtable, smurf, teardrop, udpstorm	DoS
buffer_overflow, httprunnel, loadmodule, perl, ps, rootkit, sqlattack, xterm	U2R
ftp-write, guess password, imap, multihop, named, phf, sendmail, Snmpgetattack, Snmpguess, spy, warezclient, warezmaster, worm, Xlock, Xsnoop	R2L
ipsweep, Mscan, Namp, portsweep, saint, satan	Probing

Table 2.1 Categories of different types of attacks [26]

2.4. Intrusion Detection Using Data Mining Techniques

Data mining techniques have been popular in extracting the behaviors of these harmful patterns from large volumes of data in recent years. Data mining is used in many application areas, e.g., the business world, medicinal sciences, physical sciences and engineering to make new discoveries [7] [13] [17] [22]. Data mining is used in many application areas, e.g., the business world, medicinal sciences, physical sciences and engineering to make new discoveries. Extensive studies have been performed in applying data mining techniques to network traffic anomaly detection, but the methods [11][13] have limitations that notably discredit them from use in real environments. In this thesis, we explore the possibilities of integrating data mining techniques with knowledge base system in identifying network intrusions with significant performance improvement.

Much number of data mining techniques can be used in intrusion detection, each with its own specific advantage [24].

Data mining has gained a great deal of attention in the information industry and in the society as a whole in recent years due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge. Based on Tigabu [17] and Dokas and Ertoz [15] data mining is defined as follows:

Data mining is the process of automatically discovering useful information in large repositories. Data mining techniques are deployed to scour large databases in order to find novel and useful patterns that might otherwise remain unknown.

The term knowledge discovery in databases (KDD) refers to the process of converting raw data into useful information or knowledge. Data mining is a step in the KDD process, and applies a variety algorithm for extracting patterns from data. In addition to this, as shown at fig. 2.2 bellow, the KDD process has additional steps including data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining to ensure useful knowledge is derived from the data [13].

2.5. Data Mining Tasks

According to Sujatha [22], there are two main tasks of data mining. These are: descriptive modeling and predictive modeling.

The descriptive data mining tasks characterize the general properties of the data in the database, while predictive data mining tasks perform inference of the current data in order to make prediction. Descriptive data mining focus on finding patterns describing the data that can be interpreted by humans, and produces new, nontrivial information based on the available data set. Predictive data mining involves using some variables or fields in the data set to predict unknown or future values of other variables of interest, and produces the model of the system described by the given data set.

The goal of predictive data mining is to produce a model that can be used to perform tasks such as classification, prediction or estimation, while the goal of descriptive data mining is to gain an understanding of the analyzed system by uncovering patterns and relationships in large data sets.

The descriptive task encompasses methods such as clustering, summarization, association rules and sequence analysis.

Descriptive model identifies patterns or relation hips in data. Unlike to the predictive model, it serves as a way to explore the properties of the data examined, not to predict new properties but to cluster objects [**13**]. Clustering is a tool for data analysis, which solves classification problems. Its objective is to distribute cases (people, objects, events etc.) into groups, so that the degree of similarity can be strong between members of the same cluster and weak between members of different clusters [17]. In clustering, there is no pre-classified data and no distinction between independent and dependent variables. Instead, clustering algorithms search for groups of records (the clusters composed of records similar to each other). The algorithms discover these similarities. A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression [13]. Clustering is also called data segmentation in some applications because clustering partitions large datasets into groups according to their similarity. Clustering can also be used for outlier detection, where outliers (values that are “far away” from any cluster) may be more interesting than common cases [17]. In this study descriptive modeling, K-means clustering algorithm is used to generate cases for combined reasoning system.

The goal of a descriptive data mining model is therefore to discover patterns in the data and to understand the relationships between attributes represented by the data, while the goal of a predictive data mining model is to predict the future outcomes based on passed records with known answers.

Siraj [**20**] stated that, a Predictive model makes an estimation about values of data using known results found from different data while the Descriptive model identifies patterns or relationships with the objective of identifying the natural grouping of data. Unlike the predictive model, a descriptive model serves as a way to explore the properties of the data examined, not to predict new properties but to classify them using classification algorithms. In this study, predictive models are used for acquiring rules for combined reasoning system.
Further, Sujatha [22] divides the data mining task of generating models into the following two approaches:

- Supervised or directed data knowledge discovery.
- Unsupervised or undirected knowledge discovery.

The goal in supervised or directed data mining is to use the available data to build a model that describes one particular variable of interest in terms of the rest of the available data. The task is to explain the values of some particular field. The user selects the target field and directs classification algorithms to determine how to estimate, classify or predict its value [13] [22].

In unsupervised or undirected data mining however variable is singled out as the target.

The goals of predictive and descriptive data mining are achieved by using specific data mining techniques that fall within certain primary data mining tasks. The goal is rather to establish some relationship among all the variables in the data. The user asks the system to identify patterns in the data that may be significant. Undirected modeling is used to explain those patterns and relationships once they have been found [26].

Some of the most important data mining techniques used for constructing predictive and descriptive modeling [14] [15] [22] are discussed below.

2.5.1. Classification

Classification is the process of classifying a data instance into one of several predefined categorical classes based on the training set containing known observations. A regression task begins with data instances in which the target values are known. The relationships between predictors and the target are summarized in a regression model that can be applied to different data instances in which the target values are unknown.

In the classification models Decision Trees, Artificial Neural Networks and Navie-Bayes techniques are mainly used.

Decision Trees

Decision trees technique is commonly used in data mining because their construction is cheap, their interpretation is easy, their easy integration with database systems and their good reliability. Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values [17].

Neural networks

As cited by Tigabu [17], artificial neural network models have been studied for many years in the hope of achieving human-like performance in several fields such as speech and image understanding. The networks are composed of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural networks. Computational elements or nodes are connected in several layers (input, hidden and output) via weights that are typically adapted during the training phase to achieve high among the features while the lack of possible arcs in S encodes conditional independencies [17].

Bayesian network

A Bayesian network, Bayes network, belief network, Bayes(ian) model or probabilistic directed acyclic graphical model is a probabilistic graphical model (a type of statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases [6].

Formally, Bayesian networks are DAGs whose nodes represent random variables in the Bayesian sense: they may be observable quantities, latent variables, unknown parameters or hypotheses. Edges represent conditional dependencies; nodes that are not connected (there is no path from one of the variables to the other in the bayesian network) represent variables that are conditionally independent of each other. Each node is associated with a probability function that takes, as input, a particular set of values for the node's parent variables, and gives (as output) the probability (or probability distribution, if applicable) of the variable represented by the node.[8]

2.5.2. Clustering

Clustering is the task of segmenting a diverse group into a number of similar subgroups or clusters. Clusters of objects are formed so that objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. Clustering is commonly used to search for unique groupings within a data set. The distinguishing factor between clustering and classification is that in clustering there are no predefined classes and no examples. The objects are grouped together based on self-similarity. Clustering is grouped under descriptive data mining tasks.

In clustering [17], a set of data items is partitioned into a set of classes such that items with similar characteristics are grouped together. Clustering is best used for finding groups of items that are similar. For example, given a data set of items, subgroups of items that have similar characteristics can be identified [7] and [24]. Clustering is an unsupervised learning process. Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects, so that objects within the same cluster must be similar to some extent, also they should be dissimilar to those objects in other clusters. In classification which record belongs which class is predefined, while in clustering there is no predefined classes. In clustering, objects are grouped together based on their similarities. Similarities between objects are defined by similarity functions, usually similarities are quantitatively specified as distance or other measures by corresponding domain experts [17].

According to [13] the most well-known and commonly used partitioning methods are *k-means* and *k-medoids.*

K-means clustering: is a data mining/machine learning algorithm used to cluster observations into groups of related observations without any prior knowledge of those relationships. The k-

means algorithm is one of the simplest clustering techniques [17]. The k-means algorithm is an evolutionary algorithm that gains its name from its method of operation. The algorithm clusters observations into k groups, where k is provided as an input parameter. It then assigns each observation to clusters based upon the observation's proximity to the mean of the cluster. The cluster's mean is then recomputed and the process begins again. According to [17] discussed bellow is how the k-mean algorithm works:

- The algorithm arbitrarily selects k points as the initial cluster centers ("means").
- Each point in the dataset is assigned to the closed cluster, based upon the Euclidean distance between each point and each cluster center.
- Each cluster center is recomputed as the average of the points in that cluster.
- Steps II and III repeat until the clusters converge. Convergence may be defined differently depending upon the implementation, but it normally means that either no observations change clusters when steps II and III are repeated or that the changes do not make a material difference in the definition of the clusters.

2.6. Knowledge Based System

The concept of knowledge-based systems is derived from the field of artificial intelligence (AI) [27]. AI intends understanding of human intelligence and building of computer programs that are capable of simulating or acting one or more of intelligent behaviors. Intelligent behaviors include cognitive skills like thinking, problem solving, learning, understanding, emotions, consciousness, intuition and creativity, language capacity, etc. These days some of the behaviors such as problem solving, learning and understanding are handled by computer programs [28] [29][30].

Computer systems that try to solve problems in a human expert of the area by using knowledge about the application domain and problem solving techniques are known as Knowledge based system [27] [28].

As stated by Khandelwal [27], the goal of the design of the knowledge-based system is to encapsulate tacit and explicit knowledge in a particular area and code this in a computer in a way that the knowledge of the expert is accessible to novice users.

Azeb [28] noted that, human experts use the knowledge they have about the domain and techniques that lead how to use the knowledge to solve problems. Knowledge based expert systems also designed and developed to handle problems in the same way. They represent the

knowledge about the domain area and they use one or more techniques that guides how to use the knowledge to solve problems.

According to Khandelwal [27] a knowledge base system has the following characteristics.

- It provides the high-quality performance which solves complex problems in a domain as good as or better than human experts.
- This System possesses vast quantities of domain specific knowledge to the minute details.
- Knowledge-based systems reduce the search area for a solution by applying heuristics to guide the reasoning.
- Explanation capability of such systems enables it to review its own reasoning and describe its assessments.
- This system can advice, modify, update, expand and deals with uncertain and irrelevant data.

2.6.1. Building blocks of Knowledge-based system

Every knowledge based system has two building blocks which are known as knowledge base and inference engine [29]. Knowledge base contains all necessary knowledge about the domain that is required to handle problems. The knowledge can be acquired from experts, documents, books and/or other sources. It is formalized and organized with a technique called knowledge representation. There are several ways to represent knowledge in the knowledge base. Two most used knowledge representation techniques are cases and rules.

Case refers to the record of a previous experience or problem [18]. The information recorded about this past experience will, by necessity, depend on the domain of the reasoner and the purpose to which the case will be put. According to Dawit [13], the case base in the CBR system is the memory of all previous stored cases.

On the other hand, rules are conditions represent knowledge in the form of condition/ action pairs. A Rule is a structure which has an ***IF*** component and ***THEN*** component.

IF <condition> THEN <action>

The second component of a knowledge base system is inference engine. After the system gets the required knowledge, it needs to be instructed how to use the knowledge in solving problems. Inference engine represents the reasoning technique that manipulates uses and controls the

knowledge to solve the problems. Case based reasoning and rule based reasoning are the two major reasoning techniques.

2.6.2. Case Based Reasoning Approaches

Case based systems are knowledge based systems that solve problems by remembering similar past situation and reusing its solution and lesson learned from it. Case based systems combine problem solving and learning from new experiences for future use [12]. Cases are used to represent domain knowledge of a case based system. A case refers to specific experience or knowledge tied to specific situation that is worth remembering for future use. So that cases in the knowledge base represent collection of specific experienced, captured and learned situations of the application domain [12][13].

According to Azeb [28], the knowledge base of a case based system represents situations or domain knowledge in the form of cases and the inference engine uses case based reasoning method to solve new problems or to handle new situations.

A case based reasoning technique follows four processes; retrieve, reuse, revise and retain, to accomplish its reasoning task [13]. Figure 2.4[31] shows sequence or cycle of the processes and each process is described below.

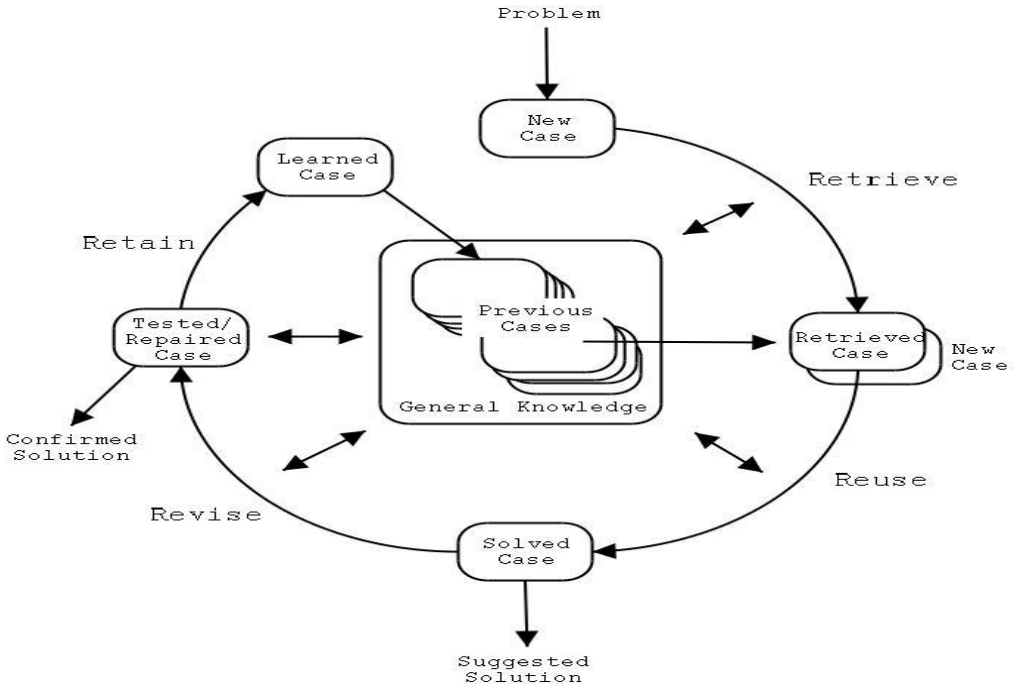

Fig. 2.4 CBR cycle and major processes

- Retrieve phase is an initial step which inquires about previous experiences that are similar to the new case. In this phase most similar cases will be retrieved from the general knowledge.
- Reuse phase is the second step which is responsible in suggesting a solution for the new case from the available solutions of the cases that were retrieved from the general knowledge.
- Revise is the evaluation stage about the selected solution in reuse phase.
- Retain is used to select cases to store tasks give authentication to the user for storing case. It enables to store case(s) into the case base..

2.6.3. Rule Based Reasoning (RBR) approaches

Rule-based reasoning [28] [33] is one of the most popular reasoning paradigms used in artificial intelligence (AI). In a rule-based systems the knowledge base usually consisting of a set of "IF ... THEN ... " rules representing domain knowledge and the inference engine usually containing some domain independent inference mechanisms, such as forward~backward chaining [7]. Main components of RBR are shown in Figure 2.5. Given an input problem, applicable rules are first

found by matching against the rules of the knowledge base; then, intermediate results are generated by the chosen inference mechanism (such as forward or backward chaining), and the process is repeated till the desired solution state is reached. According to [33] the chosen inference mechanism (forward and/or backward chaining) determines whether the antecedents (forward) or the consequents (backward) of the rules in the knowledge base are used for matching and whether the desired solution state is the attainment of a particular conclusion (forward) or the determination of the existence of certain data (backward).

The knowledge base contains the domain knowledge pertinent to the problem, and the solution is, in general, found by incrementally exploring the rule graph formed by the rules in the knowledge base [7].

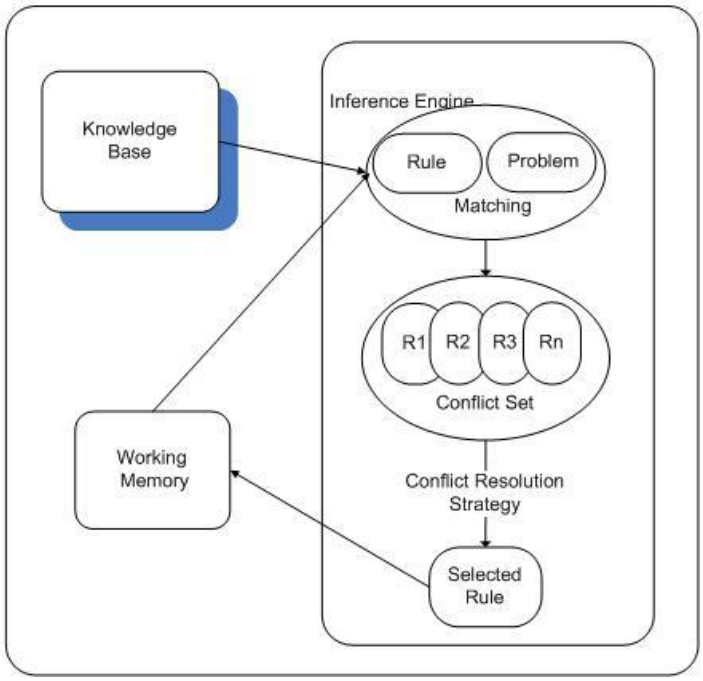

Fig. 2.5 Rule Based Reasoning system components and processes

2.6.4. Designing a Combined CBR and RBR System

2.6.4.1. Why combination is needed?

The advantages and disadvantages of the two reasoning systems and the complementary nature of the reasoning motivate us to combine them.

Rule-based Reasoning

The advantages of a rule-based approach include:

1. The ability to use, in a very direct fashion, experiential knowledge acquired from human experts. This is particularly important in domains that rely heavily on heuristics to manage complexity and/or missing information.
2. Rules map into state space search. Explanation facilities support debugging.
3. The separation of knowledge from control simplifies development of expert systems by enabling an iterative development process where the knowledge engineer acquires, implements, and tests individual rules.
4. Good performance is possible in limited domains. Because of the large amounts of knowledge required for intelligent problem solving, expert systems are limited to narrow domains. However, there are many domains where design of an appropriate system has proven extremely useful.
5. Good explanation facilities. Although the basic rule-based framework supports flexible, problem-specific explanations, it must be mentioned that the ultimate quality of these explanations depends upon the structure and content of the rules.

Disadvantages of rule-based reasoning include:

1. Often the rules obtained from human experts are highly heuristic in nature, and do not capture functional or model-based knowledge of the domain.
2. Heuristic rules tend to be “brittle” and can have difficulty handling missing information or unexpected data values.
3. Another aspect of the brittleness of rules is a tendency to degrade rapidly near the “edges” of the domain knowledge. Unlike humans, rule-based systems are usually unable to fall back on first principles of reasoning when confronted with novel problems.
4. Explanations function at the descriptive level only, omitting theoretical explanations. This follows from the fact that heuristic rules gain much of their power by directly

associating problem symptoms with solutions, without requiring (or supporting) deeper reasoning.

5. The knowledge tends to be very task dependent. Formalized domain knowledge tends to be very specific in its applicability. Currently, knowledge representation languages do not approach the flexibility of human reasoning [12] [38] [39].

Case-based Reasoning

The advantages of case-based reasoning include:

1. The ability to encode historical knowledge directly. In many domains, cases can be obtained from existing case histories, repair logs, or other sources, eliminating the need for intensive knowledge acquisition with a human expert.
2. It allows a system to avoid past errors and exploit past successes. CBR provides a model of learning that is both theoretically interesting and practical enough to apply to complex problems.
3. Extensive analysis of domain knowledge is not required. Unlike a rule-based system, where the knowledge engineer must anticipate rule interactions, CBR allows a simple additive model for knowledge acquisition. This requires an appropriate representation for cases, a useful retrieval index, and a case adaptation strategy.
4. Appropriate indexing strategies add insight and problem-solving power. The ability to distinguish differences in target problems and select an appropriate case is an important source of a case-based reasoner's power; often, indexing algorithms can provide this functionality automatically.

The disadvantages of case-based reasoning include:

1. Cases do not often include deeper knowledge of the domain. This handicaps explanation facilities, and in many situations, it allows the possibility that cases may be misapplied, leading to poor quality or wrong advice.
2. A large case base can suffer problems from store/compute trade-offs.
3. It is difficult to determine good criteria for indexing and matching cases. Currently, retrieval vocabularies and similarity matching algorithms must be carefully hand crafted; this can offset many of the advantages CBR offers for knowledge acquisition [38][40][41].

The ultimate aim of artificial intelligence discipline is to develop systems that exhibit human like, or better intelligence [34]. Most of current knowledge based systems represent some

aspects of human beings intelligence. Azeb [28] on her work discussed about the integration of two or more knowledge based techniques begets a better performance of intelligence than it would have been gained from one technique.

On the other hand Marling and Rissland[35] and Prentzas and Hatzilygeroudis[36] described that, the reasoning power of a knowledge based system depends on the explicit representation and use of different kinds of knowledge about the domain. There is no one way of knowledge representation that can represent the domain knowledge as it is in the reality [34]. The more knowledge based techniques are integrated, the more the domain knowledge is represented, which begets the more efficient system [28].

Case based reasoning and rule based reasoning techniques are two alternative ways of problem solving in intelligent systems. Their knowledge representation and reasoning methods are naturally alternatives [22]. Bellow is presented a comparison of them based on their knowledge representation and problem solving capability.

Cases represent knowledge that is accumulated from specific situations whereas rules represent general knowledge about the domain. Acquiring rules when it is compared to that of cases is really hard. Because of that maintaining or updating rule is harder than updating and maintaining cases.

In the problem solving process [7] [12] [13] [28], case based reasoning uses solutions that was solved in similar past problems whereas rule based reasoning solves problems from scratch though similar problems had been solved previously. Case based reasoning method plays greater role in handling missing or unexpected features in the problem description and selected cases than that of rule based reasoning in problem description and rules. The case base system tries to find the similarity between the problem and the cases though there are features that do not match between them. However, the rule based system tries to find rules that perfectly match with part or all of the problem description. Rule based reasoning method is better in providing explanation for the given solution than case based reasoning [7] [12] [13] [35].

Due to their interchangeable nature, combining them provides effective knowledge representation, problem solving power, and exceeding one’s weakness with the other in different areas of applications.

2.7. Standard Metrics for Evaluation

To evaluate the approach used for network intrusion detection [13], four standard metrics of true positive, true negative, false positive and false negative have been proposed. Table 2.2 shows a typically used to evaluate performance of a machine learning algorithm.

Confusion matrix		Predicted connection label	
		Intrusions (Attacks)	Normal (non intrusions)
Actual connection label	Intrusions (Attacks)	Correctly detected Attacks, True Positive (TP)	False Alarm, False Posistive (FP)
	Normal	False Negative (FN)	True Negative (TN)

Table 2.2: Standard metrics for evaluations of intrusions (attacks)

- True Positive (TP): represents the number of malicious records or attacks that are correctly identified by network intrusion detection system.
- False Positive (FP): The number of records that are incorrectly identified as attacks however in fact they are legitimate activities.
- True Negative (TN): The number of legitimate records that are correctly classified as normal or non attacks.
- False Negative (FN): The number of records that are incorrectly classified as legitimate activities however in fact they are malicious.

These metrics are used to evaluate and know the recall and precision level of the system.

Precision is the number of class members classified correctly over the total number of instances classified as class members. Technically, precision can be expressed as the extent to which the IDS detect correctly an attack, which is, actually occurred (13).

$$\boldsymbol{Presision(P) = \frac{TP}{TP+FP}} \quad \text{..... Equation 1}$$

Recall measures the number of correctly classified examples relative to the total number of positive examples. In other words the recall indicates number of class members classified correctly over the total number of class members [13].

$$\boldsymbol{Recall(R)} = \frac{\mathbf{TP}}{\mathbf{TP + FN}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation 2}$$

By manipulating further for precision and recall, it is possible to find harmonic mean of the two, which is F-Measures that measures the effectiveness of retrieval of the system.

$$\mathbf{F - Measures} = \frac{\mathbf{2 \times PR}}{\mathbf{P + R}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation 3}$$

2.8. Related works

The researcher review different research works which contains the following concepts. Network intrusion detection, data mining for network intrusion, Knowledge base system, and case based and rule based reasoning, integration of data mining with knowledge base system, integration of case based and rule based reasoning.

Tigabu [17] designed a semi-supervised intrusion detection system and he recommended designing knowledge base system which will add adaptability and extensibility features for intrusion detection. The experiments were conducted following the Knowledge Discovery in Database process model. The Knowledge Discovery in Database process model starts from selection of the datasets. The dataset used in this study has been taken from Massachusetts Institute of Technology Lincoln laboratory. After taking the data, it has been preprocessed. The major preprocessing activities include fill in missed values, remove outliers; resolve inconsistencies, integration of data that contains both labeled and unlabeled datasets, dimensionality reduction, size reduction and data transformation activity like discretization tasks were done for this study. The system has a prediction accuracy of 96.11% on the training datasets and 93.2% on the test dataset to classify the new instances as normal, DOS, U2R, R2L and probe classes. . Data mining has been used for intrusion detection systems due to the fact that they are generally more precise and require far less manual processing and input from human experts. But researches which employed data mining for intrusion detection merely generate patterns and they lack in utilizing the knowledge.

Azeb[28] attempted a research project for her master's thesis on how a rule based reasoning method can be integrated with the existing case based system called DrillEdge in order to improve the system's performance in handling complex situations in simpler and accurate ways. The research included studying how they can be integrated, and designing, implementing and testing of a demo system that demonstrates the integration. But Azeb [28] did not include the integration of data mining with the integrated reasoning approaches on her system.

Abdulkerim [7] attempted to integrate data mining with knowledge based system to design a rule-based intrusion detection system. His system is aiming at utilizing hidden knowledge extracted by employing induction algorithm of data mining, specifically JRip from sampled KDDcup'99 intrusion data set. The integrator application then links the model created by JRip classifier to knowledge based system so as to add knowledge automatically. In doing so, the integrator understands the syntax of JRip classifier and PROLOG and converts from rule representation in JRip to PROLOG understandable format. These system has scored 80.5% overall performance and the researcher recommends further investigations on different algorithms of the same case and the combination/integration of different reasoning systems to refine the knowledge base and boost the advantages of integrating data mining induced knowledge with knowledge based system.

Dawit [13] conduct a research on an integration of data mining with case based reasoning system for network intrusion detection. The system is aiming at utilizing hidden knowledge extracted by employing descriptive algorithm of data mining, specifically K-means cluster from sampled KDDcup'99 intrusion dataset. Clustered case with centroid value is mapped to COLIBRI Studio IDE and then the integrator application creates using Eclipse IDE with JDK 6. The important part of a case based reasoning model includes case retrieval; the similarity measuring stage, reuse; which allows domain expert to transfer retrieval case solution to suit for the current case, revise; to test the solution and retain to store the confirmed solution to the case base for future use. The system achieves 89.5% accuracy and further studies suggested to be done to improve the retrieval process by applying ontology based retrieval and integration of case based reasoning with rule based reasoning.

Jaiganesh [24] conducted a Ph.D dissertation on the Investigation of machine learning algorithms for network intrusion detection system, in this work; a data mining based hybrid intrusion detection system has been designed for distinguishing normal and intrusive events. The major

contributions of this research work were the proposal of a hybrid architectural framework for effective intrusion detection, the detection techniques for network and/or host intrusion detection systems that use classification and clustering algorithms to enhance the performance of the intrusion detection system. On this work, the integration was not between CBR and RBR, the detection techniques for network and/or host intrusion detection systems that use an integration of classification and clustering algorithms to enhance the performance of the intrusion detection system.

Therefore, this study focuses exploring a way to automatically use a knowledge acquired using descriptive and predictive mining models for network intrusion detection systems that combines case-based reasoning and rule-based reasoning.

CHAPTER THREE

METHODS AND APPROCHES

In this study, a combined reasoning system for knowledge based network intrusion detection is designed. This is done by combining rule-based system (which is created from predictive data mining models) and case-based system (which is created from descriptive data mining models).

3.1. Architecture of the proposed system

As a hypothesis to be tested, this research will provides a combined reasoning system for knowledge based network intrusion detection that will perform better than what has been done before. So, the system will help network administrators at different sectors to take action in accordance of the detected network attack, and the study will be able to enhance intelligence of the knowledge based intrusion detection system and provide idea on the integral of data mining with combined system.. Based on the hypothesis the following system architecture is designed for experimentation.

Figure 3.1 presents an intrusion detection system proposed in this study. The system takes intrusion data set to construct predictive model using classification algorithm. Hidden knowledge available in predictive models are rules, that are used for designing rule-based reasoning system.

In addition, descriptive models is constructed with the help of clustering algorithms. From descriptive models, cases are generated for developing case-based reasoning system. Finally, these two systems are combined together for the development of better knowledge based network intrusion detection system.

As cases or problems happens the combined reasoning system first use RBR to search for the rules/facts that matches with the new query from the rule library. If exact matches found the RBR gives the solution. If not, the CBR will receive the query.

The CBR system has the capability of retrieve, reuse, revise and retain cases. The retrieval task starts with a new case description and ends when best matching similarity case of attacks or normal cases have been found in the case base. As users enter case description (query) through interface, the system searches the best matching cases from the case base and return possible solution in ranked order to the problems described in the query. If exact matching occurred in

between the query and cases in the case base, users can directly derive the solution or if the similarity or matching is partial, then adaptation, revision and reuse tasks are done to make the proposed solution best for the problem at hand. At last, those best modified solution to the problem stored to the case base for future use.

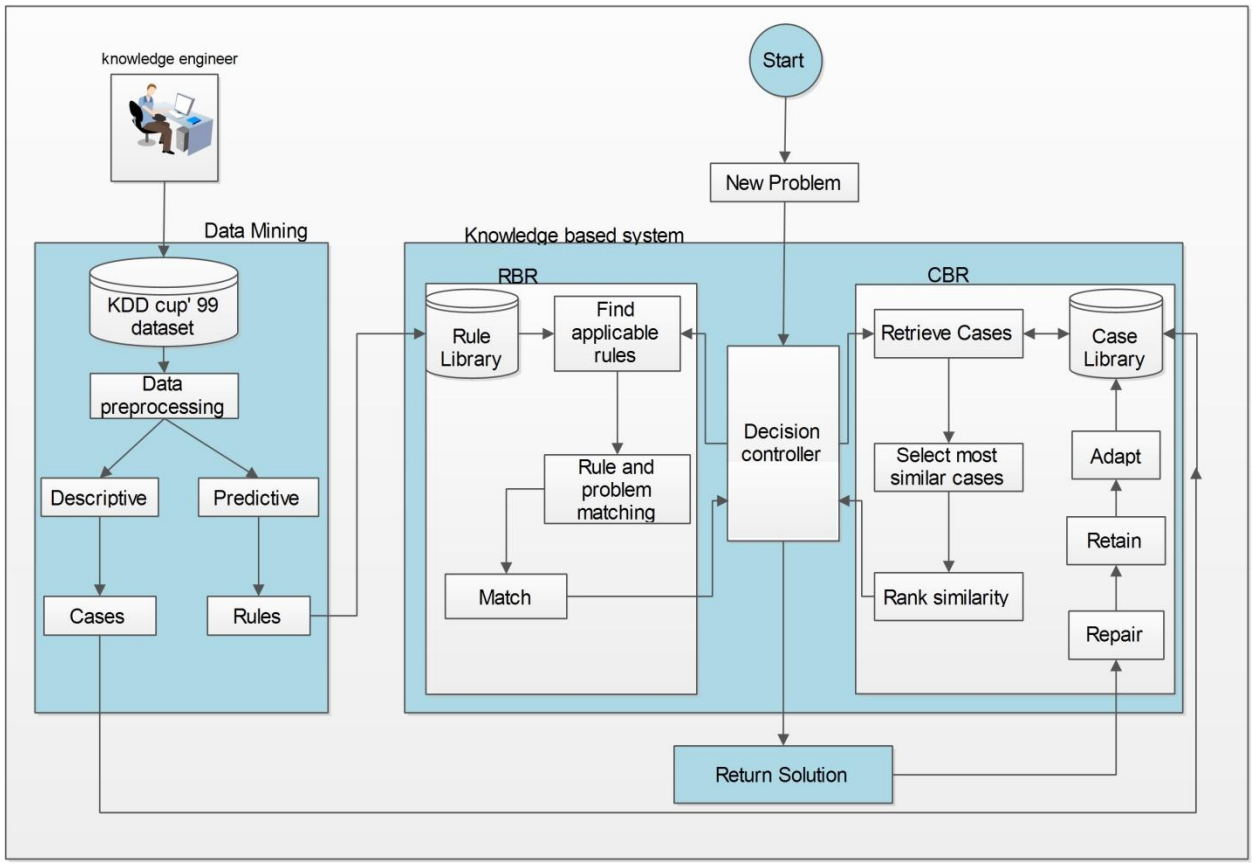

Fig. 3.1 Proposed System Architecture

3.2. Data Mining

Data Preprocessing subtask is applied for effective data selection to enhance the performance of the intrusion detection system. It contains the feature extraction component to extract the required features from KDD CUP 1999 Dataset for the process of intrusion detection. The dataset has been formed out of the selected features and the reduced instances of the KDD CUP 1999 Dataset for classification.

As cited by Abdulkerim[7], the KDDcup‘99 dataset, has inherent problems in which the most important one is the existence of redundant instances . From the collected 1,048,575 instances of KDD dataset for this study, 58.5% of them are found redundant. Therefore, before the actual

mining task is performed, these instances are removed at the data preprocessing stage.

During the preprocessing stage of instances are classified as normal and attack, and also attacks are further classified into four sections represented into DOS (deny of service), Probe (information gathering), U2R (user to root) and U2L (remote to local).

In this study, both predictive and descriptive mining models are considered.

The descriptive model identifies the patterns or relationships in data and explores the properties of the data examined by using clustering algorithms.

In this research as shown in the figure, a descriptive data mining model is applied and the best algorithms is selected to generate cases that are used to build the case library for developing knowledge based intrusion detection system.

Predictive Models is also used for this study. This model makes prediction about unknown data values from the known values using classification algorithm. This model of the data mining enables to generate rules for the purpose of generated for building the rule library in the knowledge base system to simplify intrusion detection.

3.3. Knowledge based system

This component of the system consists of two subsystems: Case based reasoner and Rule based reasoner.

Case-Based Reasoning is the process of solving new problems based on the solutions of similar past problems. An auto mechanic who fixes an engine by recalling another car that exhibited similar symptoms is using case-based reasoning [28]. Important issues in CBR of the above architecture are:

- **Representation:** This refers to the representational scheme used for the cases in the case library. There are several dimensions along which this issue can be analyzed. The data or memory structures for cases are frames or some variant of a frame system [39]. As mentioned above, the case library contain cases comes from the data mining component of the system and learn from the new cases. So when new cases or problems come to the system, search through the case library to retrieve the most relevant case.
- **Retrieval and Similarity Metrics:** While determining which case to retrieve from the case library. The local similarity is used to measure the similarity of attributes of the new case with their corresponding attributes value of old case from the case base. Local

similarity functions are used to compare simple attribute values. In this research, the following local similarity functions are used [13].

- ✓ **Equal:** equal local similarity is selected for each attribute, then there is a need for input and value of case base to match. If value matches exactly then it will get result otherwise match failure.
- ✓ **Interval:** When you select similarity interval and adjust interval value, then, the value is matched as per the given interval. Exact value match is not compulsory in this similarity measure.

Global Similarity is linked with compound attributes and used to get similarity of collected attributes in unique similarity value. Global similarity used in this research is average similarity.

- ✓ **Average:** is a type of global similarity that considers the average of all attribute local similarity values.

- ➢ **Adaptation**: - It is a technique to alter retrieved case for reproducing new solution for new problem. jCOLIBRI offers two basic adaptation methods:- the first is Direct Attribute Copy Method. This method copies the value of an attribute in the query to an attribute of a case. The second is Numeric Direct Proportion Method which performs a numerical direct proportion among attributes of the query and the case. For this study Numeric Direct Proportion Method are employed for case adaption.

Rule-Based Reasoning is the other component of the knowledge base of this system. Issues of concern in RBR are:

- ➢ **Rule Library:** is a container of rules about network attacks or signatures which are generated by JRip rule induction algorithm after mapped by the integrator to PROLOG understandable format.
- ➢ **Nature of Facts and Rules:** In conventional RBR, facts and (IF... THEN...) Rules are strictly *categorical* in nature (e.g., IF *it is cloudy,* THEN *it shall rain)* [37]. Both commonsense knowledge and domain expertise are more naturally expressed in terms of plausible rules (e.g., IF *it is cloudy,* THEN *it is POSSIBLE that it may rain).* Conventional IF...THEN... rules also require a strict Boolean match on the *premises* and the *conclusions;* however, this is very restrictive as real-world situations are often fuzzy and do not match exactly with rule premises and conclusions. For example, the premise it is *cloudy* is fuzzy and many a time the sky can be classified as *partially cloudy.* Thus, it is necessary to be able to accommodate partial degrees of matching in rule premises

and conclusions [38].

- **Inference in Rule Graph:** Given the above stated need to incorporate uncertainty in the structure of rules, some mechanisms have to be used to propagate the partial degrees of matching through the rule graph to determine the degrees of confirmation and refutation of the various conclusions. These mechanisms include means to aggregate the uncertainty of the premises, propagate this uncertainty to the conclusions and aggregate the uncertainty of the conclusion. The last task is important because a particular conclusion may be reached via multiple proof paths, each path contributing to the confirmation or refutation of the conclusion. Different uncertainty calculi are described in the literature to deal with some or all of these aspects.
- **Structure of Knowledge Base:** Large knowledge bases can contain many (hundreds or thousands of) rules, and under such conditions, it is necessary to structure the knowledge base appropriately for the purposes of efficiency in inference and ease of debugging and knowledge engineering [38]. *Partitions, abstraction hierarchies,* and *contexts* are some of the techniques used in the literature for structuring the knowledge base. *Belief revision* mechanisms are also required for maintaining the validity and consistency of the knowledge base [37].

User Interface: - is the interaction point between the user and the system. The user interface can be graphical user interface (GUI) or command line interface (CLI). In the course of integration of data mining with knowledge based system, both Graphical User Interface for the integrator and Command Line Interface for the knowledge based system are used.

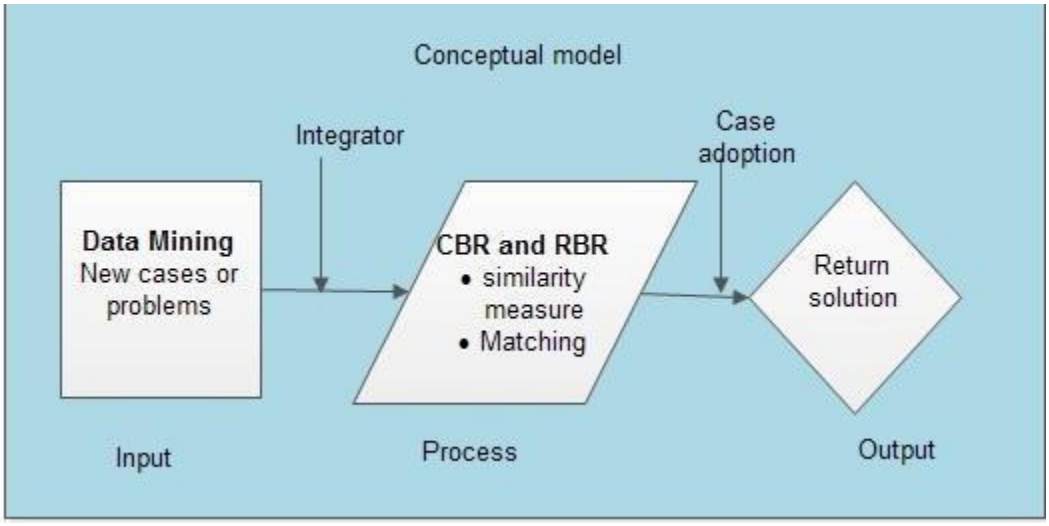

Figure 3.2: conceptual design of the proposed system

3.4. Combination of Case Based Reasoning with Rule based reasoning

Much of the current research in CBR considers it in isolation from other reasoning techniques; however, in general, CBR is just another proof (solution) path to a goal (problem). It is important to consider the interactions of CBR with other reasoning methodologies that might also be employed in a problem solver [12] [36].

RBR and CBR are largely *complementary* reasoning methodologies. RBR can better represent specialized domain knowledge in a modular, declarative fashion, whereas CBR can better represent past experiences and domain complexity [12]. Significant benefits are possible by combining RBR and CBR. For example, CBR can directly enhance RBR by providing a context for screening the knowledge base and extending the coverage of rules by representing exceptions (to the rule) in the form of cases. Going the other direction, RBR can enhance CBR by expressing domain knowledge to dynamically determine the contextually dependent relevance of a feature set (or attributes of a case) to a given goal and dynamically select the best similarity/relevancy measure to use for case retrieval [37].

This research is mainly focuses on the integration of CBR and RBR. Approaches integrating rules and cases based on component dominance are categorized in to rule dominant, case dominant and balanced approach [12]. According to [12] in rule-dominant approaches, the rule-based component prevails in the inference process, whereas the case-based component plays a complementary role. These approaches usually focus on the rule-based component and invoke the case-based component only when necessary (usually in specialized or exceptional situations). But this is not advisable when there are a large number of rules in the rule library. In balanced approaches, the role of the integrated components is balanced that is, none of the integrated components plays a supportive role. In case-dominant approaches, the case-based component plays a more important role and the rule-based component is less significant (supportive). This scheme is useful, because, when the case library contains a limited number of cases and the adaption or learning ability of CBR from previous cases or problems.

CHAPTER FOUR

DATA PREPARATION

4.1. Data Source

Normally, network traffic log data is not released by many organizations due to privacy concerns. Therefore, most IDSs are focused more on getting relevant data first since such systems lack good quality data. Therefore, the dataset for this study is collected from KDDcup'99 dataset available at ACM Knowledge Discovery [13]. The main reason for selecting KDD Cup 99 dataset is that currently, it is the most widely used comprehensive data set that is shared by many researchers. In 1999, the original TCP dump files were preprocessed for utilization in the Intrusion Detection System benchmark of the International Knowledge Discovery and Data Mining Tools Competition. To do so, packet information in the TCP dump file is summarized into connections. Specifically, "a connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows from a source IP address to a target IP address under some well defined protocol". This process is completed using the Bro IDS, resulting in 41 features for each connection; these features are grouped into four categories such as: basic features, content features, traffic features.

a. Basic Features: Basic features comprises of all the attributes that are extracted from a TCP/IP connection. These features are extracted from the packet header and includes src_bytes, dst_bytes, protocol etc

b. Content Features: These features are used to evaluate the payload of the original TCP packet and looks for suspicious behavior in the payload portion. This includes features such as the number of failed login attempts, number of file creation operations etc. Moreover, most of the R2L and U2R attacks don't have any frequent sequential patterns. This is due to the fact that DoS and Probing attacks involve many connections to some host(s) in a very short duration of time but the R2L and U2R attacks are embedded in the data portions of the packets, and generally involves only a single connection. So to detect these kinds of attacks, content based features are used.

c. Traffic Features: These include features that are computed with respect to a window

interval and are divided into two categories: “Same host” features and “Same service” features.

“Same host” features: are derived only by examining the connections in the past 2 seconds that have the same destination host as the current connection, and compute statistics related to protocol behavior, service etc.

“Same service” features: These features examine only the connections in the past 2 seconds that have the same service as the current connection. The above two types are called “time based traffic features”.

The main reason for selecting KDD Cup 99 dataset is that currently, it is the mostly used comprehensive data set that is shared by many researchers. As presented in table 4.1, the KDD cup’99 dataset has 41 attributes are used in each record to characterize network traffic behavour. Features present in KDD data set are grouped into three categories and are discussed below.

S.No	Feature Name	Description	Data Type
1.	Duration	length (number of seconds) of the connection	Integer
2.	Protocol_type	type of the protocol, e.g. tcp, udp, etc.	String
3.	Service	network service on the destination e.g.http, telnet, etc.	String
4.	Src_bytes	number of data bytes from source to destination	String
5.	Dst_bytes	number of data bytes from destinationto source	Integer
6.	Flag	normal or error status of theConnection	Integer
7.	Land	1 if connection is from/to the samehost/port; 0 otherwise	Integer
8.	Wrong_fragment	number of ``wrong'' fragments	Integer
9.	Urgent	number of urgent packets	Integer
10.	Hot	number of ``hot'' indicators	Integer
11.	Num_failed_logins	number of failed login attempts	Integer
12.	Logged_in	1 if successfully logged in; 0 otherwise	Integer
13.	Num_compromised	number of ``compromised'' conditions	Integer
14.	Root_shell	1 if root shell is obtained; 0 otherwise	Integer
15.	Su_attempted	1 if ``su root'' command attempted; 0 Otherwise	Integer
16.	Num_root	number of ``root'' accesses	Integer
17.	Num_file_creations	number of file creation operations	Integer
18.	Num_shells	number of shell prompts	Integer

19.	Num_access_files	number of operations on access control files	Integer
20.	Num_outbound_cmdS	number of outbound commands in an ftp session	Integer
21.	Is_host_login	1 if the login belongs to the ``host'' list; 0 otherwise	Integer
22.	Is_guest_login	1 if the login is a ``guest''login; 0 otherwise	Integer
23.	Count	number of connections to the same host as the current connection in the past two seconds	Integer
24.	Serror_rate	% of connections that have ``SYN'' Errors	Integer
25.	Rerror_rate	% of connections that have ``REJ'' Errors	Integer
26	Same_srv_rate	% of connections to the same service	Integer
27	Diff_srv_rate	% of connections to different services	Integer
28	Srv_count	number of connections to the same service as the current connection in the past two seconds	Integer
29	Srv_serror_rate	% of connections that have ``SYN'' Errors	Double
30	Srv_rerror_rate	% of connections that have ``REJ'' Errors	Double
31	Srv_diff_host_rate	% of connections to different hosts	Double
32	Dst_host_count	count of connections having the same destination host	Integer
33	Dst_host_srv_count	count of connections having the same destination host and using the same Service	Integer
34	Dst_host_same_srv_rate	% of connections having the same destination host and using the same Service	Double
35	Dst_host_diff_srv_rate	% of different services on the current Host	Double
36	Dst_host_same_src_rate	% of connections to the current host	Double
37	Dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts	Double
38	Dst_host_serror_rate	% of connections to the current host that have an S0 error	Double
39	Dst_host_srv_serror_rate	% of connections to the current host and specified service that have an S0error	Double
40	Dst_host_rerror_rate	% of connections to the current host that have an RST error	Double
41	Dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST Error	Double

Table 4.1 List of Features Available in KDD Cup 99 Dataset

4.2. Data Preprocessing

The need for data preprocessing can be seen from the fact that redundant data and insignificant features may often confuse the classification algorithm, leading to the discovery of inaccurate or ineffective knowledge. Moreover, the processing time will increase when all features are used. Finally, preprocessing helps to remove the redundant data, incomplete data and transforms the data into a uniform format [22].

The preprocessing module of the proposed system performs the following functionalities [17] [36]:

i. Performs redundancy check and handles null values
ii. Converts numerical data to categorical data

4.2.1. Redundancy Check

The major limitation with KDD Cup 99 dataset is the presence of redundant records. The occurrence of redundant instances causes the learning algorithm to be biased towards frequent records and unbiased towards infrequent records. After redundancy check, it has been found that most of the redundant records are present in the anomaly class than in the normal class. The detection accuracy has been increased when these redundant records have been removed. For instance, the learning algorithm is unbiased towards R2L class, as the percentage of records in R2L class is very less in the KDD Cup 99 dataset and due to the redundant and enormous records present in other classes like DoS [36].

4.2.2 Sampling data set

The KDDcup'99 data set is very huge in size; even after removal of redundant instances the remaining dataset is so huge which requires time and memory space during the mining process. Hence, the sampling is found to be paramount before using data mining to extract patterns from dataset and considerable samples are taken. While sampling, all instances of R2L and U2R are taken since their size is so small compared to others. The number of instances included in the sample for normal, Probe and DOS is based on their proportion on the cleaned dataset.

4.2.3 Data Selection

Today's real-world data are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size (often several gigabytes or more) and their likely origin from multiple,

heterogeneous sources. Low-quality data will lead to low-quality mining results [7]. One of the most important deficiencies in the KDD data set is the huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records, and thus prevent them from learning unfrequented records which are usually more harmful to networks such as U2R and R2L attacks. In addition, the existence of these repeated records in the test set will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records. Therefore, before the actual mining task is performed, repeated instances are removed at the data preprocessing stage.

The data set was arranged for data mining technique and also suitable for the selected CBR and RBR tools. As a result, the redundant instances in the record were removed during the preparation of dataset for data mining and case features for building the CBR case-base. The dataset contains four different categories of intrusions such as DOS, R2L, U2R and Probing and alsoNormal. For this study, 4,898,401 instances were collected from the website and among them 494,427 instances were found non-redundant. Table 4.2 shows the distribution of normal and attacks instances before Vs after removing redundant instance.

	Total instances		
Attacks	**Before removing redundant**	**After removing**	**Percentage**
R2L	1,126	1,126	0.22
Probe	41,102	4,107	0.83
U2R	49	49	0.11
DOS	3,883,370	391,458	79.17
Normal	972,780	97,277	19.67
Total	4,898,427	494,427	

Table 4.2: Distribution of different types of attacks and normal instances

After preprocessing the dataset is converted into comma delimited (CSV) format, then to arff format suitable for mining using WEKA 3.7.9.

4.3. Experimentation

After data preprocessing and preparation task completed and converted into suitable format for WEKA, the next task is the experimentation involving the selected algorithms. The experiment has two categories, creating predictive model and descriptive model. The sampled data set contains 54,111 instances containing normal, probe, DOS, U2R and R2L. The data set contains 41 attributes and all of them are involved in all experiments.

4.3.1. Creating Descriptive model

As this stage, to build a descriptive model and get cases for knowledge representation, three descriptive models involving K-means, Expectation maximization (EM) and Make Density Based clustering algorithm are constructed.

The first experiment is undertaken using k-means clustering involves its default value of parameters. The principle goal of employing the K-means clustering scheme is to separate the collection of normal and attack data that behave similarly into several partitions which is known as K-th cluster centroids. In other words, K-means estimates a fixed number of K, the best cluster centroid representing data with similar behavior. In this study, the number of clusters has 5 based on the number of class, so predefined K=5, representing Cluster 1, Cluster 2, Cluster 3, Cluster 4, and Cluster 5. Hence, the distance has calculated using Euclidean distance. The algorithm correctly clustered is 48,396 instances and 5,715 instances have incorrectly clustered. 13.2 seconds is used to build the model.

The second experiment is based on Make Density Based Clustere algorithm. Hence, the distance is calculated using Euclidean distance and the number of clusters has 5 based on the number of class. The correctly clustered instance 86.9% and from the total number of 54,111 instances 47,419 instances are correctly clustered and 6,701instances are not correctly clustered. 14.7 seconds is used to build the model.

The third experiment is based on Expectation Maximization (EM) algorithm. Hence, the distance has calculated using Euclidean distance and the number of clusters has 5 based on the number of class. The correctly clustered instance 80.1% and from the total number of 54,111 instances 45,300 instances are correctly clustered and 8,811 instances are not correctly clustered.

As it is clearly depicted in table 4.3, the three-cluster algorithm performed different result. Result of K-means algorithms has shown the better-clustered accuracy than other. The highest incorrectly clustered is shown by Expectation Maximization (EM) algorithm. 456.52 seconds is used to build the model.

Cluster	**Correctly clustered**		**Incorrectly clustered**		**Time taken to build model**
	No.	Percentage	No.	Percentage	
K-means	48,396	89.43%	5,715	10.57%	13.2seconds
Make Density Based Clusterer	47,419	86.9%	6,694	13.1%	14.7 seconds
EM	45,300	80.1%	8,811	19.9%	456.52seconds

Table 4.3 Performance of cluster

As shown in the table and justified by Dawit [13] on his experiment, to create descriptive models using data mining, the K-means has best performance among the three clustering algorithm and it correctly clustered 89.43% instances and the time used to build the model is 13.2 seconds. So, the k-means clustering algorithm is used to build cases for the system.

The k-mean cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity. The k-means algorithm takes the input parameter, *k*, and partitions a set of n objects into *k* clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. In this study, a combined reasoning system for knowledge-based network intrusion detection are proposed and in order to develop the knowledge base cases are store into knowledge base then retrieve based on the similarity of the query. The k-means algorithm are undertaken to perform the clustered the dataset based on the centroid value. The case retrieval process retrieves case based on their similarity.

To test k-means clustering algorithm, test dataset is prepared. The two weeks of test dataset yielded around of two million connections record and the researcher prepared 15,068 dataset by using datapreparator data preprocessing tool.

Table 4.4 shows the evaluation result of test set.

=== Evaluation on test set ===

Clustered Instances

No.	Label	Count	%
1	Normal	5,833	37.7%
2	R2L	1,657	11.9%
3	Probe	3,227	20.4%
4	DOS	4,351	29%
5	U2R	699	1%

Table 4.4 Evaluation of test set

4.3.2. Creating Predictive model

According to Abdukerim [7], to create predictive model and generate rules using data mining he used four predictive models involving J48, REPTree, PART and JRip classifier algorithms are constructed. J48 and REPTree are tree based classifiers in WEKA whereas PART and JRip are rule based classifiers.

The rule acquired from the classifier algorithms is used for constructing knowledge base. So as to develop effective knowledge base system, acquiring relevant rules is paramount. Hence, from the four algorithms the researcher selected the classifier which best performed on classifying the data set.

Classifier	**Correctly classified instances**		**Incorrectly classified instances**		**Time taken to build model**
	No.	Percentage	No.	Percentage	
PART	54,029	99.848%	82	0.1202%	20.1
JRip	54,078	**99.939%**	**33**	**0.061%**	**39.31**
REPTree	53,929	99.654%	182	0.346%	7.1
J48	54,041	99.870%	70	0.129%	12.40

Table 4.5 Performance of classifiers

The first experiment is undertaken using PART classifier. The correctly classified instance 99.848% and from the total number of 54,111 instances 54,029 instances are correctly classified and 82 instances (0.120%) are not correctly classified.

The second experiment for generating rules is undertaken using JRip classifier. The correctly classified instance 99.939% and from the total number of 54,111 instances 54,078 instances are correctly classified and 33(0.061%) instances are not correctly classified.

The third experiment for generating rules is undertaken using REPTree classifier. The correctly classified instance 99.654% and from the total number of 54,111 instances 53,929 instances are correctly classified and 182(0.346%) instances are not correctly classified.

The last experiment is undertaken using J48 classifier. The correctly classified instance 99.870% and from the total number of 54,111 instances 54,029 instances are correctly classified and 70 instances (0.129%) are not correctly classified.

As shown in Table 4.5 from the experimental results, to create predictive model, the justified experimental algorithm is applied. That is JRip classifier algorithms.

JRip has best performance among the four classifiers. Its prediction accuracy and TP rate for all types of attacks are above 99% which is very good performance in predicting attacks and normal incidents correctly. The FP rate is almost negligible for normal, Probe, DOS and R2L classes. This shows the model developed using JRip is acceptable for constructing the rule base of the knowledge base system.

JRip classifier has generated 23 rules. The rules involved 20 features/attributes among the 42 features/attributes from the sample data set. The algorithm generated 22 rules for the attacks namely, probe, DOS, R2L and U2R attacks and only one rule for normal behavior. It can be deduced from the rules that if a certain incident satisfies one of the 22 rules it is an attack otherwise it is a normal network incident. Among the attack classes, all but U2R has more than one rule. This is related to the number of instances occurred in the sample data set for U2R is the smallest as compared to the others. For this study, the rules generated by JRip algorithm is used to build the rule-based library of the combined reasoning system.

CHAPTER FIVE

USE OF THE DISCOVERED KNOWLEDGE USING DATA MINING

The objective of this study is to use the discovered hidden knowledge using predictive and descriptive data mining techniques to design a combined reasoning system for knowledge based intrusion detection . Predictive model is used for constructing rule-based system and descriptive model is also employed for designing case-based system. These systems are combined systematically to enhance the performance of knowledge-based system for network intrusion detection.

5.1. Combination of Data Mining with Case Based Reasoning

As stated by Dawit [13], the case representation is made in a way that easily fit to COLIBRI Studio. Designing of such a case structure helps easily define the features available in the case and to measure the similarity between existing and new cases. Hence, one of the applications of this research is to retrieve similar cases from the case base that can guide future reasoning of network intrusion detection. The collections of cases were structured to make efficient in retrieval process. This is done through case indexing process [13] [14].

Defining case structure in COLIBRI Studio are done by using simple manage case structure window. It is very easy to define case structure with COLIBRI Studio. Because it is simple to add attributes in description of case structure and set properties of attributes of metadata of attributes. Metadata of attributes are weight of attributes, data type of attributes and similarity function. During configuration of case structures, COLIBRI Studio creates codes automatically and saved in xml file format. Most significant attributes are set by declaring higher weight as compared to other weights.

Descriptive modeling for designing case based reasoning in network intrusion detection prototype case base has 40 description attributes and 2 solution attributes. Solution attribute is used after finding best-selected cases and show the type of attacks. Appendix VI shows the description of case attributes regarding name, data type, weight and local similarity.

Once case the structures configured in COLIBRI Studio, CBR system must access the stored cases in an efficient way. COLIBRI Studio splits the problem of case base management in two

separate although related concerns: persistency mechanisms through connectors and in-memory organization.

A connector is an object, which has the ability to access and retrieve cases from a specific case persistency when given case structure, and give those cases to the CBR system in a standardized way. Because of this, COLIBRI Studio can deal with any case persistency as long as a connector is provided [13].

5.1.1. Managing Tasks

For the development of case based network intrusion detection, the researcher used core package tasks such as PreCycle, main CBR cycle and PostCycle. The first step, PreCycle task gets all the cases in case base. Therefore, it is necessary to define dataset path of connector in its subtask. There is only one subtask called obtain case task and it is used to retrieve data from case base before the execution of the main CBR cycle.

Main CBR cycle is the main task of CBR cycle and it has sub tasks. The developer has to give path of case structure in it. It knows number of case attributes that are available. It is called obtain query task. In addition to obtain query task, there are other significant tasks under the main CBR cycle. These are retrieve tasks, reuse tasks, revise task and retain tasks.

Retrieve tasks used to retrieve case(s) from the stored case base. Retrieve tasks also decomposed in to different subtasks. The subtasks include select working cases task, compute similarity task and select the best case. Select working case task selects cases from case base and stores them into current context. Compute similarity of the stored cases with the case entered by the user using the query window. Select best case shows the best matched of case(s) after computing the similarity of stored cases against the new case. It means that the number of best-matched case(s) is shown to the user depending on the method used and the threshold.

Reuse tasks enable to reuse previously stored cases. It has three subtasks such as prepare cases for adaptation task, atomic reuse task and reuse task. Prepare cases for adaptation task select cases from case base and stores them into context. Here also specifying the path of case structure in this method is needed. Atomic reuse task should be resolved by reuse resolution method.

Revise task is the evaluation stage about the selected solution in reuse phase. After selecting the most similar cases from the retrieved results, the solution for the problem should be confirmed and validated before the solution is stored for future use.

Retain tasks also used to CBR case retention on a persistence layer. It has also its own subtasks like select cases to store task and store cases task. Select cases to store task give authentication to the user for storing case. The store cases task enables to store case(s) into the case base. The last task in managing tasks in jCOLIBRI is PostCycle. PostCycle task have only one sub task called close connectors task, which is usually executed after the main CBR cycle. Its main task is to close a connection between case base and GUI. The last task in managing tasks in jCOLIBRI is PostCycle. PostCycle task have only one sub task called close connectors task which is usually executed after the main CBR cycle. Its main task is to close a connection between case base and GUI.

To integrate automatically the K-mean clustering model (constructed by data mining) with Case Based system [13], COLIBRI Studio, Eclipse IDE Version: 3.7.0 with JDK 6.0 is used.

Before applying the first process of integration, the researcher create cbr java packages and then create class for CBRApplication and link to k-means clustered result to automatically integrated to CBR. After CBRApplication and clustered result are applied, the next step could be experiment on COLIBRI Studio. A CBR application in jCOLIBRI must implement or extend the jcolibri cbrapplications standard CBRApplication interface [13].

The second layer of Case Base management is the data structure used to organize the cases once loaded into memory. The CBR methods to access the cases in-memory organization of the cases is jcolibri.cbrcore.CBRCaseBase method should be imported. In-memory organization has three different types of structure: jcolibri.casebase.LinealCaseBase: Basic Lineal Case Base that stores cases into a List. jcolibri.casebase.CachedLinealCaseBase: Cached case base that only persists cases when closing the application. jcolibri.casebase.IDIndexedLinealCaseBase: Extension of LinealCaseBase that also keeps an index of cases using their IDs.

The system has the capability of retrieve, reuse, revise and retain cases. The retrieval task starts with a new case description and ends when best matching similarity case of attacks or normal cases has been found in the case base. As users enter case description (query) through interface, the system searches the best matching cases from the case base and return possible solution in ranked order to the problems described in the query. If exact matching occurred in between the query and cases in the case base, users can directly derive the solution or if the similarity or matching is partial, then adaptation, revision and reuse tasks are done to make the proposed

solution best for the problem at hand. At last, that best-modified solution to the problem stored to the case base for future use.

5.2. Mapping Predictive Modeling in to Rule Based Reasoning

In this study, a predictive model created by JRip classifier algorithm is used. This classifier has generated 23 rules shown in table 5.1. The rules involved 20 features/attributes among the 41 features/attributes from the sample data set. The algorithm generated 22 rules for the attacks namely, probe, DOS, R2L and U2R attacks and only one rule for normal behavior. It can be deduced from the rules that if a certain incident satisfies one of the 22 rules it is an attack otherwise it is a normal network incident. Among the attack classes, all but U2R has more than one rule. This is related to the number of instances occurred in the sample data set for U2R is the smallest as compared to the others.

In consultation with domain experts in the area of network administration, the rules are evaluated to make sure that whether or not they tell us about network behaviors. Based on the evaluation, the rules are capable of identifying attacks but question is raised that the algorithm has only used 20 features among the 41 by ignoring more than half of the features. Domain experts raised the idea that the ignored or pruned features have contribution on identifying possible attacks. Hence the automatic knowledge acquisition task takes into account rules generated from JRip classifier in the integration of data mining induced knowledge with knowledge based system.

Rule #	**Rule**	**Class**
Rule 1	(root_shell >= 1) and (duration >= 25) => class=U2R (9.0/1.0)	U2R
Rule 2	(num_failed_logins >= 1) => class=R2L (52.0/0.0)	R2L
Rule 3	(service = ftp_data) and (flag = SF) => class=R2L (25.0/0.0)	R2L
Rule 4	(service = imap4) and (dst_host_count <= 11) => class=R2L (11.0/0.0)	
Rule 5	(duration >= 12) and (dst_host_same_src_port_rate <= 0) => class=R2L (5.0/0.0)	
Rule 6	(num_access_files >= 1) and (src_bytes <= 116) => class=R2L (5.0/0.0)	
Rule 7	(is_guest_login = 1) and (duration <= 1) => class=R2L (2.0/0.0)	R2L
Rule 8	(src_bytes <= 8) and (dst_host_serror_rate <= 0.99) and (dst_host_same_src_port_rate >= 0.34) => class=Probe (3703.0/4.0)	Probe

Rule 9	(dst_host_rerror_rate >= 0.07) and (dst_host_same_srv_rate <= 0.81) => class=Probe (1611.0/0.0)	
Rule 10	(protocol_type = udp) and (src_bytes >= 100) and (service = private) => class=Probe (250.0/0.0)	Probe
Rule 11	(dst_host_srv_count <= 3) and (count <= 63) and (dst_host_count >= 99) and (count <= 44) => class=Probe (158.0/1.0)	
Rule 12	(diff_srv_rate >= 0.37) and (count >= 6) => class=Probe (37.0/0.0)	Probe
Rule 13	(dst_host_srv_count <= 8) and (count <= 5) and (dst_host_count >= 145) => class=Probe (8.0/0.0)	
Rule 14	(dst_host_diff_srv_rate >= 0.5) and (dst_host_srv_count <= 1) and (dst_host_diff_srv_rate <= 0.67) => class=Probe (6.0/0.0)	
Rule 15	(protocol_type = icmp) and (src_bytes <= 20) => class=Probe (5.0/0.0)	
Rule 16	(count >= 49) and (dst_bytes <= 0) => class=DOS (6977.0/0.0)	DOS
Rule 17	(src_bytes >= 21048) => class=DOS (766.0/0.0)	DOS
Rule 18	(dst_bytes <= 0) and (count >= 3) and (dst_host_count >= 75) => class=DOS (122.0/1.0)	
Rule 19	(flag = S0) => class=DOS (16.0/0.0)	
Rule 20	(wrong_fragment >= 1) => class=DOS (4.0/0.0)	
Rule 21	(flag = RSTR) => class=DOS (4.0/0.0)	
Rule 22	(service = ecr_i) and (src_bytes >= 1032) => class=DOS (3.0/0.0)	
Rule 23	(Duration=0) => class=normal (21999.0/4.0)	Normal

Table 5.1 Rule set generated using JRip from sampled data set[7]

As we have seen above in table 5.1, JRip algorithm generates rules in the form of *(condition)…(conclusion)* format. The *condition* part contains attribute, comparison operator and value. Two or more conditions are joined by _*and*'. After the conditions the '=>' meaning implies follows. The conclusion part of the rule has the format *class='Attack_type'*, for example *class=R2L, class=probe.*

Hence, for a certain network incident to be classified as an R2L attack both the antecedent of the rule *((is_guest_login = 1) and (duration <= 1))* should be true. In other words, if guest has logged in and stayed less than or equal to one second then that network incident is an R2L attack. If either of them is false then the conclusion *(attack class =R2L)* will be false.
However, PROLOG does not work in IF… THEN format, rather it works in reverse order. PROLOG starts with a goal and then goes to the facts that can proof the goal as true. Therefore, the above rule has to be formatted as: *attack (u2r):- (is_guest_login = 1),(duration =< 1).*

Generally, the result of JRip we found has different structure and we replaces of convert to the above form that can be supported by PROLOG. The tokens in JRip and PROLOG rules is attached in the appendix.

The diagnosis of network incident is aimed at identifying it as an attack (R2L, probe, DOS, U2R) or normal behavior of network. It is based on rules and facts in knowledge base. Combined reasoning system asks question the user via displaying attributes with their respective values in the rule set. The user answers _yes' or 'no' by comparing the values in the incident with the questions asked from the system.

The KBS is implemented as modules containing the knowledge base, asker module, and attack description module.

Rule Base: - this module is a collection of rules automatically constructed via the integrator application. For this study, the selected classifier has generated 22 rules about attacks and the last one is about normal behavior. The rule base contains one rule for U2R attack types, six rules for R2L attack types, eight rules for probe attack types and seven rules for DOS attack types

```
Rule 1: attack(u2r):-root_shell(root_shell>=1) ,duration(duration>=25) .
Rule 2: attack (r2l):-num_failed_logins(num_failed_logins>=1) .
Rule 3: attack (r2l):-service (service=ftp_data) ,flag(flag=sf) .
```

KBS first checks rules for attacks. If any of the 22 rules don‘t validated as true for a certain incident, then that incident is identified as normal.

Asker module:- this module is built aiming for creating interaction with user. RIDA-KBS presents question to the user using this module. Asker module is designed in a manner to accommodate any changes in the rules and facts. The questions displayed are based on the contents of rule and fact bases. Whenever a change in either of the two appears, the question asked also changes accordingly.

```
1 ?- start.
write Attack:
|: attack.
does attack has num_failed_logins >= 1 (y/n) ?y
does attack has service=ftp_data (y/n) ?y
does attack has flag=sf (y/n) ?y
attack type is r2l  !.
```

Following the successful integration of induced knowledge with the knowledge based system, the rule based intrusion detection KBS is built.

5.3. Combination of Case Based and Rule Based Reasonings

This paper is concerned with the combination of RBR and CBR in an uncertain and dynamic network/Internet world to detect network intruders. Rather than "patching together" two different types of representational and reasoning frameworks, we have chosen to attempt the integration within one architectural framework as discussed in the third chapter of this study.

Once rule-based and case-based reasoning's are constructed from predictive models and descriptive models respectively, an experiment is made to decide the order of using them for network intrusion detection.

Basically, there are two ways of combining the two reasoning techniques complement each other. Either rule-based or case-based reasoning is applied first and then followed by the other as show in figure 5.1. In this, if the solution given by the reasoning mechanism used in the first step is accepted, then it is used as a solution of the given problem. Otherwise, the other reasoning is consulted. To this end, we used a conditional combination model [42]. See figure 5.1.

Combination of Reasoning Models

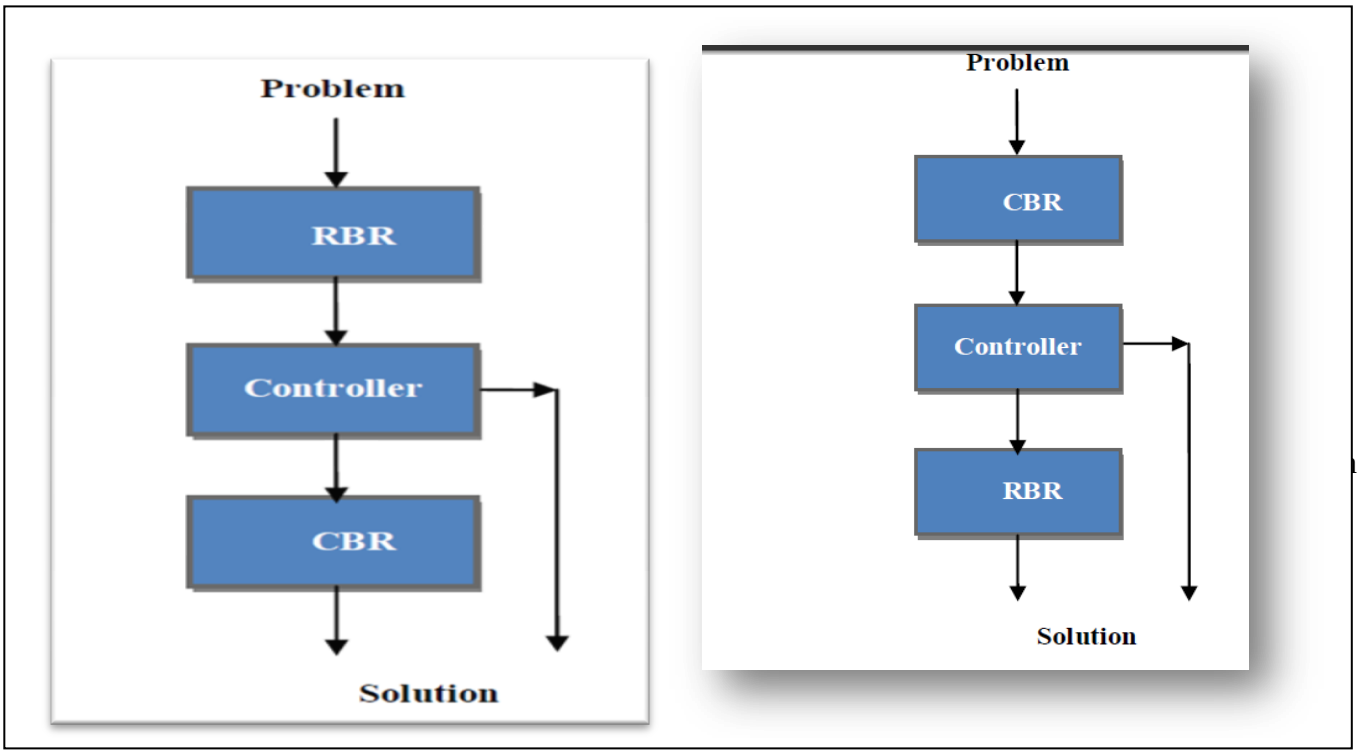

Figure 5.1 conditional combination model of CBR and RBR

5.4. Selecting combination model of RBR and CBR

From the different model of combination, based on the following logics and experimental results bellow, we used a conditional combination model, RBR-Controller-CBR combination.

Among the different conditional combination models, the RBR-Controller-CBR model of combination is selected [42]. In this model, the combined system works if the solution given in the first step is acceptable then it is used as a solution of the given problem & otherwise next steps are invoked [42]. In this research, we integrate CBR and RBR using this model. It is because of the different advantages. As stated by [39] and [42] the following are the advantages of RBR-Controller-CBR model of combination.

- RBR module uses more general knowledge to solve situations not reflected in CBR. Therefore, to solve problems of a given area, it is common to start from the general knowledge.
- RBR is a collection of rules/facts and used exact matching of problems to solve. If the problem matches, the solution will be 100% acceptable or correct, but CBR measures similarity.
- Allows shortcuts in reasoning. If appropriate rules can be found, problems can often be solved in much less time than it would take to generate a solution from cases.
- RBR cannot learn or adapt cases for further reasoning, but CBR do.

5.4.1. Experiment result

To choose which model of combination is more preferable in the network detection process, an experiment is conducted using the same test dataset and the performance of the individual reasoning model is compared. This comparison helps to see what model of combination has to take to enhance the already developed intrusion detection systems.

The experiment is done for:

- Rule-based reasoning system
- Case- based reasoning system
- CBR followed by RBR system and
- RBR followed by CBR system

The table bellow depicts the comparison of the performance of the individual reasoning systems listed above Vs percentage correctly and incorrectly identified type of attack.

Reasoning model	% of correctly identified	% of incorrectly identified
RBR	81.2%	18.8%
CBR	86.3%	13.7%
CBR-RBR	86.3%	13.7%
RBR-CBR	87.66%	12.34%

Table 5.2 Experiment result of comparison of individual reasoning models

As clearly seen in the table (Table 5.2) the first experiment is undertake RBR and CBR independently. From the experiment, the two reasonings gives 81.2% and 86.3% correctly identified result respectively.

The second experiment is undertaken for combined reasonings, the performance for identifying the type of intruders correctly scores 86.3% and 87.66% respectively.

From the experimental result, a combination reasoning model of Rule Based Reasoning followed by Case based Reasoning (RBR-CBR) performs better than the other model of reasoning's by achieving 87.66% correctly classified network intruders. The other important finding in this experiment is that, CBR alone and Case based reasoning followed by Rule Based Reasoning (CBR-RBR) score equal 86.3% correctly identified. It is because the case based reasoning component always suggests solution by measuring similarity, so that rule based reasoning do not get the chance to detect intruders.

So, in this study, the RBR-CBR combination approach is selected and used for implement a prototype network intrusion detection system.

CHAPTER SIX

IMPLEMENTATION AND EVALUATION

So far, natural grouping of intrusion are extracted from sampled collection of intrusion data set and knowledge base is constructed as case base and rule base that contains case and rules of network incidents. Hybrid Network Intrusion Detection System (HNIDS), is then designed that automatically detect a network incident as Normal, Probe, DOS, U2R and R2L.

6.1. Network intrusion detection process

In this study, the conditional combination of CBR and RBR took place by using Java eclipse to combine the two mechanisms. The detection process is undertaken by interacting with the user through presenting a serious of questions for the user. The system asks the user by displaying a yes/no question for rule and questions containing attributes with their values for case based. The user is expected to reply for the questions. Once the first reasoning i.e, RBR detects network incident as Probe, DOS, U2R and R2L attacks, it display the result. If the RBR do not detect the type of attack, the query directly move to the CBR and the CBR measure similarity, retrieve cases, display result, adopts cases, refine/revise case and retain at the end [13].

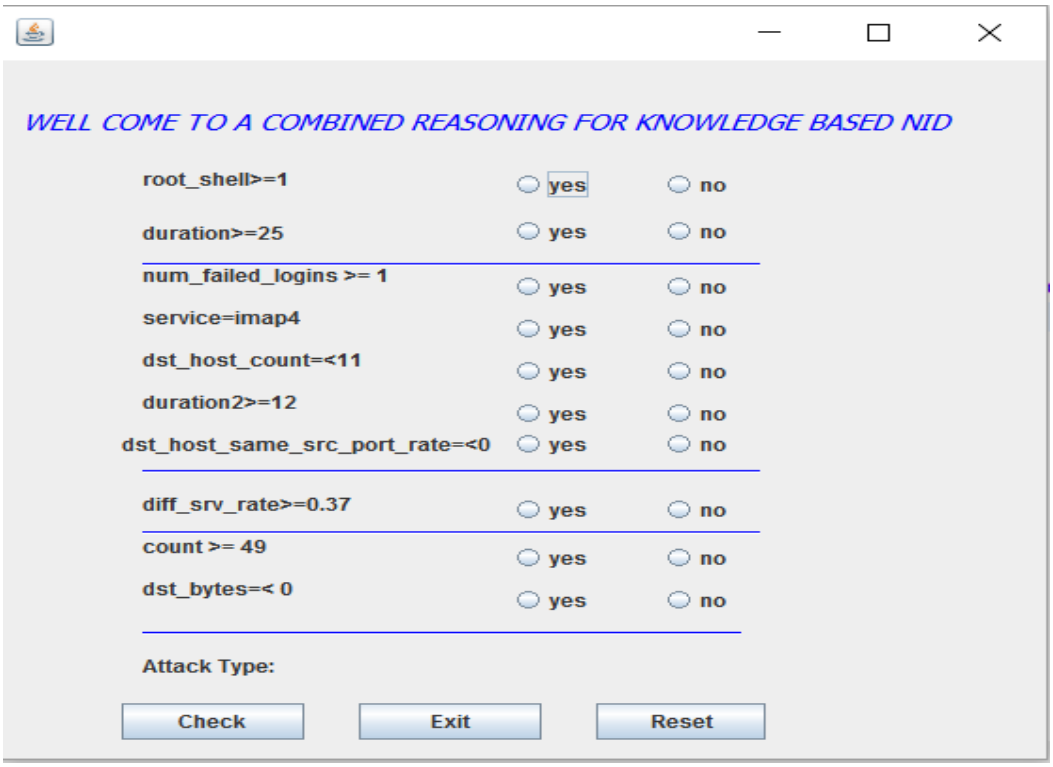

Fig 6.1 User Interface of the Combination of CBR and RBR

As clearly stated above, the first reasoning of the integration is CBR. So, figure 6.1 shows us, the user interface contain some of the rules generated by the JRip algorithm. The attributes are some of the rules used to determine the type of attack which are selected as per generated by the JRip [7].

On the above interface, It allows the user to select the radio button as per the new query and hit the check button, if the condition is satisfied, the type of attack will display at the provided space, like " Attack type: U2R".

(root_shell>=1)and (duration>=25)=>class=U2R

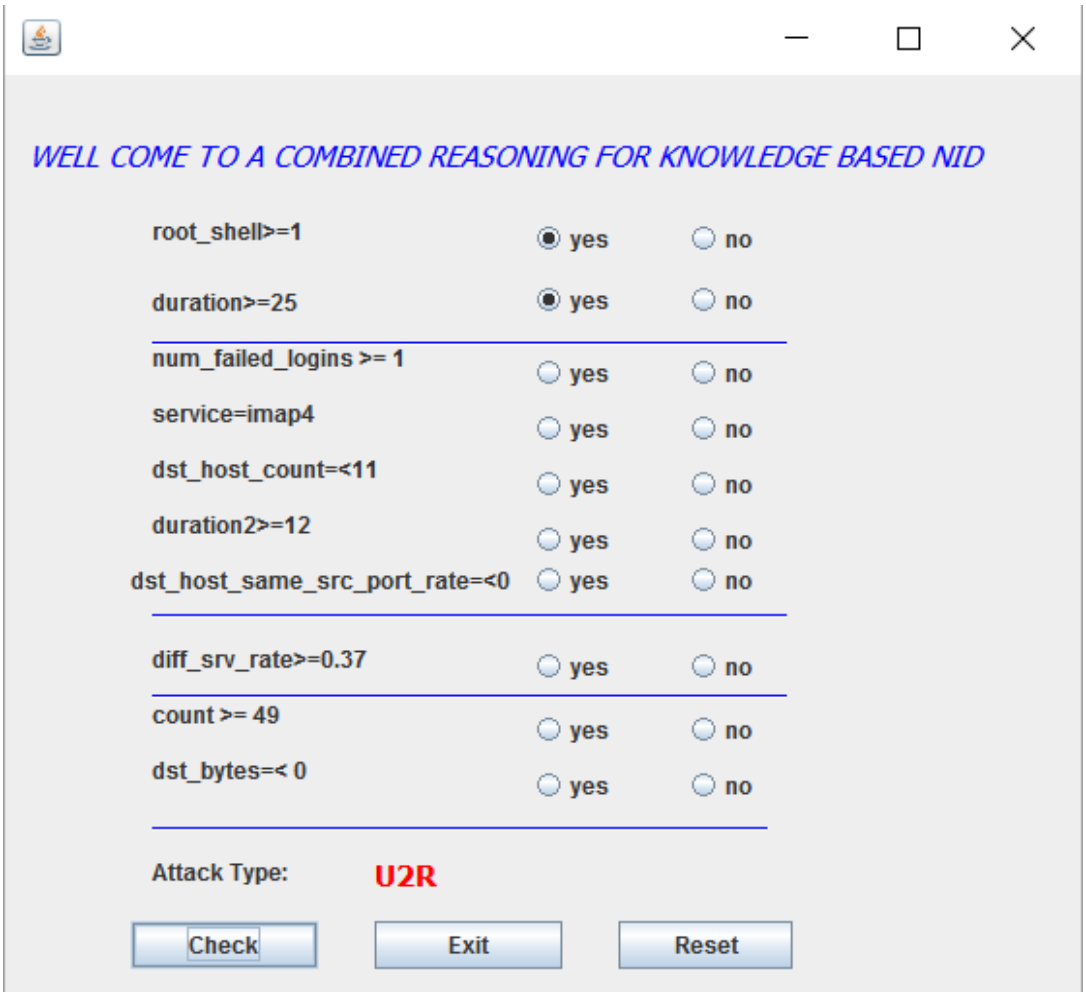

Fig 6.2 A combined user interface detecting intruder type U2R

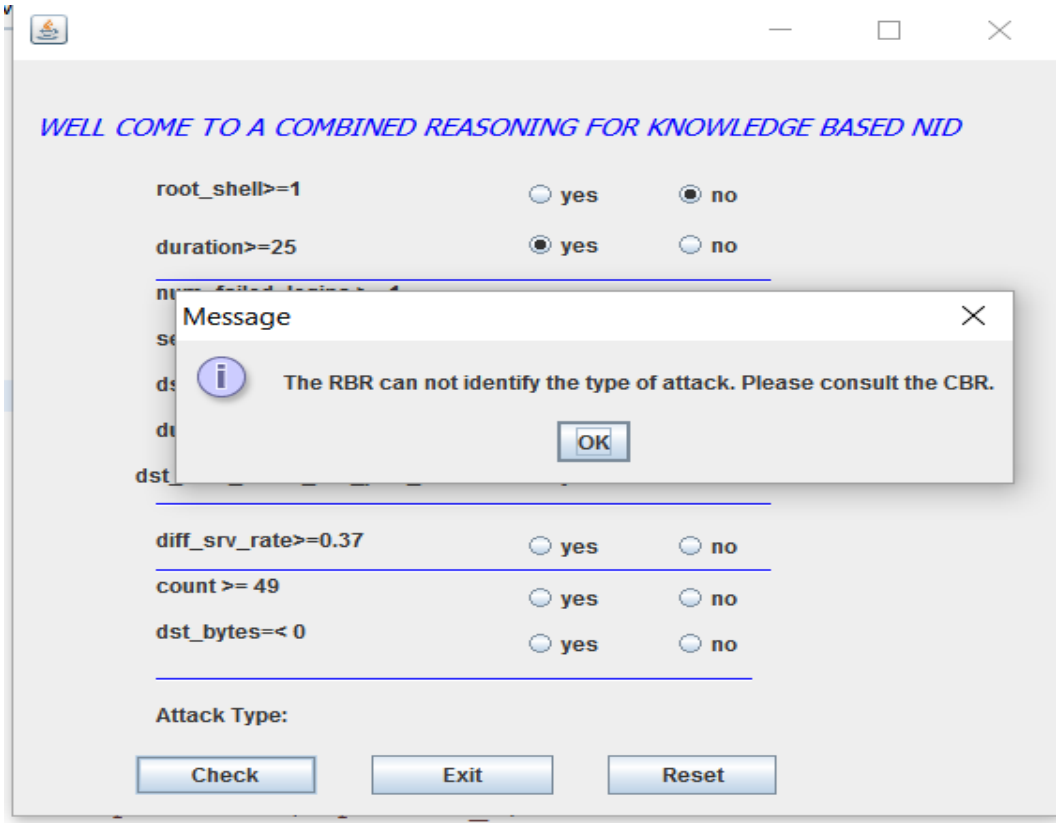

Figure 6.3 User interface of combined system showing the RBR cannot identify the type of attack

As shown in the above figure 6.3, if one of the selection of the attribute is “no”, the RBR asks or give a privilege to consult the CBR” The RBR is not able to identify the type of attack. Please consult the CBR” if “ok” is selected it automatically goes to the CBR query dialog box. The CBR query interface is displayed automatically and diagnose the type of attack.

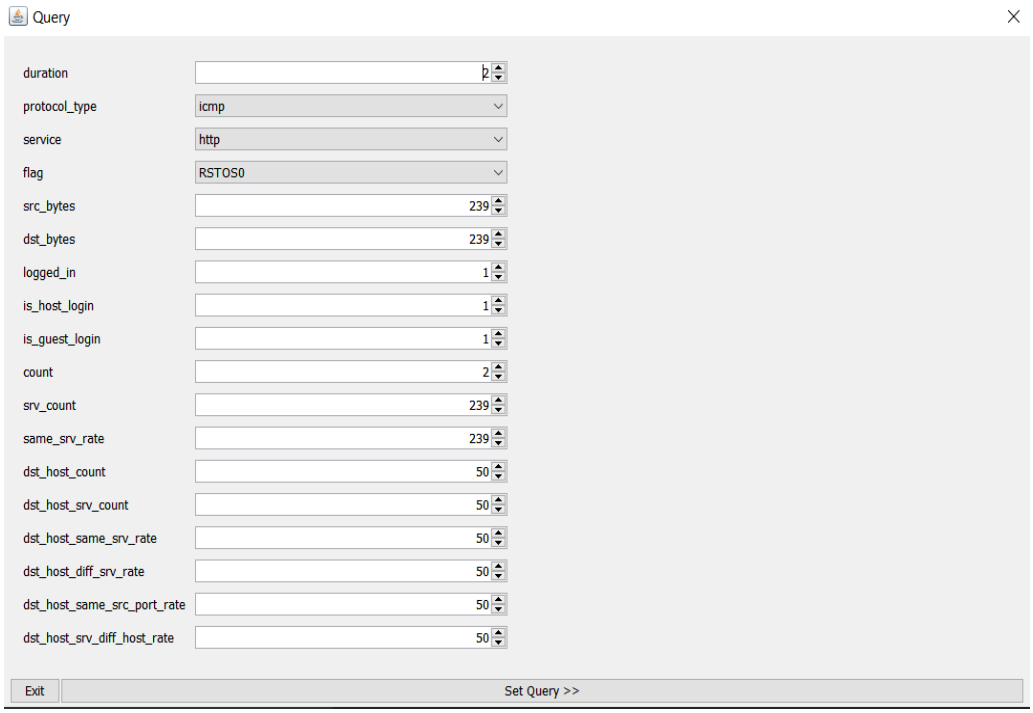

Fig 6.4. The query interface of the CBR

The CBR, Consider the value of the attributes, retrieve cases, measure similarity and display the type of attack. Retain and adapt is also took place [13].

6.2. Evaluation of the system

In order to assure the combined NID meets the requirement it is developed for both system performance testing and user acceptance testing are performed.

6.2.1. System Performance Testing

The performance of the combined system is evaluated by preparing test cases. The test cases include samples of intrusion instances taken from KDDcup‘99 intrusion data set. The instances include 30 attributes with their respective values. The test cases, which are unlabelled intrusion instances are delivered to domain experts to label them as normal, probe, U2R, R2L and DOS.

Considering the different factors, the researcher prepared only 30 test cases/instances for system performance testing. Attributes of instances with their respective values describe the behavior of certain network incident. Based on the attributes and their respective value, domain experts labeled the instances. These set of test instances are provided to combined system NID and the outputs are compared to the domain experts‘ judgment.

Confusion matrix is used for comparing the performance of Combined-NID in two ways: first comparing the performance of Combined-NID with domain experts'judgment and second comparing the performance of Combined-NID with test case.

In the confusion matrix (see table 6.1) system prediction Vs experts suggestion about the type of attack.

	Performance of Combined reasoning system						
Domain experts feedback	**Attack Class**	**Normal**	**Probe**	**DOS**	**R2L**	**U2R**	**Total**
	Normal	6	0	0	0	3	**11**
	Probe	0	5	0	0	0	**5**
	DOS	0	1	5	0	0	**6**
	R2L	0	0	0	6	0	**6**
	U2R	0	0	0	0	4	**2**
	Total	**6**	**6**	**5**	**6**	**7**	**30**

Table 6.1 Confusion matrix for evaluation of Combined-NID compared to experts'feedback

The confusion matrix on table 6.1 shows matrix of test cases evaluation by combined-NID and domain experts' suggestion. The rows illustrate evaluation of domain expert and the columns illustrate result of Combined-NID.

The entry under column *Normal* indicates that the system identified six instances as normal. The entries under column *Probe* testified that out of six instances, five of the instances are correctly identified as probe attack and one instances are incorrectly identified as DOS.

The entries for *DOS* and *R2L* columns show that the system has correctly identified five instances as DOS and six instances R2L attack types respectively. The entries in the confusion matrix under *U2R* column depict that three instances out of seven are incorrectly identified by the system as Normal and four instance is correctly classified as U2R.

The system has correctly detected 28 test instances out of 30 to their correct class. This means the system has 93.33% detection accuracy.

But, this measure alone is not enough to measure performance of the system since it only tells us the overall performance. Hence, Precision and Recall are employed to evaluate the system performance apart from detection accuracy. Recall is the proportion of real positive cases that are correctly predicted positive. Precision denotes the proportion of predicted positive cases which are correctly real positives [43].

As clearly illustrated in table 6.2, the system's performance is evaluated in term of Precision, Recall and F-measure, which enables us to view in detail how accurate, is the system is identifying network attacks and normal incidents.

	Precision	**Recall**	**F-Measure**	**Attack Class**
	0.89	0.87	0.80	Normal
	0.83	1	0.90	Probe
	1	0.87	0.93	DOS
	0.8	0.85	0.66	R2L
	1	1	1	U2R
Average	0.90	0.91	0.86	

Table 6.2 Performance evaluation based on Precision, Recall and F-measure

The system has registered its highest 100% Recall for Probe and U2R attacks and 87% for Normal and DOS, and 85% for R2L attacks. The system registered its highest 100% Precision for DOS and U2R respectively. The system has 89% Precision for Normal, 83% for Probe and 80% for R2L. As compared to Precision (90%), the system has performed well in its Recall (91%) for identifying network incidents accurately.

6.2.2. User Acceptance Testing

The aim of undertaking user acceptance testing is that to make sure how well combined-NID is performing on the eyes of users so as to make sure that the system is accepted and usable by users.

Five domain experts are selected to test the system by responding a series of questions. This experts are taken from Addis Ababa University ICT office network administrator staffs. The experts, after providing training how combined-NID works, are given test cases/ instances to use and evaluate the system. The evaluators assessed combined-NID by using the following standards [7] [13].

- Simplicity to use and interact with the system
- Attractiveness of the system
- Efficiency in time
- The accuracy of the system in reaching a decision to identify the types of network attacks
- The ability of the system to make right conclusion and recommendation
- Importance of the KBS in the domain area

Different researchers have used different types of user acceptance testing evaluation criteria. But for this study, the evaluation criteria suggested by Abdulkerim [7], Dawit [13] has been used such as Excellent = 5, Very Good =4, Good =3, Fair =2 and Poor =1

Table 6.3 depicts that summary of domain experts‘ evaluation of the system.

No	Criteria of evaluation	Poor	Fair	Good	Very Good	Excellent	Average
1	Simplicity to use and interact with the system	0	0	1	1	3	4.4
2	Attractiveness of the system	0	0	3	1	1	3.6
3	Efficiency in time	0	0	0	0	5	5
4	The accuracy of the system in reaching a decision to identify the types of network attacks	0	0	2	3	1	4.6
5	The ability of the system to make right conclusions and recommendation	0	0	1	2	2	4.2
6	Importance of the KBS in the domain area	0	0	0	2	3	4.6
		Total Average					4.4

Table 6.3 User acceptance evaluation form adapted from [17]

As we can see from the above table, most (60%) of the evaluators replied excellent, good, and very good 20% each for **simplicity to use and interact with the system**. It is because of the interface developed by java and simple to use or operate.

For the **attractiveness of the system,** 60% of respondents replied Good and the same percentage, 20% Very Good and Excellent for it. With regard to **Efficiency in time** 100% of evaluators replied excellent.

Concerning **the accuracy of the system in reaching a decision to identify the types of network attacks,** 40% of evaluators scored Good, 60% of evaluator scored very good and 20% of them as excellent. The ability of the system to make right conclusions and recommendation achieves 40% Very good and 40% excellent.

According to the evaluation filled by domain experts, Combined-NID has registered 88%user acceptance. This can be taken as a very good achievement. The domain experts also suggested the need to develop full package system that can detect automatically from the network.

Generally, in this research the conditional integration of CBR and RBR is took place by using a java eclipse interface to come together the two reasoning’s. From the result of the combination, we have seen that the performance of the intrusion detection capability of the system is much better than the CBR and RBR independently. Appling different algorithms of descriptive and

predictive modeling on WEKA to generate cases and rules, having the rules and cases at hand, using the cases and rules to develop a combined intrusion detection system is much more efficient and effective. It is clear that each type of attacks has their own way of attacking, causing damages to the victim computer and dynamic changing behavior of the attack . Identification of each class of attacks to their correct class is important to provide proper advice to network administrators so that they can respond appropriately.

As per the hypothesis and experimental result of this research, the combination of RBR and CBR perform better decision and result for identifying network intruders. The result of case based reasoning alone as reported by Dawit [12] performs 87%. On the other hand, the report of the result of rule based reasoning system by Abdulkerim [7] for network intrusion detection has scored 80.5% overall performance. The following table shows the comparison.

Reasoning system	**% performance**	**Difference with Combined system**
CBR Dawit [12]	87%	6.33%
RBR Abdulkerim [7]	80.5%	12.83%
Combined reasoning system	93.33%	

Table 6.4. The comparison of the performance of reasoning systems.

As shown in table 6.4, comparing with to the result of above two research works with the combined system, the performance of combined reasoning system for network intrusion detection score 93.33%. which is a very good enhancement and improvement for detecting network intruders.

Generally, combining case based and rule based reasoning system RBR followed by CBR for network intrusion detection has better performance for detecting network intruders than CBR and RBR can do independently.

CHAPTER SEVEN

CONCLUSION AND RECOMMENDATION

7.1. Conclusion

In spite of the wide growth of information technology, security has remained one challenging area for computer and networks. The numbers of hacking and intrusion incidents are increasing year on year as technology rolls out.

With the Internet playing a vital role in incessant communication, its effectiveness can diminish owing to effects called intrusions. Intrusion is an activity that adversely affects the targeted system. An intrusion may compromise the integrity, confidentiality and availability of resources of the attacked system. There are different ways of detecting and preventing intruders in the network. Among the methods used to detect. Knowledge Based System (KBS) and Data Mining (DM) are most wildly used. The integration of KBS and DM approach is a new research area for different applications.

In this work, a combined reasoning system for knowledge-based network intrusion detection has been designed. The combination of two reasoning systems, i.e. Case Based Reasoning (CBR) and Rule Based Reasoning (RBR) system is employed. To construct cases for a case based reasoning a descriptive modeling and for generating rules for a rule based reasoning predictive modeling are used. The KDD CUP 1999 Dataset has been used to train and test the combined reasoning system for knowledge-based-NID proposed in this research work.

After independently developing a RBR and CBR systems, the integration of the two reasoning systems took place. The method of integration used is a conditional integration model, which has a controller in between RBR and CBR. The controller or user interface is developed by Java eclipse programming language. In the integrated system, the first reasoning system that first treat a new query is RBR system. If RBR model gets a rule that exactly matches with the new query, it gives solutions and the solution is accepted as correct solution. If not, it automatically forward to the CBR system. The CBR model includes case retrieval; the similarity measuring stage, reuse; which allows domain expert to transfer retrieval case solution to suit for the current case, revise; to test the solution and retain to store the confirmed solution to the case base for future use.

The performance of the system shows that it achieves 90.5% accuracy with an average Precision and Recall of 90% and 91% respectively. User acceptance testing also shows on the average an excellent acceptance. This shows the system has registered a promising result.

The strength of this study is the reasoning system to detect intruders in a combined manner so that the probability of the intruders to be detected is high. In combination of RBR and CBR, due to their interchangeable nature, they provides effective knowledge representation, problem solving power, and exceeding one's weakness with the other. The fruit of the complementary nature of the reasoning systems also proposed here for network intrusion detection. The weakness of the proposed system is it is not automatically integrated, to accept query and automatically distribute the query for both reasoning systems at the same time, compare the results of the independent reasoning and to one best solution.

In general, the combination of rule-based and case-based reasoning methods has shown a substantial improvement with regards to performance over the individual methods. But, further investigation is required regarding the usage of local dataset, including advisory system with the combined system, integrating the system with network to automatically detect intruders, consider the dynamic changing behavior of the intruders from time to time and updateability of the knowledge base.

7.2. Recommendation

In future, based on the results of this study, it is possible to provide extensions or modifications to achieve further increased performance. The following recommendations are made for the way forward:-

- ✓ An integration that accept query, send the query for both reasoning at the same time, compare the result and select the most relevant solution is also an other research issue that needs further investigation.
- ✓ The combined intrusion detection system can be extended as an intrusion prevention system to enhance the performance of the system.
- ✓ Integration of the system with the network for automatic detection is another issue that is open for further research.
- ✓ In this study, KDD cup'99 dataset (International dataset) is used. Using local dataset is highly recommended.

- ✓ Further combinations such as artificial intelligence, soft computing and other clustering algorithms can be used to improve the detection accuracy and to reduce the rate of false negative alarm and false positive alarm.

Reference

[1] S.K Miller, "An Introduction to Computer Security," IEEE Computer, vol. 38, no.34, 2001.

[2] Ahmed H. Fares and Mohamed I. Sharawy, "Intrusion Detection: Supervised Machine Learning," Computing Science and Engineering, vol. 5, no. 4, pp. 305-313, December 2011.

[3] Usman A., Sajjad H., Salman N., and Obaid U., "A Survey of Intrusion Detection and Prevention Techniques," *International Conference on Information Communication and Management*, vol. 16, pp. 23-36, 2011.

[4] Sumit M., Mary M., Anupam J., and Tim F, "A Knowledge-Based Approach To Intrusion Detection Modeling," in *Computer Science and Electrical Engineering*, Maryland, 2006, pp. 19-30.

[5] Hervé D, "An Introduction to Intrusion-Detection Systems," in *Proceeding of connect'2000,* Zurich, 2002, pp. 1-18, IBM Research, IBM Reseach Laboratory.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth., "Knowledge discovery and data mining: Towards a unifying framework," in *Knowledge Discovery and Data Mining*, California, 1996, pp. 82-88.

[7] Abdilkerim M, "Towards Integrating Data Mining with Knowledge Based System: The Case of Network Intrusion Detection," School of Information Science, Addis Ababa University, Addis Ababa, MSc Thesis 2013.

[8] Huy A. and Deokjai C., "Application of Data Mining to Network Intrusion Detection: Classifier Selection Model," in APNOMS, Berlin, 2008, pp. 399–408.

[9] Carsten P, "Integrating and Updating Domain Knowledge with Data Mining," Leipzig Graduate School of Management, vol. III, pp. 50-62, June 2000.

[10] Kumar Ranjit, Research Methodology-A Step-by-Step Guide for Beginners, 2nd ed. Singapore: SAGE, 2005.

[11] SEO and PPC Mangement Solution. (2015, February) SEO & PPC Mangemnet Solution. [Online]. Http://www.theecommercesolution.com/useful_links/KBS_Data_Mining.php

[12] Jim Prentzas and Ioannis Hatzilygeroudis, "Categorizing Approaches Combining Rule-Based and Case-Based Reasoning," vol. 24, pp. 97-122, May 2007.

[13] Dawit Hassen "Integrating Descriptive Modelling with Case Based Reasoning in Network Intrusion Detection" Machine Learning, School of Information Science: Addis Ababa University, M.Sc Thesis 2015.

[14] Janet Kolodner, Case-Based Reasoning, 2nd ed. California, San Mateo: organ Kaufmann, 1992.

[15] Paul Dokas and Levent Ertoz, "Data Mining for Network Intrusion Detection," IEEE Computer, vol. 2nd, no. 12, pp. 18-31, 2001.

[16] Armin Stahl and Thomas R., "Rapid Prototyping of CBR Applications with the open source tool

myCBR," in German Research Center for Artificial Intelligence, Berlin, pp. 615-629, 2008.

[17] Tigabu Dagne, "Constructing Predictive Model for Network Intrusion Detection," School of Information Science, Addis Ababa University, Addis Ababa, MSc Thesis 2012.

[18] Agnar Aamodt, "Case-Based Reasoning - An Introduction," in Case Based Approchces , Dragvoll, vol. 6, no.3, pp. 39-49, August 1991.

[19] Ferri C., Flach P., and Henrandez-Orallo J, "Learning Decision Trees Using Area under the ROC Curve," in the 19th International Conference on Machine Learning, Chicago, pp. 139-146, 2002.

[20] Fadzilah Siraj, "Mining Enrollment Data Using Descriptive and Predictive Approaches," in *Knowledge-Oriented Applications in Data Mining*, Kimito Funatsu, Ed. Malasiya, Malasiya: InTech, Janualry 2011, ch. 4, pp. 53-73.

[21] Prof. Mihaela, "On the Use of Data-Mining Techniques in Knowledge-Based Systems," in Economy Informatics, Romania, pp. 22-24, 2006.

[22] Kola Sujatha "Data mining approach for hybrid intrusion detection system," Anna University, Faculty of Information and Communication Engineering, Ph.D Dissertation 2014.

[23] Sanket R Jain "Information Security: Intrusion Detection and Prevention System" 2014 [online]: http://sanketrjain.com/intrusion-detection-and-prevention-system/.

[24] Jaiganesh "Investigation on machine learning algorithms for network intrusion detection system," Department of Computer Science & Engineering, Manonmaniam Sundaranar University, Ph.D Dissertation July 2014.

[25] Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, A.J., and Tan, P.N. "Data Mining for Network Intrusion Detection." IEEE, 1987.

[26] *Intrusion Detection Systems*, 6th ed. New York: Information Assurance Tools Report IATR, 2009.

[27] Khandelwal "Knowledge based systems, problem solving competency and learnability," Suresh Gyan Vihar University, Department of Computer Science, 2014.

[28] Azeb Bekele "Integrated Case Based and Rule Based Reasoning for Decision Support," Norwegian University of Science and Technology, Department of Computer and Information Science, M.Sc Thesis 2009.

[29] Robert S. Engelmore and Edward Feigenbaum, Japanese Technology Evaluation Center, Knowledge Based Systems in Japan, March 21,2009, http://www.wtec.org/loyola/kb/toc.htm.

[30] Alex J. Champandard , AI depot , Artificial Intelligence , October 23,2008, http://aidepot.com/Intro.html.

[31] Jochen Schäfer"Capture, Distribution and Evolution of Information Needs in a Process-Oriented Knowledge Management Environment," University of Kaiserslautern, Department of Computer Science, 2003.

[32] D. Shimin, S. Huizhang, L. Hong"Research on Case-Based Reasoning Combined with Rule Based

Reasoning for Emergency", IEEE 2007.

[33] Buchanan, B. G., and Shortliffe, E. H., *Rule-Based Expert Systems*, Addison Wesley, Reading, MA, 1984.

[34] Alex J. Champandard, AI depot, Artificial Intelligence, October 23, 2008, http://aidepot.com/Intro.html .

[35] Cindy Marling, Edwina Rissland, Agnar Aamodt, "Integrations with case-based reasoning", in The Knowledge Engineering Review, v.20 n.3, p.241-245, September 2005.

[36] J. Prentzas and I. Hatzilygeroudis,"Integrations of Rule-Based and Case-Based Reasoning",in Proc. International Conference on Computer, 2003.

[37] Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani Mahbod Tavallaee, "A Detailed Analysis of the KDD CUP 99 Data Set," in *Proceedings of the 2009 IEEE Symbosium on Computational Intellignece in Security and Defense Applications (CISDA 2009)*, Ottawa, ON, 2009, pp. 1-6.

[38] Luger, George F., Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Pearson Addison Wesley, Sixth Edition

[39] Edwina L. Rissland and David B. Skalak, "Combining Case-Based and Rule-Based Reasoning: A Heuristic Approach",pp534-530

[40] Aamodt, A, 1994, Explanation-driven case-based reasoning. In Wess, S, Althoff, K and Richter, M (eds) *Topics in Case-Based Reasoning*. Berlin: Springer, pp. 274–288.

[41] A. Aamoth and E. Plaza, "Case-Based Reasoning: Foundational Issues,Methodological Variations and System Approaches", Artificial Intelligence Communications, 7(1), 1994, pp. 39-59.

[42] Kapil Khandelwal "Hybrid Reasoning Model for Strengthening the problem solving capability of Expert Systems." *(IJACSA) International Journal of Advanced Computer Science and Applications*,Jaipur, India *2013.*

[43] D.M.W. POWERS, "EVALUATION: FROM PRECISION, RECALL AND F-MEASURE TO ROC,INFORMEDNESS, MARKEDNESS AND CORRELATION," *Journal of Machine Learning Technologies*, vol. II, no. 1, pp. 37-63, 2012.

[44] P.Pu and Chen, "A User-Centeric Evaluation Framework of Recommender Systems," in *Proceedings of the ACM RecSys 2010 workshop on User-centeric Evaluation of Recommender Systems and Their Interfaces (UCERSTI)*, Barcelona, Spain, 2010, pp. 157-164.

[45] *Rvine, CA 92697-3425, October 28, 1999,* http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Appendixes

Appendix 1

Attribute description with their data type

feature name	description	type
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of "wrong" fragments	continuous
urgent	number of urgent packets	continuous
hot	number of "hot" indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of "compromised" conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if "su root" command attempted; 0 otherwise	discrete
num_root	number of "root" accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous

is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	discrete
is_guest_login	1 if the login is a "guest"login; 0 otherwise	discrete

Appendix 2

Attributes relation and data declaration

@relation 'data set for CBNID'
@attribute duration numeric
@attribute protocol_type {tcp,udp,icmp}
@attribute service {http, smtp, finger, ntp_u, ecr_i, auth, domain_u, ftp, telnet, eco_i, private, ftp_data, pop_3, mtp, domain, ctf, name, systat, time, nnsp, http_443, echo, ssh, whois, remote_job, link, gopher, hostnames, csnet_ns, uucp_path, netbios_ns, vmnet, Z39_50, iso_tsap, pop_2, other, printer, efs, courier, uucp, discard, bgp, klogin, netbios_dgm, netbios_ssn, ldap, shell, daytime, supdup, nntp, imap4, login, kshell, rje, sql_net, exec, urh_i, urp_i, netstat, IRC, sunrpc, X11}
@attribute flag {SF, S1, RSTO, REJ, RSTOS0, S0, SH, RSTR, S2, S3}
@attribute src_bytes numeric
@attribute dst_bytes numeric
@attribute land numeric
@attribute wrong_fragment numeric
@attribute urgent numeric
@attribute hot numeric
@attribute num_failed_logins numeric
@attribute logged_in numeric
@attribute num_compromised numeric
@attribute root_shell numeric
@attribute su_attempted numeric
@attribute num_root numeric
@attribute num_file_creations numeric
@attribute num_shells numeric
@attribute num_access_files numeric
@attribute num_outbound_cmds numeric
@attribute is_host_login numeric
@attribute is_guest_login numeric
@attribute count numeric
@attribute srv_count numeric
@attribute serror_rate numeric
@attribute srv_serror_rate numeric
@attribute rerror_rate numeric
@attribute srv_rerror_rate numeric
@attribute same_srv_rate numeric
@attribute diff_srv_rate numeric

@attribute srv_diff_host_rate numeric
@attribute dst_host_count numeric
@attribute dst_host_srv_count numeric
@attribute dst_host_same_srv_rate numeric
@attribute dst_host_diff_srv_rate numeric
@attribute dst_host_same_src_port_rate numeric
@attribute dst_host_srv_diff_host_rate numeric
@attribute dst_host_serror_rate numeric

@attribute dst_host_srv_serror_rate numeric
@attribute dst_host_rerror_rate numeric
@attribute dst_host_srv_rerror_rate numeric
@attribute result {normal.,DOS,R2L,Probe,U2R}

@data

60,tcp,telnet,S3,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,1,1,1,0,1,0,1,1,0,0,R2L
0,tcp,telnet,RSTO,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,2,2,0.5,0.5,0.5,0.5,1,0,0,2,2,1,0,0.5,0,0.5,0.5,0.5,0
.5,R2L
0,tcp,telnet,RSTO,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,1,1,1,0,0,3,3,1,0,0.33,0,0.33,0.33,0.67,0.6
7,R2L
0,tcp,telnet,RSTO,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,0,4,4,1,0,0.25,0,0.25,0.25,0.75,0.7
5,R2L
0,tcp,telnet,RSTO,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,1,1,1,0,0,5,5,1,0,0.2,0,0.2,0.2,0.8,0.8,R2L
0,tcp,telnet,RSTO,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,1,1,1,0,0,6,6,1,0,0.17,0,0.17,0.17,0.83,0.8
3,R2L
0,tcp,http,SF,226,1484,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,10,10,0,0,0,0,1,0,0,255,255,1,0,0,0,0,0,0,0,normal
0,tcp,http,SF,231,1600,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,11,0,0,0,0,1,0,0,255,255,1,0,0,0,0,0,0,0,normal
0,tcp,http,SF,230,1651,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,12,12,0,0,0,0,1,0,0,255,255,1,0,0,0,0,0,0,0,normal
0,tcp,http,SF,231,1721,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,13,13,0,0,0,0,1,0,0,255,255,1,0,0,0,0,0,0,0,normal
0,tcp,http,SF,231,1713,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,14,14,0,0,0,0,1,0,0,255,255,1,0,0,0,0,0,0,0,normal
0,tcp,private,SH,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,118,1,0.01,0.92,0.93,0,0.93,1,0,0,Probe
0,tcp,private,SH,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,119,1,0.01,0.92,0.93,0,0.93,1,0,0,Probe
0,tcp,sunrpc,SH,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,120,1,0.01,0.92,0.93,0,0.93,1,0,0,Probe

Appendix 3

Domain expert evaluation form/query.

Dear Evaluator,

This evaluation form is adopted from [17] aiming at measuring to what extent does the system is useable and acceptable by end users in the areas of network administration. Therefore, you are kindly requested to evaluate the system by labeling (X) symbol on the space provided for the corresponding attributes values for each criteria of evaluation.

I would like to appreciate your collaboration in providing the information.

Note: - the values for all attributes in the table are related as excellent=5, Very good=4, Good=3, Fair=2, and Poor=1.

No	Criteria of evaluation	Poor	Fair	Good	Very Good	Excellent	Average
1	Simplicity to use and interact with the system						
2	Attractiveness of the system						
3	Efficiency in time						
4	The accuracy of the system in reaching a decision to identify the types of network attacks						
5	The ability of the system to make right conclusions and recommendation						
6	Importance of the KBS in the domain area						

Appendix 4

Prolog code for welcome page and rule representation

```
go:-

write(' WEL COME TO A COMBINED REASONING SYSTEM FOR KNOWLEDGNE BASED NETWORK INRUSION DETECTION'),nl,

write(' FOR NETWORK INTRUSION DETECTION & ADVISING SYSTEM'), nl,

write('**************************************************'), nl,nl,

write(' SYSTEM IS DESIGNED '),nl,

write(' and DEVELOPED by'),nl,

write(' MESERET ASSEFA'),nl,

write('**************************************************'),nl.

start :-

        write('write Attack: '),nl,

        read(Attack),nl,

        hypothesis(Attack,Class),

        write_list([Attack,' type is ',Class, '  !.']),nl.

start :-

        write('Sorry, I don"t seem to be able to'),nl,

        write('diagnose the type of Class of Attack. Please consult the CBR.'),nl.

symptom(Attack,root_shell >= 1) :-

        write_list(['does ',Attack,' has root_shell >= 1 (y/n) ?']),

        response(Reply),

        Reply='y'.

symptom(Attack,root_shell >= m1) :-

        write_list(['does ',Attack,' has root_shell >= m1 (y/n) ?']),

        response(Reply),

        Reply='y'.

symptom(Attack,duration >= 25) :-

        write_list(['does ',Attack,' has duration >= 25 (y/n) ?']),
```

```
        response(Reply),
        Reply='y'.
symptom(Attack,num_failed_logins >= 1) :-
        write_list(['does ',Attack,' has num_failed_logins >= 1 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,service = imap4) :-
        write_list(['does ',Attack,' has service = imap4 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,duration >= 25) :-
        write_list(['does ',Attack,' has duration >= 25 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,service = mimap4) :-
        write_list(['does ',Attack,' has service = mimap4 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,duration >= m25) :-
        write_list(['does ',Attack,' has duration >= m25 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,dst_host_count <= 11) :-
        write_list(['does ',Attack,' has dst_host_count <= 11 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,duration >= 12) :-
        write_list(['does ',Attack,' has duration >= 12 (y/n) ?']),
        response(Reply),
        Reply='y'.
```

```
symptom(Attack,protocol_type = sur) :-
        write_list(['does ',Attack,' has protocol_type = sur (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,dst_host_count <= x11) :-
        write_list(['does ',Attack,' has dst_host_count <= x11 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,dst_host_same_src_port_rate =< 0) :-
        write_list(['does ',Attack,' has dst_host_same_src_port_rate =< 0 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,protocol_type = xsur) :-
        write_list(['does ',Attack,' has protocol_type = xsur (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,service = private) :-
        write_list(['does ',Attack,' has service = private (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,src_bytes >= 100) :-
        write_list(['does ',Attack,' has src_bytes >= 100 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,duration = 0) :-
        write_list(['does ',Attack,' has duration >= 0 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,protocol_type = tcp) :-
        write_list(['does ',Attack,' has protocol_type = tcp (y/n) ?']),
```

```
        response(Reply),
        Reply='y'.
symptom(Attack,service = http) :-
        write_list(['does ',Attack,' has service = http (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,flag = sf) :-
        write_list(['does ',Attack,' has flag = sf (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,dst_host_count =< 11) :-
        write_list(['does ',Attack,' has dst_host_count =< 11 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,protocol_type = udp) :-
        write_list(['does ',Attack,' has protocol_type = udp (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,count >= 49) :-
        write_list(['does ',Attack,' has count >= 49 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,dst_bytes <= 0) :-
        write_list(['does ',Attack,' has dst_bytes <= 0 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,dst_host_count >= 75) :-
        write_list(['does ',Attack,' has dst_host_count >= 75 (y/n) ?']),
        response(Reply),
        Reply='y'.
```

```
symptom(Attack,count >= 3) :-
        write_list(['does ',Attack,' has count >= 3 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,src_bytes >= 21048) :-
        write_list(['does ',Attack,' has src_bytes >= 21048 (y/n) ?']),
        response(Reply),
        Reply='y'.
symptom(Attack,diff_srv_rate >= 0.37) :-
        write_list(['does ',Attack,' has diff_srv_rate >= 0.37 (y/n) ?']),
        response(Reply),
        Reply='y'.
hypothesis(Attack,u2r) :-
        symptom(Attack,root_shell >= 1),
        symptom(Attack,service = imap4),
        symptom(Attack,duration >= 25).
hypothesis(Attack,r2l) :-
        symptom(Attack,num_failed_logins >= 1),
        symptom(Attack,service = imap4),
        symptom(Attack,duration >= 12),
        symptom(Attack,dst_host_same_src_port_rate =< 0),
        symptom(Attack,dst_host_count =< 11).

hypothesis(Attack,xdos) :-
        symptom(Attack,dst_host_count <= 11),
        symptom(Attack,protocol_type = sur),
        symptom(Attack,duration >= 12).
hypothesis(Attack,normal_network_behaviour) :-
        symptom(Attack,duration = 0),
        symptom(Attack,protocol_type = tcp),
```

```
        symptom(Attack,flag = sf),
        symptom(Attack,service = http).
hypothesis(Attack,u2r) :-
        symptom(Attack,root_shell >= 1),
        symptom(Attack,service = mimap4),
        symptom(Attack,duration >= 25).
hypothesis(Attack,ndos) :-
        symptom(Attack,count >= 49),
        symptom(Attack,dst_bytes <= 0),
        symptom(Attack,dst_host_count >= 75),
        symptom(Attack,count >= 3),
        symptom(Attack,src_bytes >= 21048).
hypothesis(Attack,probe) :-
        symptom(Attack,diff_srv_rate >= 0.37),
        symptom(Attack,src_bytes >= 100),
        symptom(Attack,service = private),
        symptom(Attack,protocol_type = udp).
hypothesis(Attack,dos) :-
        symptom(Attack,count >= 49),
        symptom(Attack,dst_bytes <= 0),
        symptom(Attack,dst_host_count >= 75),
        symptom(Attack,count >= 3),
        symptom(Attack,src_bytes >= 21048).
write_list([]).
write_list([Term| Terms]) :-
        write(Term),
        write_list(Terms).
response(Reply) :-
        get_single_char(Code),
        put_code(Code), nl,
```

char_code(Reply, Code).

Appendix 5

Java code for integration of CBR and RBR, and Graphical interface

```
// A Java program developed to integrate RBR and CBR for network Network Intrusion Detection

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import java.awt.BorderLayout;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JRadioButton;

import javax.swing.JSeparator;

import javax.swing.ButtonGroup;

import javax.swing.JButton;


import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.Font;

import java.awt.Color;

import java.io.IOException;

import java.io.InputStream;



public class mmkk {


private JFrame frame;


private final ButtonGroup buttonGroup = new ButtonGroup();

private JLabel attack_result_label;

private JButton btnNewButton;
```

```
private final ButtonGroup buttonGroup_1 = new ButtonGroup();
private final ButtonGroup buttonGroup_2 = new ButtonGroup();
private final ButtonGroup buttonGroup_3 = new ButtonGroup();
private final ButtonGroup buttonGroup_4 = new ButtonGroup();
private final ButtonGroup buttonGroup_5 = new ButtonGroup();
private final ButtonGroup buttonGroup_6 = new ButtonGroup();
private final ButtonGroup buttonGroup_7 = new ButtonGroup();
private final ButtonGroup buttonGroup_8 = new ButtonGroup();
private final ButtonGroup buttonGroup_9 = new ButtonGroup();
private final ButtonGroup buttonGroup_10 = new ButtonGroup();
        private boolean root_shell;
        private boolean duration;
    private boolean num_failed_logins;
        private boolean service;
    private boolean dst_host_count;
private boolean dst_host_same_src_port_rate;
        private boolean duration2;
    private boolean diff_srv_rate;
        private boolean count;
        private boolean dst_bytes;

    private String attack_type=null;

        /**
        * Launch the application.
        */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
                public void run() {
                    try {
```

```
                    mmkk window = new mmkk();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public mmkk() {
        initialize();
    }
    public void clearGroup1(){
        buttonGroup_1.clearSelection();
        buttonGroup_2.clearSelection();
    /*
     * private final ButtonGroup buttonGroup = new ButtonGroup();
    private JLabel attack_result_label;
    private JButton btnNewButton;
    private final ButtonGroup buttonGroup_1 = new ButtonGroup();
    private final ButtonGroup buttonGroup_2 = new ButtonGroup();
    private final ButtonGroup buttonGroup_3 = new ButtonGroup();
    private final ButtonGroup buttonGroup_4 = new ButtonGroup();
    private final ButtonGroup buttonGroup_5 = new ButtonGroup();
    private final ButtonGroup buttonGroup_6 = new ButtonGroup();
    private final ButtonGroup buttonGroup_7 = new ButtonGroup();
    private final ButtonGroup buttonGroup_8 = new ButtonGroup();
```

```
private final ButtonGroup buttonGroup_9 = new ButtonGroup();
private final ButtonGroup buttonGroup_10 = new ButtonGroup();
* */
}
public void clearGroup2(){
buttonGroup_3.clearSelection();
buttonGroup_4.clearSelection();
buttonGroup_5.clearSelection();
buttonGroup_6.clearSelection();
buttonGroup_7.clearSelection();
}
public void clearGroup3(){
buttonGroup_8.clearSelection();
}
public void clearGroup4(){
buttonGroup_9.clearSelection();
buttonGroup_10.clearSelection();
}
/**
* Initialize the contents of the frame.
*/
/************************************************************Attack
* @throws IOException */
public String getAttackType() throws IOException{
if(root_shell && duration)
{
```

```
attack_type="U2R";
}
else if(num_failed_logins && service && dst_host_count && duration2 && dst_host_same_src_port_rate){
attack_type="R2L";
}

else if(diff_srv_rate){
attack_type="probe";
}
else if(count && dst_bytes){
attack_type="dos";
}
else {
attack_type=null;
JOptionPane.showMessageDialog(null, "The RBR can not identify the type of attack. Please consult the CBR.");
/*Process pro=Runtime.getRuntime().exec("C:\\Users\\messi\\Documents\\IDS\\src\\cbr\\IDS3.jar");
InputStream in=pro.getInputStream();
InputStream err=pro.getErrorStream();*/
}
String Stype=attack_type;
return(Stype);
}
private void initialize() {
frame = new JFrame();
frame.setBounds(100, 100, 574, 526);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

JPanel panel = new JPanel();
frame.getContentPane().add(panel, BorderLayout.CENTER);
panel.setLayout(null);
```

```
JLabel lblNewLabel = new JLabel("root_shell>=1");
lblNewLabel.setBounds(76, 73, 150, 16);
panel.add(lblNewLabel);

JLabel lblNewLabel_1 = new JLabel("duration>=25");
lblNewLabel_1.setBounds(76, 110, 150, 16);
panel.add(lblNewLabel_1);

JLabel lblNumfailedlogins = new JLabel("num_failed_logins >= 1");
lblNumfailedlogins.setBounds(76, 139, 150, 16);
panel.add(lblNumfailedlogins);

JLabel lblServiceimap = new JLabel("service=imap4");
lblServiceimap.setBounds(76, 168, 150, 16);
panel.add(lblServiceimap);

JLabel lblDsthostcount = new JLabel("dst_host_count=<11");
lblDsthostcount.setBounds(76, 197, 150, 16);
panel.add(lblDsthostcount);

JLabel lblDuration = new JLabel("duration2>=12");
lblDuration.setBounds(76, 226, 150, 16);
panel.add(lblDuration);

JLabel lblDsthostsamesrcportrate = new JLabel("dst_host_same_src_port_rate=<0");
lblDsthostsamesrcportrate.setBounds(65, 255, 209, 16);
panel.add(lblDsthostsamesrcportrate);

JLabel lblDiffsrvrate = new JLabel("diff_srv_rate>=0.37");
```

```
lblDiffsrvrate.setBounds(76, 296, 150, 16);
panel.add(lblDiffsrvrate);

JLabel lblCount = new JLabel("count >= 49");
lblCount.setBounds(76, 325, 150, 16);
panel.add(lblCount);

JLabel lblDstbytes = new JLabel("dst_bytes=< 0 ");
lblDstbytes.setBounds(76, 354, 150, 16);
panel.add(lblDstbytes);

JRadioButton rdbtnNewRadioButton = new JRadioButton("yes");
buttonGroup_1.add(rdbtnNewRadioButton);
rdbtnNewRadioButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        root_shell=true;
        clearGroup2();
        clearGroup3();
        clearGroup4();
        //btnNewButton
    //arg0.setSource(btnNewButton.getAction());
        }
    });
rdbtnNewRadioButton.setBounds(271, 73, 56, 25);
panel.add(rdbtnNewRadioButton);

JRadioButton rdbtnNo = new JRadioButton("no");
rdbtnNo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        root_shell=false;
```

```
                clearGroup2();
                clearGroup3();
                clearGroup4();
                }
            });
        buttonGroup_1.add(rdbtnNo);
        rdbtnNo.setBounds(351, 73, 71, 25);
        panel.add(rdbtnNo);

    JRadioButton radioButton_1 = new JRadioButton("no");
    radioButton_1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
                num_failed_logins=false;
                clearGroup3();
                clearGroup4();
                }
            });
        buttonGroup_3.add(radioButton_1);
        radioButton_1.setBounds(351, 143, 71, 25);
        panel.add(radioButton_1);

    JRadioButton radioButton_2 = new JRadioButton("yes");
    radioButton_2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
                num_failed_logins=true;
                clearGroup1();
                clearGroup3();
                clearGroup4();
            //buttonGroup_1.clearSelection();
                }
```

```
        });
        buttonGroup_3.add(radioButton_2);
        radioButton_2.setBounds(271, 143, 56, 25);
        panel.add(radioButton_2);

        JRadioButton radioButton_3 = new JRadioButton("no");
        radioButton_3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                service=false;
                clearGroup1();
                clearGroup3();
                clearGroup4();
                }
        });
        buttonGroup_4.add(radioButton_3);
        radioButton_3.setBounds(351, 172, 71, 25);
        panel.add(radioButton_3);

        JRadioButton radioButton_4 = new JRadioButton("yes");
        radioButton_4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                service=true;
                clearGroup1();
                clearGroup3();
                clearGroup4();
                }
        });
        buttonGroup_4.add(radioButton_4);
        radioButton_4.setBounds(271, 172, 56, 25);
        panel.add(radioButton_4);
```

```
JRadioButton radioButton_5 = new JRadioButton("no");
radioButton_5.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
                dst_host_count=false;
                clearGroup1();
                clearGroup3();
                clearGroup4();
                }
        });
buttonGroup_5.add(radioButton_5);
radioButton_5.setBounds(351, 201, 71, 25);
panel.add(radioButton_5);

JRadioButton radioButton_6 = new JRadioButton("yes");
radioButton_6.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
                dst_host_count=true;
                clearGroup1();
                clearGroup3();
                clearGroup4();
                }
        });
buttonGroup_5.add(radioButton_6);
radioButton_6.setBounds(271, 201, 56, 25);
panel.add(radioButton_6);

JRadioButton radioButton_7 = new JRadioButton("no");
radioButton_7.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
```

```
                duration2=false;
                clearGroup1();
                clearGroup3();
                clearGroup4();
                }
            });
        buttonGroup_6.add(radioButton_7);
    radioButton_7.setBounds(351, 230, 71, 25);
        panel.add(radioButton_7);

JRadioButton radioButton_8 = new JRadioButton("yes");
radioButton_8.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
                duration2=true;
                clearGroup1();
                clearGroup3();
                clearGroup4();
                }
            });
        buttonGroup_6.add(radioButton_8);
    radioButton_8.setBounds(271, 230, 56, 25);
        panel.add(radioButton_8);

JRadioButton radioButton_9 = new JRadioButton("no");
radioButton_9.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
            dst_host_same_src_port_rate=false;
                clearGroup1();
                clearGroup3();
                clearGroup4();
```

```
                }
            });
        buttonGroup_7.add(radioButton_9);
        radioButton_9.setBounds(351, 251, 71, 25);
        panel.add(radioButton_9);

    JRadioButton radioButton_10 = new JRadioButton("yes");
    radioButton_10.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
                dst_host_same_src_port_rate=true;
                clearGroup1();
                clearGroup3();
                clearGroup4();
                }
            });
        buttonGroup_7.add(radioButton_10);
        radioButton_10.setBounds(271, 251, 56, 25);
        panel.add(radioButton_10);

     JRadioButton radioButton_11 = new JRadioButton("no");
    radioButton_11.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
                diff_srv_rate=false;
                clearGroup1();
                clearGroup2();
                clearGroup4();
                }
            });
        buttonGroup_8.add(radioButton_11);
        radioButton_11.setBounds(351, 296, 71, 25);
```

```
panel.add(radioButton_11);

JRadioButton radioButton_12 = new JRadioButton("yes");
radioButton_12.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        diff_srv_rate=true;
        clearGroup1();
        clearGroup2();
        clearGroup4();
        }
    });
buttonGroup_8.add(radioButton_12);
radioButton_12.setBounds(271, 296, 56, 25);
panel.add(radioButton_12);

JRadioButton radioButton_13 = new JRadioButton("no");
radioButton_13.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        count=false;
        clearGroup1();
        clearGroup2();
        clearGroup3();
        }
    });
buttonGroup_9.add(radioButton_13);
radioButton_13.setBounds(351, 329, 71, 25);
panel.add(radioButton_13);

JRadioButton radioButton_14 = new JRadioButton("yes");
radioButton_14.addActionListener(new ActionListener() {
```

```
public void actionPerformed(ActionEvent arg0) {
count=true;
clearGroup1();
clearGroup3();
clearGroup2();

}
});
buttonGroup_9.add(radioButton_14);
radioButton_14.setBounds(271, 329, 56, 25);
panel.add(radioButton_14);

JRadioButton radioButton_15 = new JRadioButton("no");
radioButton_15.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
dst_bytes=false;
clearGroup1();
clearGroup2();
clearGroup3();
}
});
buttonGroup_10.add(radioButton_15);
radioButton_15.setBounds(351, 358, 71, 25);
panel.add(radioButton_15);

JRadioButton radioButton_16 = new JRadioButton("yes");
radioButton_16.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
dst_bytes=true;
clearGroup1();
```

```
                clearGroup2();
                clearGroup3();
                }
            });
        buttonGroup_10.add(radioButton_16);
        radioButton_16.setBounds(271, 358, 56, 25);
        panel.add(radioButton_16);

        JRadioButton radioButton = new JRadioButton("no");
        radioButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                duration=false;
                clearGroup4();
                clearGroup2();
                clearGroup3();
                }
            });
        buttonGroup_2.add(radioButton);
        radioButton.setBounds(351, 105, 71, 25);
        panel.add(radioButton);

        JRadioButton radioButton_17 = new JRadioButton("yes");
        radioButton_17.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                duration=true;
                //clearGroup1();
                clearGroup2();
                }
            });
        buttonGroup_2.add(radioButton_17);
```

```
radioButton_17.setBounds(271, 105, 56, 25);

panel.add(radioButton_17);


JSeparator separator = new JSeparator();

separator.setBounds(385, 133, -360, -3);

panel.add(separator);


JLabel lblNewLabel_2 = new JLabel("WELL COME TO A HYBRIDE NETWORK INTRUSION
DETECTION SYSTEM");

lblNewLabel_2.setFont(new Font("Tahoma", Font.ITALIC, 15));

lblNewLabel_2.setForeground(Color.BLUE);

lblNewLabel_2.setBounds(12, 25, 505, 35);

panel.add(lblNewLabel_2);


JSeparator separator_1 = new JSeparator();

separator_1.setForeground(Color.BLUE);

separator_1.setBounds(76, 139, 328, 2);

panel.add(separator_1);


JSeparator separator_2 = new JSeparator();

separator_2.setForeground(Color.BLUE);

separator_2.setBounds(76, 281, 328, 2);

panel.add(separator_2);


JSeparator separator_3 = new JSeparator();

separator_3.setForeground(Color.BLUE);

separator_3.setBounds(76, 323, 328, 2);

panel.add(separator_3);


JSeparator separator_4 = new JSeparator();

separator_4.setForeground(Color.BLUE);
```

```
separator_4.setBounds(76, 392, 318, 2);
panel.add(separator_4);

btnNewButton = new JButton("Check");
btnNewButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
//JOptionPane.showMessageDialog(null, "test");
try {
attack_result_label.setText(getAttackType());
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
});
btnNewButton.setBounds(65, 441, 97, 25);
panel.add(btnNewButton);

JButton btnExit = new JButton("Exit");
btnExit.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
System.exit(0);
}
});
btnExit.setBounds(191, 441, 97, 25);
panel.add(btnExit);

JLabel lblAttackType = new JLabel("Attack Type:");
lblAttackType.setBounds(76, 407, 131, 16);
panel.add(lblAttackType);
```

```
attack_result_label = new JLabel(" ");
attack_result_label.setForeground(Color.RED);
attack_result_label.setFont(new Font("Tahoma", Font.BOLD, 15));
attack_result_label.setBounds(191, 407, 187, 21);
panel.add(attack_result_label);

JButton btnNewButton_1 = new JButton("Reset");
btnNewButton_1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {

buttonGroup_1.clearSelection();
buttonGroup_2.clearSelection();
buttonGroup_3.clearSelection();
buttonGroup_4.clearSelection();
buttonGroup_5.clearSelection();
buttonGroup_6.clearSelection();
buttonGroup_7.clearSelection();
buttonGroup_8.clearSelection();
buttonGroup_9.clearSelection();
buttonGroup_10.clearSelection();
//attack_result_label.setText("");

}
});
btnNewButton_1.setBounds(317, 441, 90, 25);
panel.add(btnNewButton_1);
}
public JLabel getAttack_result_label() {
return attack_result_label;
```

```
    }
public JButton getBtnNewButton() {
        return btnNewButton;
    }
}
```