ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL SCIENCE
SCHOOL OF INFORMATION SCIENCE

Towards Integrating Data Mining with Knowledge Based System: The Case of Network Intrusion detection

By

Abdulkerim Mohammed

June , 2013

ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL SCIENCE
SCHOOL OF INFORMATION SCIENCE

Towards Integrating Data Mining with Knowledge Based System: The Case of Network Intrusion detection

A Thesis Submitted to the College of Natural Science of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Science

By

Abdulkerim Mohammed

June, 2013

ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL SCIENCE
SCHOOL OF INFORMATION SCIENCE

Towards Integrating Data Mining with Knowledge Based System: The Case of Network Intrusion detection

A Thesis Submitted to the College of Natural Science of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Science

By

Abdulkerim Mohammed

Names and Signature of Members of the Examining Board

_____	_____
Chair person, Examining Board	Signature
_____	_____
Advisor	Signature
_____	_____
Examiner	Signature

DEDICATION

I would like to dedicate this study for my beloved mother Hawa Hassen and my beloved father Mohammed Yibre.

ACKNOWLEDGEMENT

First and foremost I would like to thank the almighty ALLAH for giving me the courage and strength to finish this study.

Next my best gratitude goes to my advisor Million Meshesha (PhD) for his, unreserved comments, encouragement, guidance and motivation he gave me to accomplish this thesis. His support has not been limited to advising this research but also giving support during my study in this program.

I sincerely thank my whole family, without their support and encouragement I would have not been here today. They have always been with me supporting, helping and cherishing in my journey to be a better man.

In addition, I would like to take this opportunity to thank Tigabu Dagne, Aminu Mohammed and Mifta Hassen who consulted me on different issues regarding the thesis and reviewed my thesis write-up for better quality.

Moreover, Se'ad Ahmedin (seadu) and my class mates Mengistu Belete and Getachew Adefa I am grateful for being with me during my difficult moments and giving me hope and encouragement.

Last but not least I would like to thank Hassen Ali and Mohammed Endris for their unforgettable help.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS AND ACRONYMS

ARFF:-Attribute Relation File Format

CBR:-Case Based Reasoning

CSV:-Comma Separated Values

DOS:- Denial of Service

IDS:-Intrusion Detection System

IPS:- Intrusion Prevention System

KB:-Knowledge base

KBS: - Knowledge Based System

KDD:-Knowledge Discovery in Database

KE: - Knowledge Engineer

MIT:- Massachusetts Institute of Technology

NIDS:-Network Intrusion Detection System

PROLOG:-Programming in Logic

R2L:- Remote to Local

RIDA-KBS:- Rule based Intrusion Detection and Advising Knowledge Based System

SMOTE:- Synthetic Minority Oversampling TEchnique

TCP:- Transmission control protocol

U2R:-User to Root

WEKA-Waikato Environment for Knowledge Analysis

ABSTRACT

Network intrusion is one of cyber attacks which bypass the security mechanisms of computer systems. Protection of such types of attacks ensures organizations from unplanned shut down of networks which have otherwise bad consequent on the organization. Intrusion detection systems respond to malicious activities. Misuse detection searches for patterns or user behaviors that match known intrusion scenarios, which are stored as signatures. Anomaly detection keeps normal behavior of network and it label as an attack behaviors which are beyond this. 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.

In this study, rule based intrusion detection and advising knowledge based system is proposed. The 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.

Finally, the performance of the system is evaluated by preparing test cases. Twenty test cases are prepared for system performance test and provided to domain experts. For user acceptance test users are trained and evaluated the system. Generally the system has scored 80.5% overall performance which is a promising result. But further exploration has to be done to refine the knowledge base and boost the advantages of integrating data mining induced knowledge with knowledge based system.

Keywords:- Intrusion detection, data mining, knowledge based system, Integrator

CHAPTER ONE

INTRODUCTION

1.1 Background

Due to the wide spread application of computers and the exponential growth of computer networks such as the Internet, great changes are taking place in the area of information supply and demand. Today the world of computing is faced with the ever-increasing likelihood of unplanned downtime due to various attacks and security breaches. Those companies around the globe which are maintaining the continuity of their services and retaining their computing power enjoy a significant competitive advantage [**1**]. Network downtime results in financial losses and harms the credibility of organizations. Minimizing or eliminating the unplanned and unexpected downtime of networks can be achieved by identifying, prioritizing and defending against misuse, attacks and vulnerabilities

Computer Security is the ability to protect a computer system and its resources with respect to confidentiality, integrity, and availability. Various protocols and firewalls are in existence to protect from computer threats. As defined in Nagaraju et al. [**2**], intrusion is a type of cyber attack that attempts to bypass the security mechanism of a computer system. Such an attack can be done by an outsider who attempts to access the system, or an insider who attempts to gain and misuse non-authorized privileges [**2**].

Ali et al. [**1**] defined intrusion detection as, the process of identifying and responding to malicious activities targeted at computing and network resources. Citing Bace(2000) Fashoto et al. [**3**] also defined intrusion detection as the process of monitoring and analyzing the events occurring in a computer system in order to detect signs of security problems. Intrusion detection systems are basic components in network security infrastructure. They examine system or network activities to find possible intrusions or attacks and trigger security alerts for the malicious actions [**1**].

Intrusion Detection Systems are classified based on their functionality as misuse detectors and anomaly detectors [**4**]. Misuse detection system uses well defined patterns of attack which are matched against user behavior to detect intrusions. Misuse detection is simpler than anomaly detection as it uses rule based or signature comparison methods. Anomaly detection requires storage of normal usage behavior and operates upon audit data generated by the operating system [**4**].

Data mining has made tremendous progress in the last ten years. According to Mihaela [**5**], Data mining (DM) is a subfield of Machine Learning that enables finding interesting knowledge (patterns, models and relationships) in very large databases. It is the most essential part of the knowledge-discovery process, which combines databases, statistics, artificial intelligence and machine learning techniques.

Han [**6**] also defined data mining as the process of discovering interesting knowledge from large amounts of data stored either in databases, data ware houses or other information repositories. Therefore, according to Han [**6**], data mining uses a number of data stores on which mining can be performed including relational databases, data warehouses, transactional databases, advanced database systems (such as object-oriented databases ,spatial databases, text databases etc), flat files and the world wide web. However, a large gap remains between the results a data mining system can provide and taking actions based on them [**7**].This implies that there is a gap in extracting knowledge employing data mining and using it for action.

A Knowledge Base System is a computer system which represents knowledge about a specific problem domain and can be used to apply this knowledge to solve the problems from the problem domain [**8**].

Mihaela [**5**], citing Milen et al.(1997), stated that in order to make knowledge extraction as much correct as possible different techniques could be applied . Among these techniques, data mining or knowledge discovery techniques became the most used in the

recent years [**5**]. In addition, as mentioned in [**8**] the cornerstone of an effective Knowledge-Based System is data mining. Data mining uses machine learning and statistical analysis to develop better business decisions than could be made using conventional methods. Data mining improves decision making by giving insight into what is happening today and by helping predict what will happen tomorrow. Many data mining tools on the market today can help to build powerful Knowledge-Based Systems [**8**].

The application of a data mining approach in knowledge base development involves a set of techniques for searching through data sets, looking for hidden correlations and trends which are inaccessible using conventional data analysis techniques. The basic techniques for data mining include: decision-tree induction, rule induction, instance-based learning, artificial neural networks, Bayesian learning, support vector machines, ensemble techniques, clustering, and association rules [**5**] .

This study is aimed at designing prototype rule based intrusion detection and advising knowledge based system by using an automatically constructed knowledge base based on knowledge acquired from data mining models and providing advice for network administrators.

1.2 Statement of the problem

As explicitly mentioned by Ali et al. [**1**] , various network attacks and security breaches are causing an ever-increasing likelihood of unplanned down time for the world of computing. In this environment of uncertainty with increasing number of hackers and malicious threats, those companies around the globe which are the best at maintaining the continuity of their services and retaining their computing power, enjoy a significant competitive advantage. Minimizing or eliminating the unplanned downtime of the network establishes the continuity of the computing services and this in return results in credibility of the organization and minimizing financial losses caused by network downtime.

There is problem in Network Intrusion Detection Systems because they are tuned specifically to detect known service level network attacks [**9**]. Moreover, as stated in [**1**] , [**10**], [**11**], [**12**] current Intrusion Detection System has limitations of generating false alarms and data mining can help improve intrusion detection by addressing these problems.

Data mining tools and techniques are proven to work well in data voluminous environment by extracting hidden knowledge in the data. Data mining uses historical data as a baseline for comparison with current activity. It also serves to aid network administrators, security officers, and analysts in the performance of their duties by allowing them to ask questions that would not have occurred to them a priori. In addition, [**13**] stated that data mining has tools for converting data into patterns with an underlying assumption that the patterns are created solely from the data, and thus are expressed in terms of attributes and relations appearing in the data

A number of research works have been done in the area of network intrusion detection by applying different techniques and algorithms of data mining. Adamu [**14**] tried to study a machine learning intrusion detection System that investigate the application of cost sensitive learning using data mining approach to network intrusion detection. Adamu [**14**]proposed learning approach for network intrusion detection performed using cost sensitive learning techniques by testing decision tree algorithm on labeled records. Another researcher, Zewdie [**15**] attempted to develop a model for network intrusion detection using information gain value for feature selection. Tigabu [**16**] constructed an Intrusion detection system which detect an attack and inform administrators such that they have to take proper actions.

However, Domingos [**7**] stated that, there is a gap between the results a data mining system can provide and taking action based on them. Both researchers [**14**], [**15**] and [**16**]

developed predictive model for network intrusion detection and classify the data set as normal or an intrusion. It is possible to say that their work lack to deploy the knowledge extracted for assisting those who have a concern on it.

From the recommendation of [**16**] and the criticism on data mining results by [**7**], it is understood that integrating data mining with knowledge base system is essential to deploy the knowledge extracted from data mining models. Tigabu [**16**] recommended designing a knowledge base system which will add adaptability and extensibility features of the Intrusion Detection System and connect to data mining model as one of the future research directions. In addition, commercial network intrusion detection system mostly generates alarms when they get attacks according to their knowledge base and the action to be taken is left to the network administrator [**9**]. At this point, developing knowledge base system is paramount to identify different types of attack and give advice accordingly to help the administrators which action to take. The integration of data mining induced knowledge with knowledge based system allows utilizing interesting and previously unseen knowledge extracted from data mining models for knowledge base system. This again lessens the problems of commercial intrusion detection systems from merely notifying while detecting attacks by adding values like providing advice and information about the detected network attacks.

During development of knowledge based system, knowledge must be acquired about the problem to be solved. Knowledge can be acquired from different sources such as making interview with domain experts, document analysis, observation and others [**17**]. Since tacit knowledge is personal and the knowledge expert may not tell all the knowledge he knows during interview, there is hidden knowledge about the problem. To alleviate this problem, automatic knowledge acquisition is proposed [**18**]. Data mining has been proposed for extracting hidden and previously unknown knowledge from datasets by different researchers [**10**]. Therefore, for this study knowledge for the KBS is acquired using data mining techniques.

As to the researcher's knowledge there are no research works done for integrating data mining models with knowledge based system for detecting network attacks and providing advice to user accordingly. Consequently, this study is aimed at developing knowledge based system that automatically acquire knowledge from network intrusion data set, taken from KDDcup'99, by applying data mining techniques. Apart from this, the number and types of network attacks are increasing from time to time. Therefore, the knowledge based system should have learning capability so as to update the knowledge base as new types of network attacks are identified and accordingly providing advice about types of intrusions for network administrators.

In this regard, in an attempt to integrate data mining with knowledge based system, this study explores and finds answers to the following research questions:-

- Is it possible to use rules or attack signatures resulted from production rules in data mining to construct rule-based knowledge based system and provide advices for user?
- How it is possible to update the knowledge based system from knowledge extracted using data mining techniques?
- What are the challenges for integrating data mining with knowledge based system?

1.3 Objective of the study

1.3.1 General objective

The general objective of this study is to construct a prototype knowledge base system which can update its knowledge base using the hidden knowledge extracted from intrusion dataset by using data mining techniques.

1.3.2 Specific objectives

Towards achieving the general objective, this study specifically attempts the following:

- To review literatures pertinent to the study.
- To identify suitable data mining techniques and create predictive model based on intrusion data set

- To acquire knowledge from the predictive model for knowledge base construction.
- To build integrator aiming an automatically building knowledge base from the predictive model.
- To update knowledge base based on the new knowledge obtained from data mining
- To evaluate the performance and user acceptance levels of the knowledge based system.

1.4 Scope and limitation of the study

The aim of this study is integrating data mining with knowledge based system such that the system is enabled to automatically acquire hidden knowledge using intrusion data set collected from KDDcup'99. The integration yields rule based knowledge based system for detecting attacks and providing advice.

The knowledge acquisition for the knowledge based system is effected automatically employing data mining techniques rather than undertaking interview with experts. Data mining results, JRip Rules, are mapped or integrated to knowledge base by using integrator application. The integrator directly creates knowledge after mining rules from the data set. It has no interface for selecting evaluated rules. Whenever the integrator re-runs following a change in the size of the data set, prevailing rules are overwritten by the new coming rules. Which means it does not keep old rules.

In addition, the study did not include real life network data due to shortage of time to process such type of data in making them ready for mining task.

Moreover, there are 21 types of network attacks from the KDDcup'99 data set which are grouped into four classes namely Probe, DOS, R2L and U2R. The system is designed to detect based on the four classes rather than considering all 21 attack types.

The advice after detecting and identifying a network attack is targeted mainly for beginner network administrators.

1.5 Significance of the study

Sterry [**9**] stated that, detecting intrusions allows administrators to identify areas where their defenses need improvement, such as by identifying a previously unknown vulnerability, a system that was not properly patched, or a user that needs further education against social engineering attacks .

The proposed system provides an advice about detected intrusion to network administrators regarding which action to take in accordance to the detected network attack.

In addition, this study will motivate future researchers to work on integrating data mining models and knowledge based system in other fields of studies especially in areas where there is shortage of domain experts to acquire knowledge but there exist tremendous data such that knowledge can be acquired automatically from the data.

1.6 Methodology

The following methodologies have been used in the course of this study.

1.6.1 Literature review

Different research works previously done in the area of intrusion detection, data mining, application of data mining for intrusion detection, and knowledge base system are referred to gain ample knowledge in the areas. In addition, research works which attempted to integrate data mining with knowledge base system are also referred.

1.6.2 Knowledge Discovery Process

Knowledge Discovery in Database (KDD) model is followed for the data mining task. Knowledge Discovery Databases is the process of extracting and refining useful knowledge from large databases [**5**]. KDD has been used by different researchers to discover knowledge from large collection of records. It has seven steps in which data mining is included as the fifth step. The steps involved in knowledge discovery are depicted in the Figure 1.1

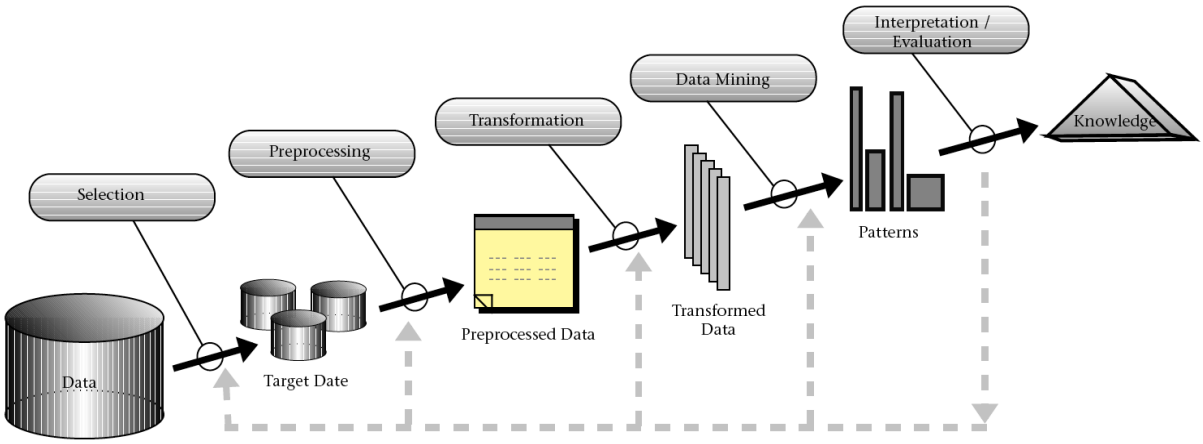

Figure 1-1 an overview of the steps that compose the KDD process

KDD is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. KDD focuses on the overall process of knowledge discovery from data, including how the data is stored and accessed, how algorithms can be scaled to massive data sets, run efficiently, how results can be interpreted and visualized, and how the overall man-machine interaction can usefully be modeled and supported.

As illustrated in figure 1-1 data mining is a step in the KDD process. The data-mining component of KDD currently relies heavily on known techniques from machine learning, pattern recognition, and statistics to find patterns from data in the data-mining step of the KDD process. The data-mining Component of the KDD process is concerned with the algorithmic means by which patterns are extracted and enumerated from data [**19**].

The KDD process involves using the database along with any required selection, preprocessing, sub sampling, and transformations of it; applying data-mining methods (algorithms) to enumerate patterns from it; and evaluating the products of data mining to identify the subset of the enumerated patterns deemed knowledge. The KDD process is

interactive and iterative, involving numerous steps with many decisions made by the user. The KDD process as noted in [**10**] consists of seven steps as mentioned below:

- **Data cleaning** :- to remove noise and inconsistent data
- **Data integration** :- where multiple data sources may be combined
- **Data selection** :- where data relevant to the analysis task are retrieved from the database
- **Data transformation**: - where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.
- **Data mining** :- an essential process where intelligent methods are applied in order to extract data patterns
- **Pattern evaluation**:- to identify the truly interesting patterns representing knowledge based on some interestingness measures
- **Knowledge presentation**: - where visualization and knowledge representation techniques are used to present the mined knowledge to the user.

In an attempt to acquire knowledge using KDD, the following steps are included in this study.

- Data selection: - Intrusion data is collected from KDDcup'99 data set from Lincoln lab available at ACM Knowledge Discovery web site [**20**].
- Data preprocessing: - the data set is preprocessed before the actual mining task is undertaken on the data. Considering the time it takes to process data sets and coming up patterns, a sample size of 35,778 instances are taken for this study. During the preprocessing task of instances are clustered into five classes namely; normal, Probe, DOS, R2L and U2R.
- Data mining: - so as to build predictive model from the sampled data set, Decision tree and production rule classifiers are used are purposively selected to generate rules.
- Evaluation and interpretation:- the predictive model, which are rules about intrusions, are evaluated by domain experts. In addition, the performance of

classifier algorithms is compared and the one which performed best is selected as prime choice for the knowledge acquisition step.

1.6.3 Knowledge Representation

Rule based knowledge representation approach is used to represent knowledge. A rule is a conditional statement that links given conditions to action or outcomes [**21**]. Rules are constructed in the form of *if-then* format. These *if-then* rules statements are used to formulate the conditional statements that constitute knowledge base. Rule based representation is highly expressive, is easy to interpret and easy to generate. In addition, it also classifies new instances rapidly [**22**]. Hence, rules are used to represent knowledge for the knowledge based system.

1.6.4 System development methodology

Prototyping approach is followed to develop the knowledge based system. Prototyping allows participating users and domain experts for evaluating systems performance and efficiency.

1.6.5 Implementation tool

In order to mine hidden knowledge from the pre-processed dataset and compare the performance of classifiers, WEKA 6.8.1 is used. WEKA 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. It contains tools for data preprocessing, clustering, regression, classification, association rules and visualization. WEKA is written in the Java language and contains a GUI for interacting with data files and producing visual results. It also has a general Application Page Interface (API), so WEKA can be embedded like any other library in applications [**11**].

In addition, in order to develop an application which maps the knowledge acquired from the data mining classifiers with knowledge based system Java NetBeans IDE 7.3 with JDK 6 is employed. NetBeans offers easy and efficient project management, has best support for latest java technologies, and can be installed on all operating systems supporting java.

To represent rules in knowledge base and constructing the Rule based Intrusion Detection and Advising Knowledge based system PROLOG is used. PROLOG is used because the researcher is more familiar than other AI programming languages used to develop knowledge based system. SWI-PROLOG editor is used to represent rules.

A prolog program consists of a set of facts accompanied by a set of conditions that the solution must satisfy; the computer can figure out for itself how to deduce the solution from the given facts. This is called logic programming [**23**]. Prolog is based on formal logic and solves problems by applying techniques originally developed to prove theorems in logic. It is a versatile language.

Prolog was invented by Alain Colmerauer and his colleagues at the University of Aix-Marseille, in Marseilles, France, in 1972. The name stands for *programming in logic*. These days it is used in artificial intelligent applications especially in automated reasoning systems. Prolog has an automated reasoning procedure called in inference engine which is built into it. As a result programs that perform logical reasoning are much easier to write in Prolog.

Prolog derives its power from a procedural interpretation of logic- that is, it represents knowledge in terms of procedure definitions, and reasoning becomes a simple process of calling the right procedures [**23**]. To see how it works consider the following examples.

i. For any X, X is an attack if X is in probe
ii. Ipsweep is in probe.

A collection of information containing the above two is called knowledge base. Item i is called rule since it enables to infer one piece of information from another, and item ii is called a fact because it does not depend on any other information. Rules contain an "if" but facts don't. Facts and rules are the types of clauses in prolog. A fact need to be true statement about the real world and also it is sometimes called ground clause because it is the basis from which other information is inferred.

Prolog has its own notation for representing knowledge the above sample knowledge base can be represented as:

```
is_attack(X) :- is_probe(X).
is_probe(Ipsweep).
```

To prove that Ipsweep is an attack or not first the procedure '`is_attack(X)`' is called which in return called the procedure '`is_probe(X)` 'which returns the answer "yes".

1. 6.6 Evaluation methods

The set of discovered rules has to be verified for accuracy (the rules portray the dataset), consistency (no redundant or contradictory rules) and usefulness (rules showing the decision making process) for knowledge base being developed [**18**].

The accuracy of the models developed using data mining techniques are evaluated based on detection accuracy of classifiers, Precision, Recall, F-measure and True Positive rate.

The KBS is evaluated using system performance testing by preparing test cases. Moreover, it is also tested by users to ensure user acceptance and check the extent to which the system meets user requirements.

1.7 Organization of the study

This study comprises of six chapters. Chapter one discusses an introduction about the study giving highlight about the area to which the study is focused. It provides highlight about network intrusion, data mining and knowledge based system. Problem statement, objective, significance, scope, and methodology used in the study is also included.

Chapter two is basically dedicated for literature review. In this chapter, a detailed discussion about data mining and its tasks pertinent for this study are included. The concept of intrusion detection, type of detection mechanisms and the advantages of detection are discussed. Moreover, since the concern of the study is an integration of data mining and knowledge base system for intrusion detection, literatures focused on employing data mining model for construction of knowledge based system are discussed. Discussion about knowledge base system including types of reasoning for knowledge

base system, how knowledge is acquired and represented so as to develop knowledge base system is also covered. In addition, related works in the area of intrusion detection by using data mining techniques and knowledge base are also included in this chapter.

Chapter three presents the knowledge acquisition process. The focus here is on automatic knowledge acquisition techniques through data mining. Knowledge discovery steps such as data set preparation, preprocessing and predictive model creation and experimentation are also discussed in the chapter. The results of WEKA classifier algorithms are analyzed, interpreted, evaluated and compared with each other.

Chapter four discusses integration of network attack signatures induced from data mining techniques with knowledge base system. The automatic construction of knowledge base based on the best performing classifier's model is thoroughly discussed.

Chapter five is all about Rule based Intrusion Detection and Advising Knowledge Based system. The basic functionality of the knowledge base, performance evaluation and user acceptance testing are discussed under the chapter.

Finally, Chapter six is dedicated for conclusion and recommendation. In this chapter based on the result obtained from the study the researcher's concluding remarks and recommendation for future works are presented.

CHAPTER TWO

Literature Review

Internet is a global public network. With the growth of the Internet and its potential, there has been subsequent change in business model of organizations across the world. More and more people are getting connected to the Internet every day to take advantage of the new business model popularly known as e-Business. Internetwork connectivity has therefore become very critical aspect of today's e-business. There are sides of business on the Internet. On one side, the Internet brings in tremendous potential to business in terms of reaching the end users. At the same time it also brings in lot of risk to the business. There are both harmless and harmful users on the Internet. While an organization makes its information system available to harmless Internet users, at the same time the information is available to the malicious users as well [**24**]. As mentioned in Ali et al. [**1**], various network attacks and security breaches are causing an ever-increasing likelihood of unplanned down time for the world of business computing. In this environment of uncertainty which is full of hackers and malicious threats, those companies around the globe which are the best at maintaining the continuity of their services and retaining their computing power, enjoy a significant competitive advantage. Minimizing or eliminating the unplanned downtime of the system establishes the continuity of the computing services and this in return results credibility of the organization and minimizing financial losses caused by network downtime.

2.1 Network Intrusion Detection

Security is a big issue for all networks in today's enterprise environment. Hackers and intruders have made many successful attempts to bring down high-profile company networks and web services [**1**] [**25**]. Minimizing unexpected and unplanned network downtime can be done by identifying, prioritizing and defending against misuse, attacks and vulnerabilities. Greensmith and Aickelin noted that [**26**] , during the design phase of a distributed system, security policies are developed which account for the measures

taken to ensure both the *confidentiality* and *integrity* of the system, when it is necessary. Confidentiality in this context refers to access constraints on users, and the existence of ways to protect the data. The integrity refers to the correct running of the system and the data contained on the system. Additionally, the usability of the system must be preserved, which is tied in with preserving the integrity of the system so that it is still functioning at the use level.

As stated by Greensmith and Aickelin [**26**] noted, there are several ways in which a system can be compromised.

- *Interception* can occur when an unauthorized user gains access to a service or to a resource, such as the illegal copying of data after breaking into a restricted file system.
- *Interruption* can occur when files are corrupted or erased, occurring as the result of denial of service attacks or from the action of a computer virus.
- *Modification* involves an unauthorized user or program making changes to data or system configuration, and can also include the modification of transmitted data, leading to a breakdown of trust between parties.
- *Fabrication* is where data or activities are generated which would not normally occur. An example of this would be the addition of information to a password file in order to compromise a system. To prevent such events from taking place within a system, a security policy must be put into place, and the necessary measures taken. Such measures can include the encryption of data, correct authentication and authorization of users with respect to data access and command execution, and the conscientious audit of log files monitoring system activity.

Many tools and techniques exist with the purpose of ensuring the confidentiality and integrity of a system. Firewalls, intrusion detection systems and anti-virus scanners are among the tools. The use and deployment of these tools depends upon where in the system they are placed, and indeed, the architecture of the system itself.

Firewall systems are commonly implemented throughout computer networks. They act as a measure of control, enforcing the relevant components of the security policy [**1**]. A firewall can be a number of different components such as router or a collection of host machines. However, the basic function of firewall is to protect the integrity of the network which is firewall controlled. However, as stated in [**27**], Firewalls and other simple boundary devices lack some degree of intelligence when it comes to observing, recognizing, and identifying attack signatures that may be present in the traffic they monitor and the log files they collect.

Signature is a pattern that is found in a data packet. It is used to detect one or more multiple types of attacks. For example the presence of "scripts/iisadmin" in a packet going to your web server may indicate an intruder activity.

Intrusion is an attempt to break or misuse a system. An intrusion normally exploits specific vulnerability and must be detected as quickly as possible [**1**] with the help of Intrusion detection system. An ***Intrusion detection system*** (IDS) deals with detecting and responding to malicious network traffic and computer misuse. Intrusion detection is the process of identifying and (possibly) responding to malicious activities targeted at computing and network resources [**1**]. Any hardware or software automation that monitors detects or responds to events occurring in a network or on a host computer is considered relevant to the intrusion detection approach. An intrusion-detection system acquires information to perform a diagnosis on the security status of the latter. The goal is to discover breaches of security, attempted breaches, or open vulnerabilities that could lead to potential breaches [**28**]. Rafeeq [**25**] also defined intrusion detection system as IDS is software, hardware or combination of both used to detect intruder activity.

2.1.1 Types of Network Attacks

Shanmugavadivu and Nagarajan [**12**] stated that, network attacks can be categorized as Probe, Denial of Services (DOS), User to Root (U2R), and Remote to Login (R2L).

- A **Denial of Service** attack is an attack where the attacker constructs some computing or memory resource fully occupied or unavailable to manage legitimate requirements, or reject legitimate users right to use a machine.
- **User to Root Attacks:** User to Root exploits are a category of exploits where the attacker initiate by accessing a normal user account on the system (possibly achieved by tracking down the passwords, a dictionary attack, or social engineering) and take advantage of some susceptibility to achieve root access to the system.
- **Remote to User Attacks:** A Remote to User attack takes place when an attacker who has the capability to send packets to a machine over a network but does not have an account on that machine, makes use of some vulnerability to achieve local access as a user of that machine.
- **Probes:** Probing is a category of attacks where an attacker examines a network to collect information or discover well-known vulnerabilities.

2.1.2 Types of Intrusion detection systems

Intrusion detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level. Intruders have signatures that can be detected using software. Based upon a set of signatures and rules, the detection system is able to find and log suspicious activity and generate alerts [**25**].

Debar et al. [**29**] introduces five ways to classify intrusion detection systems are introduced.

- The *detection method* describes the characteristics of the detector. When the intrusion detection system uses information about the attacks, it is qualified as knowledge based. But when the intrusion detection system uses information about the normal behavior of the system it monitors, it is qualified as behavior based.
- The other classification metrics is *behavior on detection* which describes the response of the intrusion detection system to attacks. When the intrusion detection system actively reacts to the attack by taking either corrective or pro-active actions, it is

said to be active. But if the intrusion detection system merely generates alarms it is said to be passive.

- The *audit source location* discriminates intrusion detection systems based on the kind of input information they analyze. The source of the audit information can be a host, network packets, and application logs.
- The *detection paradigm* describes the detection mechanism used by the intrusion detection system. IDS can evaluate states or transitions for identifying intrusion.
- The *usage frequency* is another classifying metrics. Some IDSs have real-time continuous monitoring capabilities whereas others have to be run periodically.

2.1.2.1 Signature Based IDS

A signature based IDS will monitor packets on the network and compare them against a database of signatures or attributes from known malicious threats [**30**]. Signature based IDS are also called *misuse detection* IDS [**1**] [**10**]or *Knowledge based* IDS [**29**] [**31**].Intrusions are detected by matching actual behavior recorded in audit trails with known suspicious patterns [**1**]. It searches for patterns of program or user behavior that match known intrusion scenarios, which is stored as signatures. If a pattern match is found, an alarm is raised [**10**]. While signature based detection is fully effective in covering known attacks, it is useless when faced with unknown or novel forms of attacks for which the signatures passes all possible variations of the attack. Any mistakes in the definition of these signatures will increase the false alarm rate and decrease the effectiveness of the detection technique. In addition, maintenance of the knowledge base of the intrusion detection system requires careful analysis of each of vulnerability and is therefore a time-consuming task [**29**].

2.1.2.2 Anomaly Based IDS

Different from misuse detection, anomaly detection is dedicated to establishing normal activity profiles for the system. It is based on the assumption that all intrusive activities are necessarily anomalous. Anomaly detection studies start by forming an opinion on what the normal attributes for the observed objects are, and then decide what kinds of activities should be flagged as intrusions and how to make such particular decisions [**1**].

Anomaly based detection is also called behavior based detection [**29**]. An intrusion can be detected by observing a deviation from the normal or expected behavior of the system or the users. The model of normal or valid behavior is extracted from reference information collected by various means. The IDS compares this model with the current activity and an alarm is generated in case a deviation from a normal behavior is found [**29**] [**30**].

The primary advantage of anomaly detection is its capability to find novel attacks; as such it addresses the biggest limitation of misuse detection [**1**] [**29**]. Anomaly or behavior based detection can even contribute to the automatic discovery of new attacks. They are less dependent on operating system specific mechanisms [**29**]. However, due to the assumptions underlying anomaly detection mechanisms, their false alarm rates in general is very high. The main reason for this is that one the user's normal behavior model is based on data collected over a period of normal operations. The other reason is anomaly detection techniques can hardly detect stealth attacks because these kinds of attacks are usually hidden in large number of instances of normal behaviors. Advanced statistics models, rule-based models, learning models, biological models and signal processing techniques based models are used as anomaly detection techniques.

2.2 Intrusion Detection Using Data Mining Techniques

Most commercial intrusion detection systems have limitations and do not provide a complete solution [**10**]. Misuse detection searches for patterns or user behaviors that match known intrusion scenarios, which are stored as *signatures*. These hand-coded signatures are laboriously provided by human experts based on their extensive knowledge of intrusion techniques. If a pattern match is found, this signals an event for which an alarm is raised. Human security analysts evaluate the alarms to decide what action to take. An intrusion detection system for a large complex network can typically generate thousands or millions of alarms per day, representing an overwhelming task for the security analysts [**10**]. Because systems are not static, the signatures need to be updated whenever new software versions arrive or changes in network configuration

occur. In addition, major drawback is that misuse detection can only identify cases that match the signatures. That is, it is unable to detect new or previously unknown intrusion techniques [**10**].

Anomaly detection is that it may detect novel intrusions that have not yet been observed. Typically, a limiting factor of anomaly detection is the high percentage of false positives [**1**] [**29**] [**30**]. A false positive occurs when normal behavior is mistakenly classified as malicious and treated accordingly [**11**].
In addition, as stated by Lappas and Pelechrinis [**11**], traditional intrusion detection systems have a number of significant draw backs. Current IDS are usually tuned to detect known service level network attacks. This leaves them vulnerable to original and novel malicious attacks. Another aspect is data overload which does not relate directly to misuse detection but is extremely important is how much data an analyst can efficiently analyze. That amount of data it needs to look at seems to be growing rapidly. Depending on the intrusion detection tools employed by a company and its size there is the possibility for logs to reach millions of records per day. In addition, false negative is the other drawback of traditional IDS. This is a case where an IDS does not generate an alert when an intrusion is actually taking place Provided the above mentioned limitations of traditional Intrusion Detection System, data mining can help improve intrusion detection by addressing the aforementioned problems [**1**] [**10**] [**11**] [**12**] [**32**].

One main challenge in intrusion detection is that we have to find out the concealed attacks from a large quantity of dataset. Several data mining and machine learning (ML) algorithms, including Neural Network , Support Vector Machine , Genetic Algorithm , Fuzzy Logic , and Data Mining and more have been extensively employed to detect intrusion activities both known and unknown from large quantity of complex and dynamic datasets [**12**]. Generating rules is vital for IDSs to differentiate standard behaviors from strange behavior by examining the dataset. A number of researches with

data mining as the chief constituent has been carried to find out newly encountered intrusions [**12**].

Data mining based intrusion detection systems can be categorized into two major groups [**10**] [**33**]: *misuse detection* and *anomaly detection*. In misuse detection, a model is trained with labeled data to recognize the patterns of "normal" visits and "intrusion" attempts. A classifier can then be derived to detect known intrusions. Research in this area has included the application of classification algorithms, association rule mining, and cost sensitive modeling [**10**]. Decision tree generate classifiers by learning based on a sufficient amount of normal or abnormal audit data [**1**]. Signatures of different types of intrusions are learnt automatically, and they are much more powerful than manually defined signatures in recording the subtle characteristics. Misuse detection has been shown to be very successful in detecting previously known attacks. However, since the misuse model is highly dependent on the labeled data used in the training stage, its capabilities of detecting new intrusion types is limited.

Different from misuse detection, anomaly detection first establishes a model of normal system behaviors, and anomaly events are then distinguished based on this model [**10**]. The implicit assumption is that any intrusive activity will be anomalous. Anomaly detection is able to detect newly emerging attacks, but it also has some drawbacks [**10**]. It may fail to detect some known attacks if the behaviors of them are not significantly different from what is considered to be normal. Moreover, the false alarm rate tends to be high when the data of some normal system behaviors are not involved in the training phase. Anomaly detection research has included the application of classification algorithms, statistical approaches, clustering, and outlier analysis. The techniques used must be efficient and scalable, and capable of handling network data of high volume, dimensionality, and heterogeneity.

Therefore, in comparison to traditional intrusion detection systems, data mining based one is generally more precise and require far less manual processing and input from human experts [**10**].

2.3 Data mining and knowledge discovery

Data mining (DM) is a subfield of Machine Learning that enables finding interesting knowledge (patterns, models and relationships) in very large databases [**5**]. It is the most essential part of the knowledge-discovery process, which combines databases, statistics, artificial intelligence and machine learning techniques. Joyce [**34**] also stated that, data mining is an extension of traditional data analysis and statistical approaches in that it incorporates analytical techniques drawn from a range of disciplines including, but not limited to numerical analysis, pattern matching and areas of artificial intelligence such as machine learning, neural networks and genetic algorithms. While many data mining tasks follow a traditional, hypothesis-driven data analysis approach, it is commonplace to employ an opportunistic, data driven approach that encourages the pattern detection algorithms to find useful trends, patterns, and relationships.

Han and Kamber [**10**] stated, data mining refers to extracting or mining knowledge from large amount of data. Many people treat data mining as synonym for Knowledge Discovery in Database (KDD). The term Knowledge Discovery in Databases (KDD) is generally used to refer to the overall process of discovering useful knowledge from data, where data mining is a particular step in this process [**34**].

Knowledge Discovery consists of the seven steps (the steps are discussed under section 1.6.3). Han and Kamber [**10**] agreed that data mining as essential step in Knowledge Discovery in Database but adopted the more general term data mining rather than knowledge discovery. The authors adopted data mining as the process of discovering interesting knowledge from large amounts of data stored in databases, data warehouses, or other information repositories.

Data mining has attracted a great deal of attention 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 [**10**]. The information and knowledge gained can be used for applications ranging from market analysis, fraud detection, intrusion detection and customer retention, to production control and science exploration.

2.4 Data mining tasks

Data mining functionalities are used to specify the kind of patterns to be found in data mining tasks. In general, data mining tasks can be classified into two categories: descriptive and predictive [**10**]. Citing Dunham(2003) Siraj [**35**] 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.

Descriptive model identifies patterns or relation hips in data. Unlike to the predictive morel, it serves as a way to explore the properties of the data examined, not to predict new properties [**36**]. The descriptive task encompasses methods such as clustering, summarization, association rules and sequence analysis. In this study, predictive models are used for acquiring knowledge.

2.5 Classification Algorithms

Classification and prediction are two main forms of data analysis that can be used to extract predictive models describing important data classes or to predict future data trends. Such analysis can help provide us with a better understanding of the data at large. Whereas classification predicts categorical (discrete, unordered) labels, *prediction* models

continuous valued functions. For example, we can build a classification model to categorize network incidents as either "normal" or an "attack".

Classification is the derivation of a function or model which determines the class of an object based on its attributes. A set of objects is given as the training set in which every object is represented by a vector of attributes along with its class. A classification function or model is constructed by analyzing the relationship between the attributes and the classes of the objects in the training set. Such a classification function or model can be used to classify future objects and develop a better understanding of the classes of the objects in the database [**37**]. As mentioned in [**38**] and [**10**] , classification is also called supervised learning. It is called supervised learning because it works on labeled attributes in which there is a specially designated attribute and the aim is to use the data given to predict the values of that attribute for instances that have not yet been seen. The designated attributes in classification are categorical such as 'very good', 'good', or 'poor' [**38**].

Classification is a two step process [**10**] consisting of model construction and model usage. In the first step, a classifier is built describing a predetermined or labeled set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database instances and their associated class labels. This step is called model construction.

Generally, classification is a process of building model that describe data class and used to predict the class of objects whose class label is unknown. It finds out the relationship between predictor value and the target value. The model is based on the analysis of a set of training data. The data; historical, for a classification is typically divided into two datasets: one for building the model; the other for testing the model. Thus the various classification approaches can be employed on network data for obtaining specific information and detecting intrusion. Decision tree, K-nearest neighbor, Byes classifier,

neural network, support vector machine and rule based learning are some of the classification data mining techniques. In this report, decision tree and rule based learning (production rules) are discussed

2.5.1 Decision tree

Decision trees classify instances by sorting them based on feature values [**39**]. As depicted in figure 2.1 each node in a decision tree represents an optimal attribute of 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. The features that best divides the training data would be the root node of the tree [**39**].

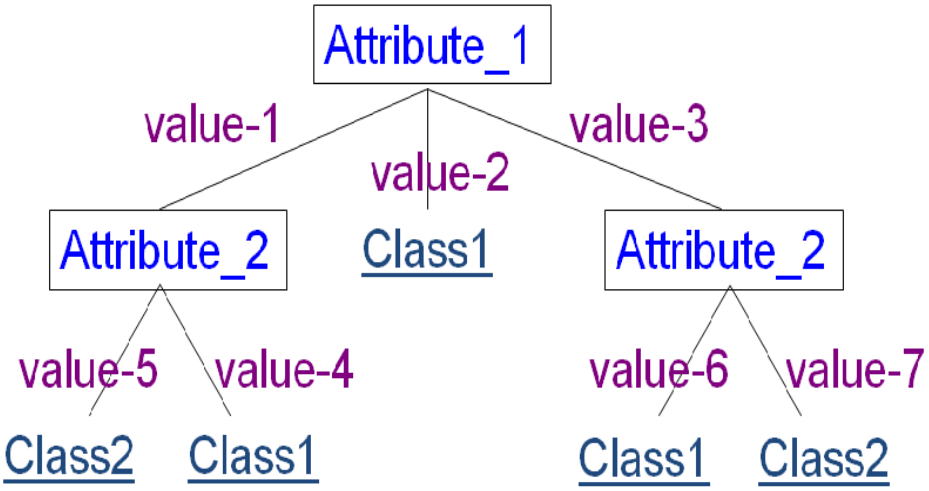

Figure 2-1 A hypothetical Decision tree

Decision tree aims at developing classification rules from the data in the training set [**38**]. A decision tree is created by a process called splitting on the value of attributes, i.e. testing the value of an attribute and then creating a branch for each of its possible values [**10**] [**38**]. In the case of continuous attributes the test is normally whether the value is 'less than or equal to' or 'greater than' a given value known as the split value [**38**].

ID3, C4.5 and CART, in their respective order of invention and usage, are algorithms used in decision tree construction. They adopt a greedy approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner. Most algorithms for decision tree induction also follow such a top-down approach, which starts a training set

of instances and their associated class labels. The training set is recursively partitioned into subsets as the tree is built [**10**].

The basic algorithm for decision tree induction is greedy algorithm that constructs decision trees in a top-down recursive divide-and –conquer manner [**39**]. The algorithm is summarized as follows.

Create a node N;
If samples are all of the same class, C then
Return N as a leaf node labeled with the class C;
If attribute-list *is empty then*
Return N as a leaf node labeled with the most common class in samples;
select test-attribute, the attribute among attribute-list with the highest information gain;
label node N with test-attribute;
for each known value a_i *of test-attribute*
grow a branch from node N for the condition
test-attribute= a_i*;*
let s_i *be the set of samples for which test-attribute=* a_i*;*
If s_i *is empty then*
attach a leaf labeled with the most common class in samples;
else attach the node returned by
Generate_decision_tree(si,attribute-list_test-attribute)

2.5.1.1 Attribute selection measures

An attribute selection measure for building decision tree is a heuristic for selecting the splitting criterion that best separates a given data partition of class-labeled training instances into individual classes. All the instances are pure (i.e., all of the instances that fall into a given partition belong to the same class). The best splitting criterion is the one that most closely results in such a scenario. The attribute selection measure provides a ranking for each attribute describing the given training instances. The attribute having the best score for the measure is chosen as the *splitting attribute* for the given instances. If the splitting attribute is continuous-valued or if we are restricted to binary trees then, respectively, either a *split point* or a *splitting subset* must also be determined as part of the splitting criterion. The tree node created for partition, let's say *D*, is labeled with the

splitting criterion, branches are grown for each outcome of the criterion, and the instances are partitioned accordingly [**10**]. This section describes three popular attribute selection measures—*information gain, gain ratio*, and *gini index*.

- **Information gain**

Information gain for attribute selection measure is based on the work of Claude Shannon on information theory, which studied the value or information content of messages. ID3 uses information gain for attribute selection measure. The notion used is as follows:-

Let D , the data partition, be a training set of class labeled instances. Suppose the class label attribute has *m* distinct values defining *m* distinct classes, C_i*(for i=1, …, m).* let $C_{i,D}$ be the set of instances of class C_i in D. let $|D|$ and $|C_{i,D}|$ denote the number of instances in D and $C_{i,D,}$ respectively.

The attribute with the highest information gain is selected as the splitting attribute. This attribute minimizes the information needed to classify the instances in the resulting partitions and reflects the least impurity in these partitions. Entropy (impurity) is used to measure the information content of the attributes. High entropy means the attribute is from a uniform distribution where as low entropy means the attribute is from a varied distribution. Entropy is defined as follow. Let pi be the probability that an arbitrary instance in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$. Expected information (entropy) needed to classify an instance in D is given by:

$$E(D) = -\sum_{i=1}^{m} pi \log(pi) \text{…….(2.1)}$$

E (D) (entropy of D) - is the average amount of information needed to identify the class label of an instance in D. The smaller information required, the greater the purity.

At this point, the information we have is based solely on the proportions of instances of each class. A log function to the base 2 is used, because the information is encoded or measured in bits.

Suppose attribute A can be used to split D into v partitions or subsets, {D1,D2,…, Dv}, where Dj contains those instances in D that have outcome aj of A. Information needed (after using A to split D) to classify D:

$$\text{Info}_A(D)=\sum_{j=1}^{v}\frac{Dj}{D}\ \text{X}\ \text{Info(Dj)} \text{.....(2.2)}$$

The smaller the expected information required, the greater the purity of the partitions. Information gained by branching on attribute A is given by

$$\boldsymbol{Gain(A)=E(D)-Info_A(D)}\text{……..(2.3)}$$

Information gain increases with the average purity of the subsets. The attribute that has the highest information gain among the attributes is selected as the splitting attribute.

- **Gain ratio**

The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values. This may result in selection of an attribute that is non-optimal for prediction. C4.5, a successor of ID3, uses an extension to information gain known as *gain ratio*, which attempts to overcome this bias. It applies a kind of normalization to information gain using a "split information" value defined analogously with *Info*(*D*) as:

$$\text{SplitInfo}_A(D)= -\sum_{j=1}^{v}\frac{|Dj|}{|D|}\text{X}log\frac{|Dj|}{|D|}\text{……(2.4)}$$

This value represents the potential information generated by splitting the training data set, *D*, into *v* partitions, corresponding to the *v* outcomes of a test on attribute *A*. Note that, for each outcome, it considers the number of tuples having that outcome with respect to the total number of tuples in *D*. It differs from information gain, which measures the information with respect to classification that is acquired based on the same partitioning [**10**]. The gain ratio is defined as:

$$\text{GainRatio(A)}=\frac{Gain(A)}{SplitInfo(A)}\text{……..(2.5)}$$

The attribute with the maximum gain ratio is selected as the splitting attribute. Note, however, that as the split information approaches 0, the ratio becomes unstable. A constraint is added to avoid this, whereby the information gain of the test selected must be large—at least as great as the average gain over all tests examined.

- Gini index

The Gini index is used in CART. Using the notation described above, the Gini index measures the impurity of *D*, a data partition or set of training tuples [**10**], as

$$\text{Gini(D)}=1-\sum_{i=1}^{m} pi^2 \ldots\ldots..(2.6)$$

where *pi* is the probability that a tuple in *D* belongs to class *Ci* and is estimated by $|C_{i,D}|/|D|$. The sum is computed over *m* classes. To determine the best binary split on A, we examine all the possible subsets that can be formed using known values of A and need to enumerate all the possible splitting points for each attribute. If A is discrete-valued attribute having v distinct values, then there are 2v-v possible subsets. When considering a binary split, we compute a weighted sum of the impurity of each resulting partition. If data set D is split on A into two subsets D1 and D2, the *gini* index *gini(D)* is defined as [**10**] :

$$\text{Gini}_A\text{ (D)}= \frac{|D1|}{|D|}\text{Gini}_{(D1)} + \frac{|D2|}{|D|}\text{Gini}_{(D2)} \ldots\ldots..(2.7)$$

First we calculate Gini index for all subsets of an attribute, then the subset that gives the minimum Gini index for that attribute is selected. The strategy is similar to that described for information gain. The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute. The reduction in impurity that would be incurred by a binary split on attribute A is

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \ldots\ldots..(2.8)$$

The attribute that maximizes the reduction in impurity (or has the minimum Gini index) is selected as the splitting attribute.

To summarize, the three measures for attribute selection are used mostly. Information gain is biased towards multi valued attributes. Whereas Gain ratio tends to prefer unbalanced splits in which one partition is much smaller than the others. Gini index is biased to multi valued attributes and has difficulty when the number of classes is large.

- **J48 classification algorithm**

J48 is an implementation of Quilan algorithm (C4.5). J48 classifier build a decision tree for the given data set, whose nodes represent discrimination rules acting on selective features by recursive partitioning of data using depth-first strategy [**40**].

The algorithm used each attribute of the data to make decision by splitting the data into smaller subsets. All the possible tests are considered during decision making based on information gain value of each attribute.

2.5.2 Rule based classification

Rule based classifiers group instances by using a set of "IF…..THEN" rules.

Rule :(Condition) -> X

Where,

- **Condition** is a conjunction of attributes like (A_1=v_1) and (A_2=v_2) and …. and (A_n=v_n) and
- **X** is a class label.

For example: (service=imap4) ^ (dst_host_count =< 11) → R2L

Rules are comprised of Left Hand Side (LHS) also called antecedent or condition and Right Hand Side (RHS) also called rule consequent or conclusion [**41**]. A given rule *r* covers an instance *z* if the attributes of the instance satisfy the condition (LHS) of the rule [**22**]. Rule based classification techniques are divided into two namely; direct method and indirect method. Rules based classifiers which extract rules directly from data, for example RIPPER, are called direct methods [**42**]. Indirect methods are those that extract

rules from other classification model like decision tree, for example C4.5 rules [**43**]. Direct methods first grow a single rule (Rule growing) then remove instances from this rule (Instance Elimination) after that prune the rule (Stopping Criterion and Rule Pruning) and then finally add rules to current rule set. PART and JRIP are algorithms which are rule based classifiers.

- **JRIP**

JRIP is a prepositional rule learner, i.e Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [**42**].

Rules in this algorithm are generated for every class in the training set and are then pruned. The discovered knowledge is represented in the form of IF-THEN prediction rules, which have the advantage of being a high-level and symbolic knowledge representation contributing towards the comprehensibility of the discovered knowledge.

JRip is based on the construction of a rule set in which all positive instances are covered by partitioning the current set of training instances into two subsets namely a growing set and a pruning set. The rule is constructed from instances in the growing set. Initially the rule set is empty and the rules are added incrementally to the rule set until no negative instances are covered. Following this the algorithms substitutes or revises individual rules by using reduced error pruning in order to increase the accuracy of rules. To prune a rule the algorithm takes in account only a final sequence of conditions from the rule and sorts the deletion that maximizes the function **[40]**.

- **PART**

PART is rule based classifier which generates rules repeatedly producing partial decision trees [**44**].As cited by Datta and Saha [**22**] Frank and Witten (1998) stated that, the PART technique avoids global optimization in which pruning is effected after all rules are generated used in C4.5 and RIPPER. It builds a partial decision tree to obtain a rule using C4.5's procedures to build a tree. It identifies the rule that identifies many instances using

separate and conquer then separate them out, repeat and makes the best leaf into a rule [**22**] [**44**].

2.7 Knowledge Based System

Knowledge based system (KBS) is a system that uses artificial intelligence (AI) to solve problems. It consists of a repository of expert knowledge with utilities designed to facilitate the knowledge retrieval in response to specific queries, along with learning and justification, or to transfer expertise from one domain of knowledge to another, in particular, KBS focus on using knowledge based techniques to support human decision making, learning, and action. Such systems are capable of cooperating with human users and are being used for problem solving, training , and assisting users and experts of the domain for which the system are developed, in some cases KBSs are even better than human are, as they are enriched with the virtues of efficiency and effectiveness [**45**].

Many KBS solutions currently are in use. In fact, a KBS is computer-based system that uses and generates knowledge from data, information, and knowledge [**45**]. These system are capable of understanding the information being processed and can make decision based on it, where as the traditional computer systems such as transaction processing, management information systems do not know or understand the data/information they process [**46**].

With the availability of advanced computing facilities and other resources, focus is now turning to more demanding tasks that might require intelligence [**46**]. Society and industry are becoming knowledge-oriented and relying on different experts' decision – making abilities to solve problems. A KBS can act as an expert on demand, anytime and anywhere. A KBS can save memory by leveraging experts, allowing users to function at a higher level and promoting consistency. A KBS is productive tool that offers collective knowledge of one or more experts [**46**].

The primary goal of knowledge-based systems is to make expertise available to decision-makers who need answers quickly. Expertise is often unavailable at the right place and

the right time. Portable computers loaded with in-depth knowledge of specific subjects can bring years' worth of knowledge to a specific problem [**47**].

The sources of knowledge are of two types [**45**]; documented knowledge and undocumented knowledge. Undocumented knowledge found in people's mind. Knowledge can be identified and collected by using one or several of human senses. Or it can be identified and collected by machines such as sensors, scanners, cameras, pattern matchers, intelligent agents). This multiplicity of knowledge sources and the types of knowledge contributes to the complexity of knowledge acquisition [**45**].

2.7.1 Categories of knowledge

According to [**17**], knowledge is categorized as declarative knowledge, procedural knowledge and Meta knowledge.

Declarative knowledge is a descriptive representation of knowledge. It tells about facts and is expressed in a factual statement. It is considered shallow or surface level, information that experts cab verbalize. Declarative knowledge is important in the initial stage of knowledge representation.

Procedural knowledge deals with the manner in which things work under different situations. It includes step-by-step sequences and how to types of instructions and may also include explanations. It involves automatic responses to stimuli and also tells us how to use declarative knowledge and how to make inferences. Declarative knowledge relates to a specific object. It includes information about the meaning, roles, environment, resources, activities, associations and outcomes of the object. Whereas procedural knowledge relates to the procedures used in problem-solving process for example, information about problem definition, data gathering, the solution process, evaluation criteria.

Meta knowledge is knowledge about knowledge. It is knowledge about the operation of the knowledge based system i.e. about their reasoning capabilities.

2.7. 2 Knowledge Engineering

Knowledge engineering (KE), is the process of obtaining knowledge from experts and building a knowledge base [**17**]. As cited in [**17**] Feigenbaum and McCorduck (1983), defined the activity of knowledge engineering as the art of bringing the principles and tools of artificial intelligence research to show on difficult applications problems requiring experts' knowledge for their solutions. KE involves cooperation of human experts in the domain working with the knowledge expert to codify and make explicit the rules that a human expert uses to solve real problems. KE can be viewed from two perspectives: narrow and broad [**48**]. The narrow perspective KE deals with knowledge acquisition, representation, validation, inferencing, explanation, and maintenance. According to the broad perspective the term defines the entire process of developing and maintaining intelligent systems. A major goal of KE is to help experts articulate what they know and document the knowledge in a reusable form [**48**].

2.7.3. Architecture of Knowledge based System

Figure 2-2 depicts a knowledge based system architecture suggested by Krishnamoorth and Rajeev [52].

- Knowledge acquisition: - involves the acquisition of the knowledge from human experts, books, documents, sensors or computer files. The knowledge may be specific to problem domain or general knowledge or it may be Meta knowledge.
- Knowledge representation:- is an activity of organizing acquired knowledge so that it will be ready for use. It involves preparation of a knowledge map and encoding of the in knowledge in knowledge base.
- Knowledge validation: - involves validating and verifying the knowledge and showing testing results to domain experts to verify its accuracy.
- Inferencing: - involves the design of software to enable the computer to make inferences based on the stored knowledge and the specifics of the problem so that the system can provide advice to users who are not experts.

- Explanation and justification: - involves the design and programming of explanation capability. For example, the ability to answer questions such as why a specific piece of information is needed by the computer or how a certain conclusion was derived by the computer

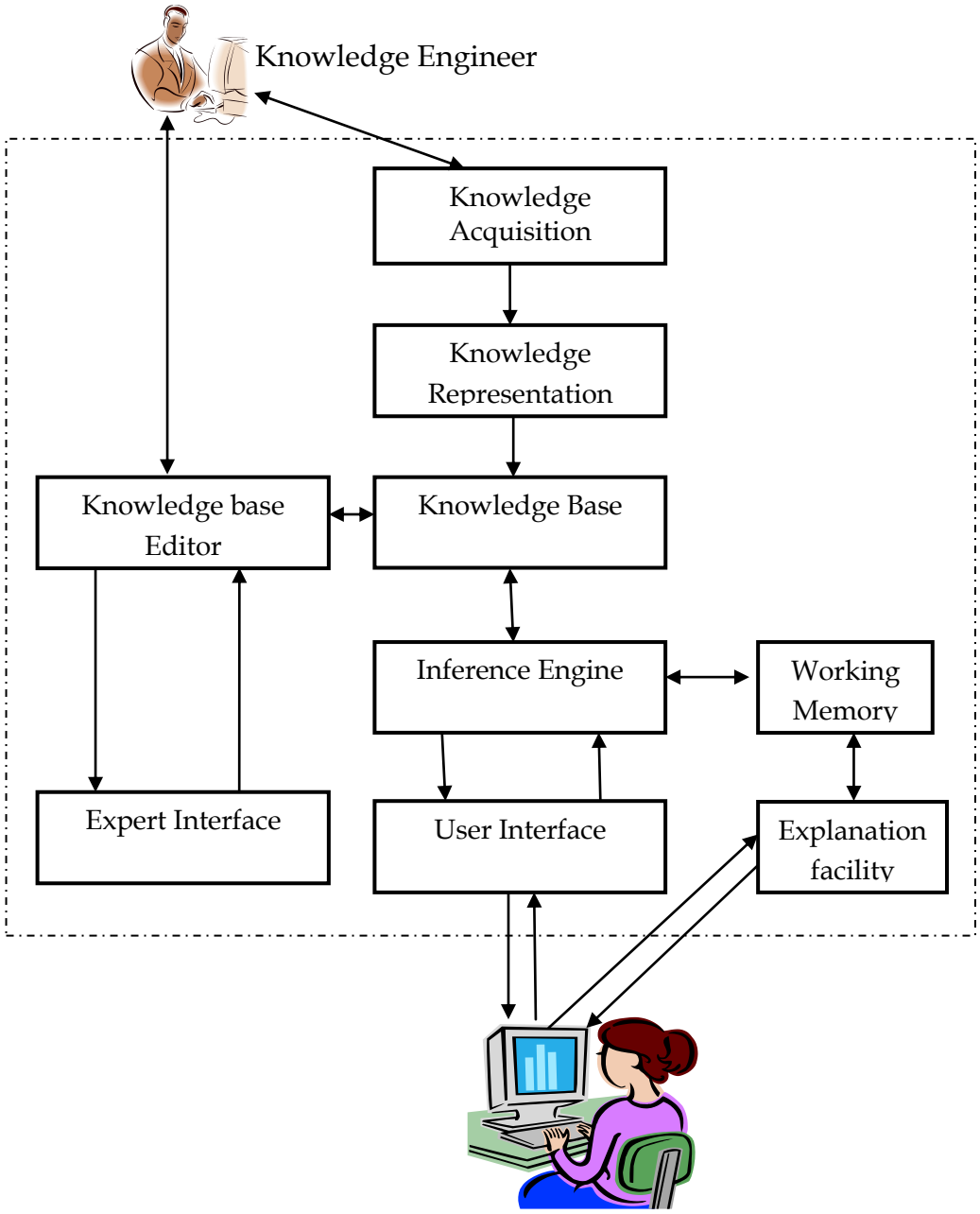

Figure 2-2 Architecture for Knowledge Based system

2.7.3.1 Knowledge acquisition

Knowledge acquisition is a process of identifying the knowledge, representing the knowledge in a proper format, structuring the knowledge, and transferring the knowledge to a machine. This process can be affected by the roles of the knowledge engineer, the expert and the end user.

Knowledge acquiring from experts is difficult task. Some of the factors that made it difficult are listed below.

- Experts may not know how to articulate their knowledge or even may not able to do so.
- Experts may not be willing or have time scarcity
- System builders tend to gather knowledge from one source, but the relevant knowledge is distributed across several sources.
- The knowledge collected may not be complete due to builder's attempt to collect from documented knowledge
- Experts may not show consistent behavior when they are observed or interviewed.

Elicitation of knowledge from expert can be seen as a process of modeling and can be made manually or with the help of computers.

Knowledge acquisition can be accomplished in three ways. These are manual, semiautomatic and automatic.

Manual methods: - includes interview (structured and unstructured), process tracking, protocol analysis and observation [**17**].

Semiautomatic knowledge modeling methods: - in this method the process of knowledge acquisition can be supported by computer-based tools. These tools offer a situation in which knowledge engineers or experts can identify knowledge through an interactive process [**17**].

Automatic knowledge modeling methods: - this method is similar in the manner of using computers to aid the knowledge acquisition process in which knowledge can be extracted automatically from existing data. The process of extracting knowledge from data is called knowledge discovery. Automatic knowledge acquisition is advantageous as compared to semiautomatic and manual elicitation methods due to good knowledge engineers are expensive and difficult to find and domain experts are usually busy and sometimes uncooperative. Machine learning and data mining approaches can be followed for automatic knowledge acquisition [**17**].

The following are the objectives of knowledge acquisition automatically:

- To increase the productivity of knowledge engineering (reduce the cost)
- To drastically reduce or even eliminate the need for an expert
- To increase the quality of the acquired knowledge.

2.7.3.2 Knowledge Representation

Knowledge representation is a way of representing a validated knowledge collected from experts or induced from a set of data in a manner understandable by humans and executable on computers. It is a systematic means of encoding knowledge of human experts in an appropriate medium [**49**]. There are different ways of representing knowledge [**49**] [**17**]; the most popular ways/methods are production rule, Frames, Decision trees, Objects and Logic

Production rules: - are the most popular methods or knowledge representation. Production rules 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>

IF condition (premise or antecedent) occurs, THEN some action (conclusion or consequence) will occur. Rules can be viewed as a simulation of the cognitive behavior of human experts. Knowledge based rules are dependant to each other in reality or better to

say highly interdependent. For example, a new rule that is added may conflict with an existing rule or may require a revision of attributes or rules.

Two types of rules are common in AI; knowledge and inference rules. Knowledge rules or declarative rules state all the facts and relationships about a problem. Whereas inference rules or procedural rules advise on how to solve a problem provided that certain facts are known. The knowledge engineer separates the two types of rules where knowledge rules go to the knowledge base and inference rules become part of the inference engine. Production rules have the following advantages **[50]**:

- Production rules have advantage of notational convenience that is it easy to express suitable pieces of knowledge in this way
- Restricted power of expression
- Declarative form of knowledge representation.
- Allows expanding the knowledge base by adding more rules at the end of the rule base.
- Rules are very easy to understand, criticize and improve.

Decision trees:- simplify the process of knowledge acquisition process. Decision trees can easily be converted to rules. The conversion can be accomplished by a computer program. In fact machine learning methods can extract decision trees automatically from textual sources and converting them into rules bases.

Predicate calculus

Predicate logic is used for showing logic relationships and their reasoning. Facts and observations in a problem domain are defined as premises, which are used by the logical process to derive new facts and conclusions. Symbolic logic system is used as a method to represent rules and procedures in order for a computer to perform reasoning using logic. Symbolic logic is permits to draw inference from premises using logical techniques.

There are two forms of computational propositional logic (propositional calculus) and predicate logic (predicate calculus). A proposition is a statement that is either true or false. Rules are used to determine the truth or falsity of new proposition. In propositional logic symbols such as letters are used to represent propositions, premises, or conclusions. Logical connectives such as AND, OR, NOT, IMPLIES and EQUIVALENT are used to form a more complex by combining two or more propositions.

Propositional logic deals with complete statements and has a limitation of representing real world knowledge. Hence, artificial intelligence uses predicate logic. Predicate logic allows breaking a statement down into its component parts. In addition, it allows using variables and functions of variables in a symbolic logic statement. Predicate logic is a basis for PROLOG (Programming in Logic). Predicate provides the theoretical foundation for rule-based systems. This fact and rules expressed within the language form basis for inferencing.

2.7.3.3 Knowledge Base

Knowledge base system consists of a set of rules (called the rule base) and data or facts (called the database). The rule base and the database of a KBS together are called the knowledge base (KB) [**48**]. It is the central component of a knowledge-based agent. Knowledge base is a set of sentences expressed in knowledge representation language and each sentences are an assertion about the world [**51**]. It contains domain specific knowledge required to solve the problem. The knowledge base is created by the knowledge engineer using different knowledge acquisition techniques [**52**].

2.7.3.4 Inference Engine

The inference engine is usually set up to mimic the reasoning, or problem-solving ability, that the human expert would use to arrive at a conclusion. The inference engine simulates the evaluation process of relating the information and rules in the knowledge base to the answers to a series of questions given by the operator [**47**]. The inference engine deduces facts or draws conclusions from the knowledge base based on the user input and the facts from the knowledge base [**49**]. It is a software program that refers to the existing

knowledge, manipulates the knowledge and makes decisions about actions to be taken. It generally uses pattern matching and searching techniques for drawing conclusions. Through these procedures, the inference engine examines existing rules and facts and adds new facts when possible. In other words, inference engine in addition to referring knowledge available it also infers new knowledge when needed [**45**]. The inference engine and knowledge base exist as two separate modules that work closely together [**49**].

2.7.4 Knowledge validation

According to Mehdi [**18**], knowledge discovered from database needs to be validated to make sure its accuracy, consistency, and completeness. Knowledge base testing schemes employed in expert system can be adapted to validate the set of generated rules from databases. In general, errors in rule-based knowledge bases can be classified into two namely, consistency (which includes redundant, conflicting and subsumed rules) and completeness (which includes unreferenced attributes, illegal attributes, and unreachable rules).

Mehdi [**18**] has also pointed out approaches to validating knowledge bases discovered from databases. He stated that there are some sort of similarities between the knowledge bases discovered or generated from databases and those defined for expert systems in that both can have redundant, contradictory, subsuming and missing rules. As a result knowledge base established schemes for expert system can be used for those generated from databases. However, knowledge discovered from databases is different from expert system in the way they are created, which in return affects the way they are validated. He stated a number of ways of validating discovered knowledge from database but for the sake of this research validation of discovered knowledge based on domain knowledge is employed.

Domain or background knowledge can be defined as any information that is not unequivocally presented in the database. Domain knowledge can be used to verify whether a contradictory discovered knowledge is indeed contradictory or if a consistent discovered knowledge is in reality, accurate or inaccurate. In addition, domain

knowledge can be used to make sure whether discovered rules are incomplete rules and redundant rules [**18**]. In general, domain knowledge is given by domain expert and represents some knowledge about some attributes in the database.

- Domain knowledge can be used to verify whether *contradictory discovered rules* are really *contradictory* or *accurate*
- Analyzing the discovered rules (known as statistical dependencies) with the available functional dependencies (known as domain knowledge) is a scheme for validating the completeness or incompleteness of the discovered knowledge.

Domain Expert: - is a person who expertise in his/her domain area. For example a medical doctor who gives medical aid for diabetic patients is a domain expert. In addition, a network administrator who manages and administers a given network is a domain expert in his domain.

Knowledge Engineer: - is one who gathers knowledge from experts through interview or using automatic knowledge acquisition techniques. The knowledge engineer has to have the knowledge of a knowledge base development technology and should know how to develop knowledge based system using a development environment. It is not necessary that the knowledge engineer be proficient in the domain in which the expert system is being developed but general knowledge and familiarity with key terms is desirable [**52**].

2.7.5 Forward and Backward Chaining

Depending on the type of applications they are aimed at, inference engines in rule based system can use different strategies to derive the goal (i.e. new fact). The most common strategies used are Forward Chaining and Backward Chaining [**53**].

Forward chaining is data driven strategy [**52**] [**53**] in that the system starts with the initial set of elements in the working memory and keeps on firing rules until there are no rules which can be applied or the goal has been reached. Consequently, the system is moving forward from the current state to a goal state. Different from forward chaining, backward

chaining is a goal driven strategy [**52**] [**53**]. It involves dividing a problem into sub-problems and solving each one of them. Which means the goal is reduced to sub-goals and each sub-goal is reduced further, and so on until they are solvable directly.

The order in which rules appear in the rule base plays a major role in the way inference is carried out in forward chaining, whereas such order does not play any role in backward chaining. But the order in which conditions are listed in a rule is important in backward chaining. Backward chaining tries to establish goals in the order in which they appear in the knowledge base [**52**].The order in which questions are asked to the user for response depends on this order. Hence, before formulating the rule base, the knowledge engineer should decide whether backward chaining or forward chaining is going to be adopted for reasoning [**52**]. Forward chaining strategy is suitable for applications in which the number of goal states is large. But, backward chaining is used when the number of possible initial states are large, compared to the number of goal states. The tasks of classification and diagnosis are best suited for backward chaining [**53**].

2.7.6 AI Programming Languages

Prolog and Lisp are two of the most popular AI (Artificial Intelligence) computer programming languages [**54**].

PROLOG:- is an AI programming language which belongs to the family of logic programming. Prolog is declarative language, in which computations are carried over by running queries over the relations defined as rules and fact [**55**]. It was first used for natural language processing, but now it is being in use for different tasks such as expert systems, automated answering systems, games, and advanced control systems. Prolog has data type called term. A Term can be an atom, number, variable or a compound term [**54**]. Numbers can be float or integers. Prolog supports lists and strings as collection of items. Prolog defines relations using clause, which can be either rules or facts. Prolog allows iteration through its recursive predicates. A program is executed by an inference engine that answers a query by searching relations systematically to make inferences that will answer a query [**55**].

LISP:- The name LISP comes from "**LIS**t **P**rocessing" and the it name implies lisp's major data structure is linked list. The whole source is written using lists (using prefix notation), or more correctly parenthesized lists. Therefore, it is called an expression oriented language, where all data and code are written as expressions [**54**].

As compared to PROLOG, LISP is functional language whereas the former is a logic programming language and declarative language [**54**] [**55**]. LISP is very flexible due to its fast prototyping and macro features, so it allows extending the language to suit for certain problem. LISP has been in extensive use in areas of AI because of its rapid prototyping ability. PROLOG is ideal for AI problems with symbolic reasoning, database and language parsing applications [**54**].

2.7.7 Evaluation of the models

In order to evaluate the performance of the classifier Prediction Accuracy, True Positive, False Positive, Precision, Recall are commonly used [**38**]. Confusion matrix helps to see a breakdown of a classifier's performance by showing how frequently instances of a class let us say class X are classified as class X or misclassified as some other class, say class Y [**38**].

Table 2-1 Confusion Matrix

		Predicted class		Total instances
		+	-	
Actual Class	+	True Positive	False Positive	Positive
	-	False Negative	True Negative	Negative

2.7.7.1 Prediction Accuracy

Prediction accuracy measures the proportion of instances that are correctly classified by the classifier.

$$\text{Predictive Accuracy} = \frac{Total\ Number\ of\ instances\ correctly\ classified}{Total\ Number\ of\ intances} * 100 \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.9)$$

2.7.7.2 True Positive rate and False Positive rate

In contrary to Predictive Accuracy TP rate and FP rate values do not depend on the relative size of positive and negative classes [**38**]. TP rate is the proportion of positive or correctly classified instances as positive or correct instances.

$$\text{True Positive rate} = \frac{True\ positive}{Positive} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.10)$$

FP rate also called False Alarm, measures the proportion of negative instances that are erroneously classified as positive.

$$\text{False Positive rate} = \frac{False\ Positive}{Negative} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.11)$$

2.7.7.3 Precision and Recall

Precision measures the proportion of instances that are classified as positive that are really positive.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad \ldots\ldots\ldots\ldots \quad (2.12)$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.13)$$

$$\text{F-measure} = \frac{2 * Precision * Recall}{Precision + Recall} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.14)$$

2.8 Related works

Ever since intrusions cause attacks on computer networks, a number of researches has been done to detect intrusions and identify which network traffic are normal or an attack. Different researchers around the world have been doing researches using different techniques of data mining to detect intrusions. In this section some research works which employed data mining techniques and knowledge based intrusion detection are reviewed. Few researches are done locally in intrusion detection using different data mining techniques. Kindie [**56**] proposed integrated multiple classifier learning algorithms comprising of C4.5, CART, NBTree, REPTree and PART intending for improving the performance of NIDS. He used KDDcup'99 intrusion data set for his experiment.

According to the researcher the algorithms performed differently for Normal, probe, DOS, U2R and R2L. Consequently the researcher selected C4.5 for probe attack, CART classifier for DOS attack, the NBTree classifier for R2L attacks, REPTree algorithm for normal and PART classifier for R2L attacks. He concluded that integrating multiple classifier algorithms perform better for all attack types than the KDD wine result.

Zewdie [**15**] studied an optimal feature selection for network intrusion detection. He proposed filter approach to select important features; namely, Information Gain Ration and Correlation Feature Selection to illustrate the significance of feature selection in classifying KDD intrusion detection dataset. Algorithms CS-CM4 (direct) and C 4.5(indirect) cost sensitive classifiers were on selected features. He reported that the CS-M4 and C4.5 provided an overall classification accuracy of 99.5% and 99.45% respectively.

Most recently Tigabu [**16**] designed a semi-supervised intrusion detection system employing J48 and Naïve Bayes classifier algorithms. He has used 21,533 intrusion dataset taken from Massachusetts Institute of Technology Lincoln laboratory. His experiment showed that J48 decision tree algorithm and a prediction accuracy of 96.11% on the training dataset 93.2% on the test dataset to classify the new instances as normal, DOS, U2R, R2L and probe classes. The researcher recommended designing knowledge based system which will add adaptability and extensibility features for intrusion detection.

Sumit et al. [**57**] presented a situation aware intrusion detection model that integrates heterogeneous data sources and build a semantically rich knowledge-base to detect cyber threats/vulnerabilities. Their work focused on integration of conventional signature-based intrusion detection with other data sources along with ontological reasoning which can lead to an intrusion detection system that has the ability to potentially link and infer means and consequences of cyber threats and vulnerabilities whose signatures are not yet available. The components of this system would be IDS/IPS sensor information module, data from different sensor streams, text-data from web, domain expert knowledge, the

ontology knowledge base and the reasoned. In this study however we use KDDcup'99 intrusion dataset as data source and knowledge is extracted from it

Guy et al. [**58**] proposed an artificially intelligent system for intrusion detection and countermeasures. The system would be built using distributed intelligent agents to apply data mining approach to intrusion detection. To develop prototype intelligent agent they used data from the University of New Mexico containing system call traces for normal and abusive use of the sendmail program as run on SunOs 4. Guy and his friends [**58**] study focused on semdamil program that run on SunOs 4 and it doesn't tell for which types of network attacks is the proposed system works.

Norbik et al. [**59**] proposed a hybrid Intrusion Detection System combining anomaly, misuse and host based detection. They proposed a dynamic model intelligent Intrusion Detection System. Data mining techniques such as association rules, fuzzy association rules and Self Organized Maps (SOM) are used to process network data. They recommended their system with an artificial immune system.

Genapathy et al. [**60**] an Intelligent Agent based Feature Selected Hybrid Classifier for detecting the intruders in wireless ad hoc network. Intelligent agent-based weighted outlier detection algorithm in combination with an intelligent agent-based enhanced multiclass Support Vector Machine algorithm for classification have been proposed in order to classify the attacks effectively. In addition, so as to improve the performance a new intelligent agent-based attribute selection algorithm is also used. A classification accuracy of 99.77% for DOS, 99.70% for Probe and 79.72% for other attacks has been achieved. The research of Genapathy and his friends [**60**] tailored to wireless and ad hoc networks only and lacks utilizing the model to take an action.

Khan [**61**], in his research used Genetic Algorithm to design Rule based network intrusion detection. The researcher used KDDcup'99 dataset for his study and focused on DOS and Probing types of attacks. It is determined that increasing the number of iteration of application algorithm contributes to the accuracy of the data. Besides the researcher

recommended development of knowledge base as a result of Genetic Algorithm application can be utilized for further investigation for identification of attribute which contribute for accurate classification of attack. A reliability result of 93.45% for normal and 94.19% for attack types were achieved for correctly classifying using 2000 iteration. Khan [**61**] has not included U2R and R2L attacks which can cause serious damage therefore, it lacks identifying these attacks.

ZhSong et al. [**62**], in their research presented an intrusion detection model based on neural network and expert system. They aimed at taking advantage of classification abilities of neural network for unknown attacks and the expert-based system for the known attacks. KDDcup'99 data is employed for training and test the feasibility of their proposed neural network component. They achieved 96.6% detection rate for DOS and probing attacks and 0.04% of false alarm rate. They pointed out that expert system can detect U2R and R2L attacks more accurately than neural network and concluded a hybrid model improves the performance to detect intrusions. But the study of ZhSong et al. [**62**] lacks in integrating neural network classifier with the expert based-system. They simply used the two separately and compared their performance.

A number of local researches have been done in designing and developing knowledge based systems in the health sector. But most recently Solomon [**63**] attempted to design and develop a prototype self learning knowledge based system that can provide advice and for physicians and patients to facilitate the diagnosis and treatment of diabetic patients. In this research knowledge is acquired using structured and unstructured interviews from domain experts. In addition relevant document analysis method is also followed to capture explicit knowledge. His experimentation result shows that the knowledge based system has achieved 84.2% overall performance.

In the area of Geographic Information System and Remote Sensing Huang and Jensen [**64**] studied a machine learning approach to automated building of knowledge bases for image analysis expert system incorporating Geographic Information System data. They used decision tree and production rule machine learning algorithms to automatically

build a knowledge base for a remote sensing image analysis expert system. The proposed machine learning approach yielded an overall accuracy of 74.46%. The researchers concluded that building a knowledge base for a rule-based expert system for remote sensing image analysis with Geographic Information System data is easier than using conventional knowledge acquisition approach. This work motivates to diversify machine learning knowledge acquisition for building knowledge based system.

The above local researches by Kindie [**56**], Zewdie and Tigabu used data mining techniques and merely generated model which lack utilizing the extracted knowledge for further action. This research work is not restricted to merely extracting hidden knowledge using data mining techniques form KDDcup'99 intrusion dataset; rather it adds value to the extracted knowledge by integrating it with knowledge based system with a goal developing rule based intrusion detection and advising prototype knowledge based system. This study is unique from locally undertaken knowledge based system researches. The first one is the use of automatic knowledge acquisition techniques, effected via data mining techniques, for developing knowledge based system. In addition, the prototype knowledge based system designed by Solomon [**63**] has learning capability which only updates facts but not rules. In this study, an attempt is made to update both rules and facts of the knowledge base. The study contributes by motivating future local researchers to diversify an automatic knowledge acquisition techniques rather than the conventional one for developing knowledge based system in other fields of study.

CHAPTER THREE

Knowledge Acquisition using Data Mining

Knowledge can be acquired from domain experts (also called implicit knowledge) and from documents, which is codified knowledge (also called explicit knowledge). In addition, knowledge can also be acquired from large collection of dataset by using knowledge discovery tools. This type of knowledge is called hidden knowledge. Data mining is proven to extract hidden knowledge from large collection of data by employing different mining techniques. In this study, which focuses on designing and developing knowledge base system for intrusion detection, data mining specially classification algorithms are employed to generate rules for the knowledge base. For rule induction, classifier algorithms such as J48, REPTree, PART and JRip are employed and their result is compared to generate rules for the knowledge base system.

Knowledge acquisition is a complex and time-consuming stage during knowledge based system development [**18**]. Induction tools and knowledge discovery tools can be used to process data in large volume database to generate knowledge base (in the form of rules). The critical component of knowledge based system is the knowledge base which contains facts and rules. Knowledge representation schemes such as frames, semantic rules, and rule-based systems exist to represent knowledge. However, rule-based systems are most common in many knowledge based systems [**48**].

Traditional knowledge acquisition techniques including on-site observation, protocol analysis, structured and unstructured interviewing and others can be used. But there are significant problems with each of these techniques. None of them guarantees consistency and integrity in the knowledge base [**48**]. Some of the problems are:

- They are labor intensive
- These techniques are expensive to implement.
- Underestimating past events
- Expert conservatism and unwarranted biases

- Inability of the expert to explain rules for decisions

Due to the aforementioned problems knowledge engineers look for other means to expand rule set and verify the rules already in the knowledge base. As a result [**18**] and [**32**] stressed the need for developing an automated techniques for knowledge acquisition. Considering the limitations mentioned above for acquiring knowledge from experts using traditional knowledge acquisition techniques, data mining techniques are used for the development of the knowledge base.

3.1 Data Selection and preparation

The dataset for this study is collected from KDDcup'99 dataset available at ACM Knowledge Discovery [**20**]. KDDcup'99 is a dataset collected from simulated network connection and made available for researchers to conduct study towards intrusion detection.

According to [**20**], the 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The 1999 KDD intrusion detection contest uses a version of this dataset. Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

Originally for this study 1,048,575 instances were collected from the website and among them 614,447 instances were found non-redundant. The table 3-1 shows the distribution of the instances before and after removing duplicates.

As mentioned by Mahbod et al [**65**], One of the important limitation raised on KDD data set is the large number of redundant instances. These redundant instances causes the classifier algorithms to be biased towards the frequent instances and consequently prevent them from learning unfrequented records (U2R and R2L) which are more harmful type network attacks. Thus in the data set redundant instances are removed from the experimental data set.

Table 3-1 Distribution of normal and attack instances

Before removing duplicates		**After removing duplicates**	
Normal	595,797	Normal	559,276
Attacks	452,778	Attacks	55,171
Total	1,048,575		614,447

3.2 Data Reduction and Processing

Before the actual mining task is commenced, data reduction is one of the tasks in data mining. Clustering and sampling are ways of reducing data size. In the dataset the attack type are 21 (see Appendix I). According to [**20**]and [**66**] there are 21 types of network attacks and normal instances in the data set in which they are grouped into five classes or clusters namely; normal, probe, DOS, U2R, and R2L(see Appendix I).

Since 614,447 instances are huge for processing to the classifier and it takes time for processing, the researcher re-sampled the dataset into 35,778 instances. The size of U2R and R2L is much smaller as compared to the others. Therefore, the original number of instances of U2R and R2L are included in the sample dataset. The number of instances for normal, probe and DOS in the sample is based on their percentage of share in 614,477 instances. Therefore, the number of normal, probe and DOS instances in the sample

35,778 is based on the ratio they have in 614,477. The distribution of the instances is shown in the table 3-2.

Table 3-2 Proportion of the sample instances for each attack types

Attack type	Number of Instances	Percentage
Normal	22,000	61.4%
Probe	5775	16.1%
DOS	7892	22.05%
R2L	102	0.2%
U2R	9	0.0002%
Total	35,778	

After preparing the required sample intrusion dataset, it is converted into Comma delimited Excel file (CSV format), then to the *arff* file suitable for mining using WEKA 3.6.8.

3.3 Experimentation

Having finished the data preprocessing and preparation task in format suitable for WEKA data miner, the next task is undertaking the experiment involving the selected classifier algorithms.

3.3.1 Experimental set up

A total of four experiments aiming at building predictive models are undertaken. The sampled data set contains 35,778 instances containing normal, probe, DOS, U2R and R2L. The data set contains 42 attributes and all of them are involved in all experiments. In addition after undertaking a number of experiments, default value of parameters is taken into consideration for each classifier algorithm since it allows achieving better accuracy compared to modifying the default parameters' values. The parameters involved with their respective description and values are indicated in table 3-3.

Table 3-3 Default parameters and values for algorithms

Parameters	Description	Default values			
		J48	REPTree	JRip	PART
binarySplit	Tells about whether to use binary splits on nominal attributes when building the trees	False	-	-	-
minNumObj	The minimum number of instances per leaf	2	-	-	-
useLaplace	Whether counts at leaves are smoothed based on Laplace.	False	-	-	-
minNum	The minimum total weight of the instances in a leaf	-	2.0	-	-
noPrunning	Whether pruning is performed or not	-	False	-	-
checkErrorRate	Whether check for error rate >= is included in stopping criterion			true	-
minNo	The minimum total weight of the instances in a rule			2.0	-
optimization	The number of optimization runs			1	-
usePrunning	Whether pruning is performed				
binarySplits	Whether to use binary splits on nominal attributes when building the partial trees.				false
minNumObj	The minimum number of instances per rule.				2
unprunned	Whether pruning is performed.				false
seed	The seed used for randomizing the data when reduced-error pruning is used.	1	1	True	1
debug	If set to true, classifier may output additional info to the console.	False	False	False	False
numFolds	Determines the amount of data used for reduced-error pruning. One fold is used for pruning, the rest for growing the rules.	3	3	3	3
reducederror Prunning	Whether reduced-error pruning is used instead of C.4.5 pruning.	False	-	-	False
folds	Determines the amount of data used for pruning. One fold is used for pruning, the rest for growing the rules	3			

3.3.2 Creating Predictive model

At this stage, 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. All the four classifiers are capable of generating rules. The first two experiments experiment 1 and 2 are decision tree classifiers and the next two experiments (experiment 3 and 4) are rule classifiers.

3.3.2.1 Experiment 1- J48 classifier

This experiment is undertaken using J48 classifier involving its default value of parameters. The parameters included and their respective default value is indicated in the table 3.3. 10-fold cross-validation is selected as test option. Hence, the model developed using J48 classifier has a tree of size 205 and number of leaves 170. The algorithm has correctly classified 35,732 instances and only 46 instances are classified incorrectly taking 9.37 seconds to build the model. Among the 42 attributes in dataset the algorithm used 22 attributes for generating the tree and has determined dst_bytes attribute as the most important for creating the tree.

3.3.2.2 Experiment 2- REPTree classier

The second experiment is based on REPTree algorithm. This experiment has involved the default parameters with respective values and 10-fold cross-validation test mode. Then, REPTree has correctly classified 34,646 instances out of 35,778, which means it has incorrectly classified 132 instances taking 3.57 seconds to build the model. In addition, the size of the tree generated is 192.

3.3.2.3 Experiment 3-PART classier

In this experiment PART rule induction algorithm is employed. It generated 25 rules by involving all the attributes of the dataset and 10-fold cross-validation test option. The algorithm registered prediction accuracy of 99.8798% in which 35,735 instances out of 35, 778 are correctly classified. The algorithm has incorrectly classified only 43 instances by taking 12.77 seconds to build the model.

3.3.2.4 Experiment 4 – JRip classifier

The other rule induction algorithm selected for this study is JRip. Therefore, to generate IF-THEN rules from the experimental intrusion dataset JRip algorithm with its default values of the parameter (see table 3-3) and 10-fold cross-validation test mode is employed. JRip correctly classified 35,737 instances from 35,778. The numbers of incorrectly classified instances are 41. The algorithm has generated 23 rules.

As mentioned earlier, two decision tree and two decision rule induction algorithms are used for the experiments. All the selected algorithms allow generating rules from the data set. The results of the algorithms are evaluated based on prediction accuracy in classifying the instances of the data set into normal, probe, DOS, R2L and U2R.

As indicated in the table 3-4, the classifiers performed almost the same. There is a slight difference among the classifiers in terms of classifying the data set correctly.

Table 3-4 Performance of Classifiers

Classifier	**Correctly classified instances**		**Incorrectly classified instances**		**Time take to build the model(in second)**
	No.	**percentage**	**No.**	**percentage**	
PART	35,735	99.8798%	43	0.1202%	12.77
JRip	**35,737**	**99.8854%**	**41**	**0.1146%**	**44.1**
REPTree	35,646	99.6311%	132	0.3689%	3.59
J48	35,732	99.8714%	46	0.1286%	9.37

Even though their slight difference, JRip has registered the best prediction accuracy by classifying 35,737 instances out of 35,778 correctly. Results of PART, JRip and REPTree show that nearly equal number of incorrectly classified instances. The highest incorrect classification is registered by REPTree algorithm. Table 3-5 depicts the confusion matrix for best performing classifier.

Table 3-5 Confusion matrix for JRip classifier

	Classified as					
	Normal	**Probe**	**DOS**	**U2R**	**R2L**	
Actual class	21,985	7	6	1	1	**Normal**
	9	5762	3	0	1	**Probe**
	4	3	7885	0	0	**DOS**
	3	0	0	6	0	**U2R**
	2	1	0	1	98	**R2L**

Prediction accuracy shows us the general classification accuracy of the algorithms. Apart from prediction accuracy, classifiers are also evaluated to measure how they correctly classified each class to their correct class or incorrectly classified to another class. Hence, to evaluate the performance of the classifiers employed in this study True Positive rate, Precision, Recall and F-measure are used. Table 3-4 illustrates the performance of the four classifiers. JRip has registered the best result in terms of the precision, recall and F-measure values as compared to other classifiers all over the five classes.

Furthermore, the True Positive rate of the classifiers is also compared. Table 3-6 also illustrates that TP rate of the algorithms for detecting normal, probe and DOS is almost similar. JRip has registered the highest (95.1%) TP rate for R2L attack class and PART has registered 85.3% TP rate for U2R attack class.

Table 3-6 Precision, Recall and F-measure of classifiers with respect to classes

Classifier	**class**					
PART		Normal	Probe	DOS	U2R	R2L
	Precision	99.9%	99.8%	99.9%	94.6%	55.6%
	Recall	100%	99.8%	100%	85.3%	55.6%
	F-measure	99.9%	99.8%	99.9%	89.7%	55.6%
	TP Rate	99.9%	99.8%	100%	85.3%	55.6%
JRip	**Precision**	**99.9%**	**99.8%**	**99.9%**	**98%**	**75%**
	Recall	**99.9%**	**99.8%**	**99.9%**	**95.1%**	**66.7%**
	F-measure	**99.9%**	**99.8%**	**99.9%**	**96.5%**	**70.6%**
	TP Rate	**99.9%**	**99.8%**	**99.9%**	**66.7%**	**95.1%**
REPTree	Precision	99.9%	99.7%	99.1%	89.6%	75%
	Recall	100%	99.9%	99.9%	84.3%	66.7%
	F-measure	99.9%	99.5%	99.9%	86.9%	70.6
	TP Rate	99.8%	98.3	99.7%	66.7%	84.3%
J48	Precision	99.9%	99.9%	99.9%	93.9%	33.3%
	Recall	100%	100%	100%	75.5%	22.2%
	F-measure	99.9%	100%	100%	83.7%	26.7%
	TP Rate	99.9%	99.8%	100%	22.2%	75.5%

With regard to FP rate, PART, JRip and J48 have registered almost similar values for normal, probe and DOS classes. JRip has registered the least FP rate (4.9%) for R2L class as compared to the other three algorithms. Moreover, for U2R class the least FP rate registered is by PART algorithm. The graphical representation of the algorithms with respect to classes for TP is indicated in Figure 3-1.

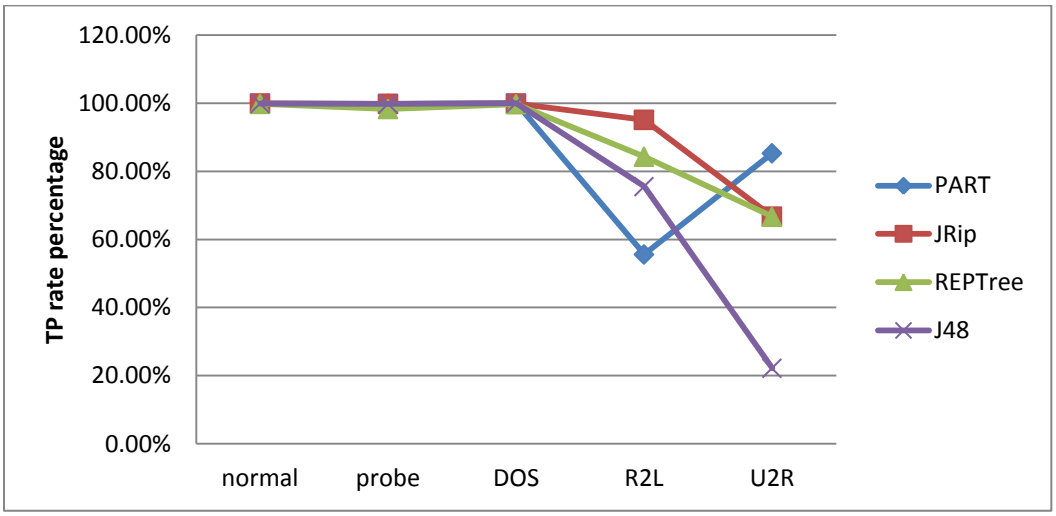

Figure 3-1 TP rate of classifiers

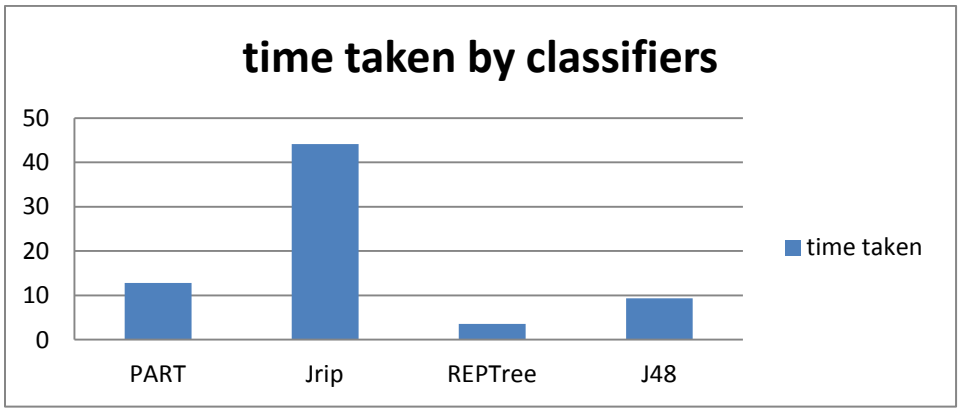

Figure 3-2Time taken to build model by classifiers

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.

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. However, as shown in the above figure 3-3 the model took much time than other.

JRip classifier has generated 23 rules shown in table 3-8. 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.

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 42 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.

Table 3-8 Rule set Generated Using JRip from sampled data set

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)	
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)	
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)	
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)	
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)	
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	=> class=normal (21999.0/4.0)	normal

CHAPTER FOUR

Integration of Data Mining Results with Knowledge Based System

The aim of this study is integrating data mining result for the development of knowledge based system. It is obvious that, knowledge base is the core for certain knowledge based system [**52**]. For that knowledge acquisition is done using JRip rule induction algorithm, which performs best for the given KDDcup'99 network intrusion dataset. The challenge here is how is it possible to integrate data mining and knowledge base system? Or how is it possible to use the hidden knowledge extracted from data mining for knowledge based system? The subsequent sections discuss the nuts and bolts of this issue.

4.1 System Design and Architecture

Figure 4-1 shows the overall system design and framework of integrating data mining induced hidden knowledge about network attacks and their types based on KDDcup'99 dataset with knowledge based system.

The Framework shows that the data mining tasks used for generating knowledge from large collection of dataset. Then following the validation of rules, the generated knowledge (rule set) is encoded to the knowledge base. The detail of the framework is discussed below.

KDDcup'99 data set: - is data set which has been in use since 1999 for evaluation of anomaly detection methods. The dataset is built based on the data captured in DARPA'98 Intrusion detection System evaluation program. KDD training data set consists of nearly 4.9 million single connection vectors each of which contains 41 features and is labeled as either normal or an attack such as probe, DOS, R2L and U2R [**65**].

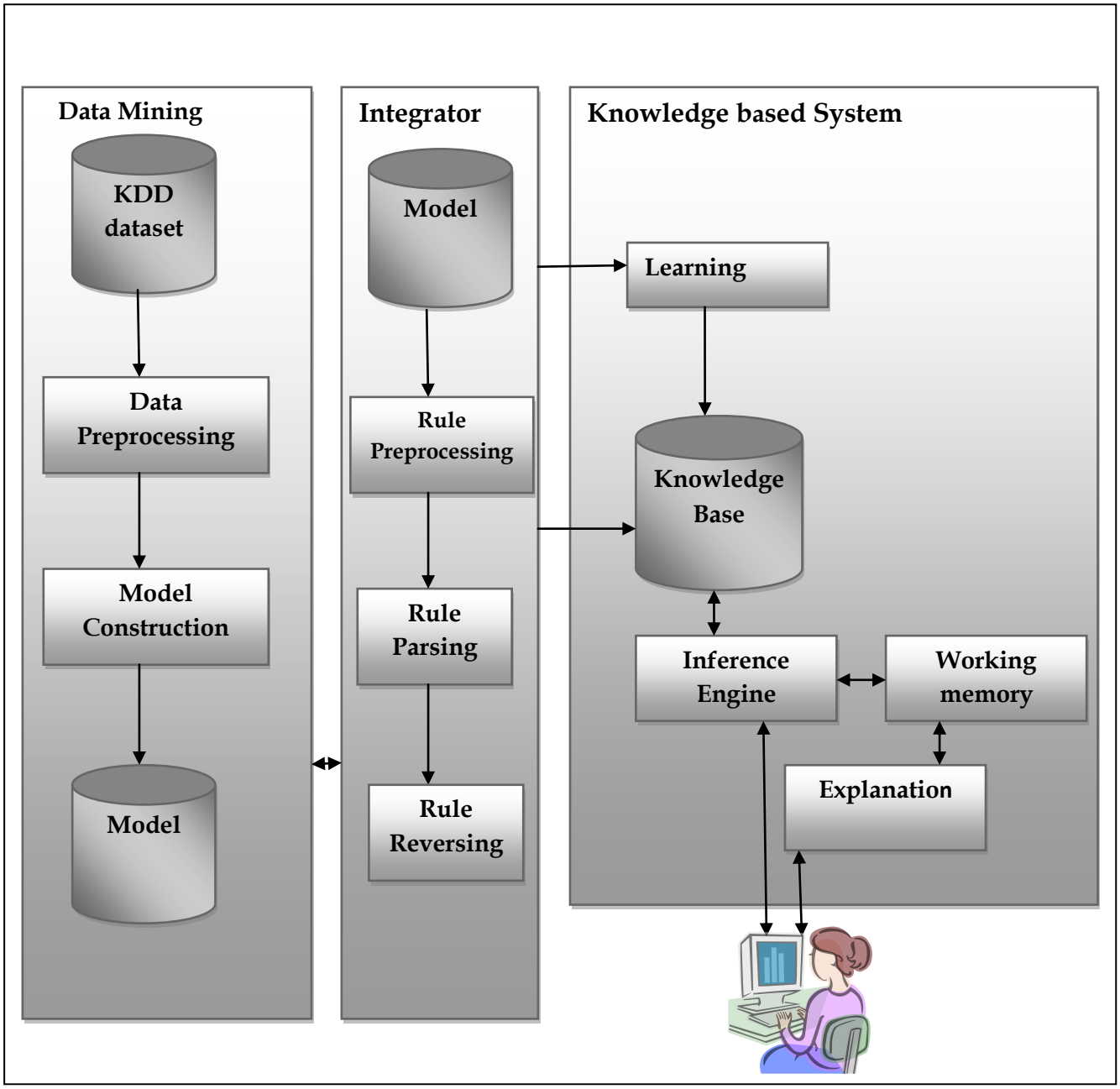

Figure 4-1 General Framework of Integration of data mining model with knowledge based system

Data preprocessing: - According to Han **[10]**, real world databases are prone to noisy, missing, and inconsistent data due to their huge size and their likely origin from multiple sources. Low quality data will lead to low-quality mining results. The KDDcup'99 dataset, which is used for mining knowledge for this study, has inherent problems in which the most important one is the existence of redundant instances **[65]**. 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.

Sampling data set: - KDD dataset 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 to generation or extraction of knowledge from the dataset and considerable samples are taken. While sampling, all instances of R2L and U2R are purposively 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.

Rule induction: - for creating predictive model that classify instances in to labeled classes, rule induction algorithms are used. In this study, algorithms such as J48, PART, JRIP and REPTree which are capable of generating rules are selected and employed for the mining task.

Knowledge validation:- induction algorithms generate knowledge in the form of rules from the data set. These rules should be validated to make sure that they are reflection of the dataset. Attributes or combination of attributes together form the rules. These rules should be evaluated so as to make sure that the attribute(s) and values of the attributes represent or reflect the conclusion.

For this study a number of rules are generated by the algorithm to identify an instance of the KDD dataset as normal, probe, DOS, U2R and R2L. For that, most rules used combination of attributes and few of them used a single attribute with the respective values for attributes. Therefore, before using the generated rules as part of the knowledge base, the rules are evaluated in consultation with domain experts in the area of network and data communication.

Knowledge Base:- 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.

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, Graphical User Interface for the integrator and Command Line Interface for the knowledge based system.

4.2 Automatic Integration of Data Mining Model with Knowledge base

In this study, an attempt is made to design an automatic integration of the result of data mining model with knowledge base. To achieve this Java NetBeans IDE 7.3 with JDK 6.0 and PROLOG has been used. This is done by understanding the standard format followed by JRip rule construction and PROLOG formalism.

4.2.1 Structure of JRip rule and PROLOG rule

JRip algorithm generates rules in the form of (condition)…(conclusion) format. The algorithm has generated 23 rules from the sampled KDDcup'99 data set. The format of the some of the rules is indicated in the table 4-1**.** 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`.

Table 4-1 Sample JRip rules for R2L and probe attack.

No	Rule	
	Condition	conclusion
1	`(is_guest_login = 1) and (duration <= 1) =>`	`class=R2L (2.0/0.0)`
2	`(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`
3	`(dst_host_rerror_rate >= 0.07) and (dst_host_same_srv_rate <= 0.81) =>`	`class=Probe (1611.0/0.0)`

As shown in table 4-1 the rules are in IF….THEN format. For example:

```
(is_guest_login = 1) and (duration <= 1) => class=R2L (2.0/0.0)
```

This rule can be read us:

IF `(is_guest_login = 1) and (duration <= 1)` THEN `class=R2L (2.0/0.0)`

The attribute `is_guest_login` is either 0 or 1. It tells us about whether guest logged into the system or not. If the guest is logged in, its value is 1; otherwise its value is 0. The `duration` attribute is the length (number of seconds) of the connection. It takes continuous value and that value tells us for how long (in seconds) the connection stayed.

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.

But 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 which can proof the goal as true. Therefore, the above rule has to be formatted as:

```
attack (u2r):- (is_guest_login = 1),(duration =< 1).
```

As illustrated above, the conclusion comes first with predicate 'class' and followed by ':-' replacing '=>' in JRip and then antecedents joined by ',' replacing 'and' in JRip rule. And finally PROLOG rules terminate by period (.) whereas JRip rules terminate with new line. Table 4-2 shows the content structure difference between prolog and JRip rule.

Table 4-2 Tokens in JRip and PROLOG rules

Token	JRip tokens	PROLOG equivalent	Tokenization option
Special character	'(', left brace ')', right brace	Same	Replace by empty space
Comparison operator	>,<,=,>=	same	Keep them
	<=	=<	Replace by '=<'
Logical operator	AND	,	Replace by ','
	=>	:-	Replace by':-'
Miscellaneous words and characters	JRIP rules: , normal (21998.0/3.0)., Number of Rules : 27 '==========' (10 equal to signs)		Replace by empty space
	class=		Replace by 'attack'

4.2.2 High level Conceptual Design of Integration Process

In section 4.2.1 it is discussed that JRip generated rules are in the IF… THEN format and PROLOG format is in reverse order. To convert JRip rules following PROLOG syntax, there is a need to design an integrator interface as shown in figure 4-2.

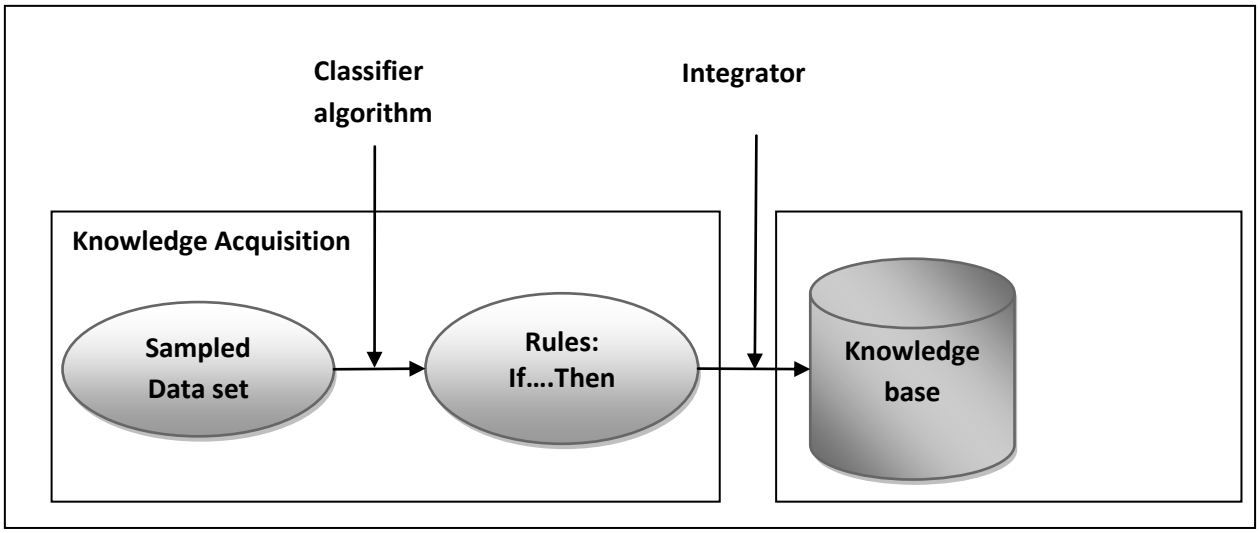

Figure 4-2 conceptual design of the integration process

Figure 4-2 is high level conceptual design for the integration of rules generated using data mining algorithm to knowledge base following PROLOG structure. The knowledge acquisition process is for acquiring knowledge from sampled intrusion data taken from the KDDcup'99 data set.

The work flow in Figure 4-3 shows tasks undertaken in the course of integrating JRip generated rules with knowledge based system.

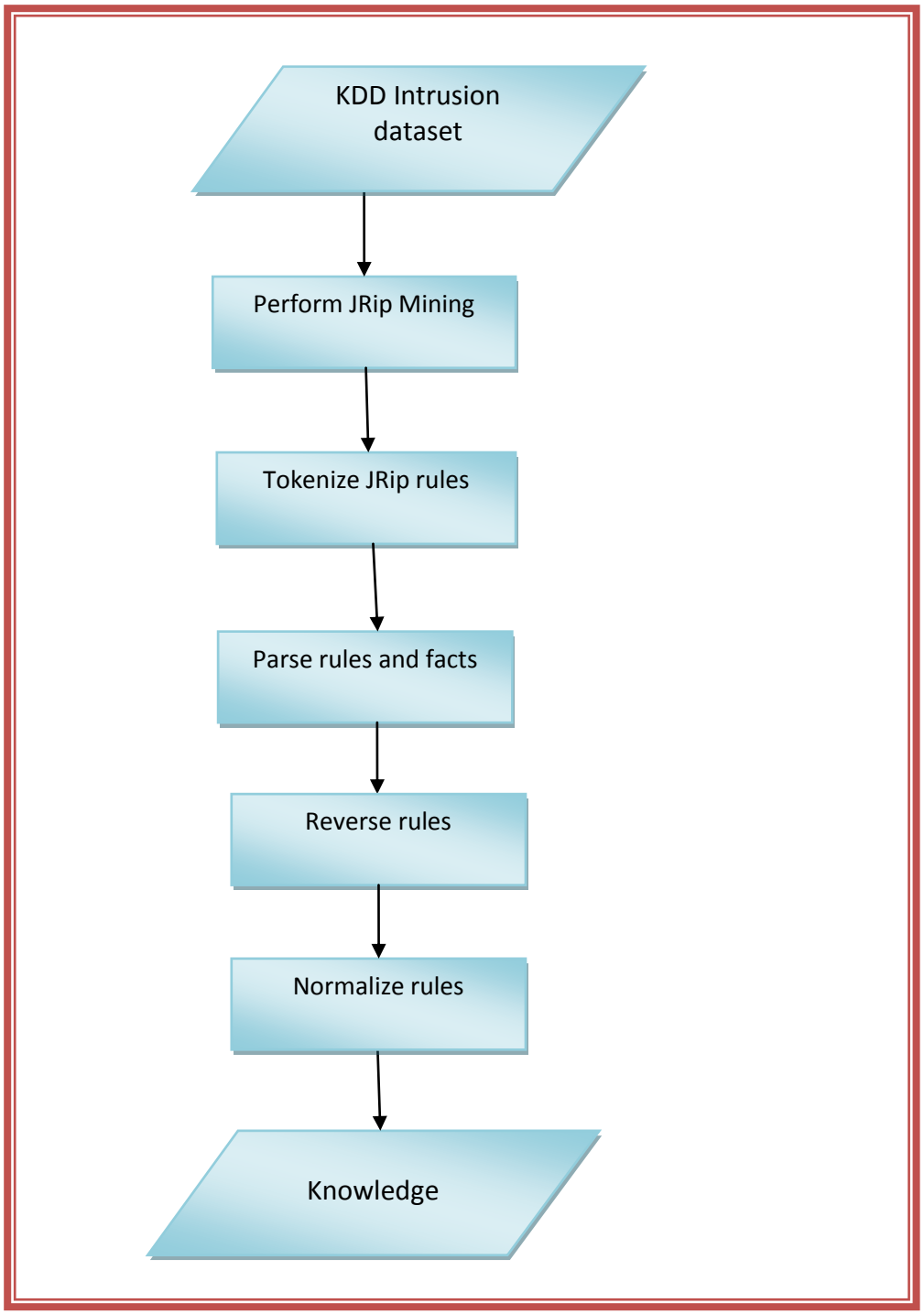

Figure 4-3 Work flow diagram for Rule mapping from JRip into prolog format.

Perform JRip Mining:- at this stage rules are extracted from the dataset involving JRip classifier.

Tokenize rules: - a given JRip rule contains special characters, attributes, comparison operators and logical operators. The tokenization process focuses on removing characters which are undesirable and replacing some tokens with other tokens. Some special characters are replaced by empty space and some others are replaced by another character bearing some meaning. For example, comparison operator '<=' (less than or equal to) in JRip rules is replaced by its PROLOG equivalent '=<'.

The conjunction operator 'and' is replaced by its PROLOG equivalent ',' bearing same meaning and function for joining two conditions. The '=>' is replaced by ':-'which means *IF* in PROLOG. In addition, the token 'class=' is replaced by 'attack' to make it predicate for head of rules. The tokens 'JRIP rules:', '==========' (10 equal to signs) which are at the beginning of rules has no relevance for the desired mapping of JRip rules to PROLOG rules.

At the end of JRip rule set, the tokens 'Number of Rules: 27', 'normal (21998.0/3.0).' and white spaces around the beginning and end or JRip rules again has no relevance then they are removed from the rules. Table 4-2 shows tokens and tokenization options. After understanding the structure of JRip rule, the algorithm depicted in figure 4-4 is designed to undertake the rule tokenization process.

Parse rules and facts. is the process of analyzing a string of symbols, either in natural language or in computer languages, according to the rules of a formal grammar [**67**]. In this research context, parsing is analyzing the components of JRip rules. A given JRip rule is composed of: (*condition*) implication (*conclusion*). If *condition* is evaluated true then *conclusion* is executed. Figure 4- 6 shows the structure of JRip rule.

```
function tokenizer(line)
        loop i=0 till length_of_line-1
         if length of line >0
              if token is in ['(',')', '===========',' JRIP',' rules:',' Number of
                                  Rules : ','normal(digit/digit)'] then
                             remove token
                  else if token='and' then
                             replace it by comma(,)
                  else if  token is '=>' then
                             replace it by ':-'
                  else if token is 'class=' then
                             replace it by 'attack'
                  else if token has the format '(digit/digit)'  at the end then
                              remove it
                  end if
end if
end loop
```

Figure 4-4 Algorithm for rule tokenization

The *condition* part is also divided into one or more conditions. In case there are two or more conditions, they are connected by logical operator (AND). A condition is composed of an *attribute*, *comparison operator* and *value*. An *attribute* is a property or characteristic describing about something for example, duration, service, flag etc are attributes describing a certain network incident. *Comparison operator* is used for comparing an attribute with value which can be number or string. For example, 'is_guest_login = 1', 'duration <= 1' and 'dst_host_rerror_rate >= 0.07', service='http'. Figure 4-5 illustrates algorithm designed to parse components of JRip rule.

Reverse rule: - the reverse rule stage is used to exchange the place of Left Hand Side (LHS) of the rule and Right Hand Side (RHS) of rule. One can manually reverse the right hand side of the rule to left hand side to come up the desired output. But this is so tedious and error prone especially when the rules are high in number.

```
function rule_parser(line)
        read rule
        RULE_COMPONENT=split rule by '=>'
        CONDITION= left_of '=>'
        CONCLUSION=right_of '=>'
        CONDITIONS [ ] =CONDITION split by 'and'
        Loop:
            i=0 up to number_of(and)
            ANTECEDENT[i]=CONDTIONS[i]
            ATTRIBUTE [i]= left_of_comparision_operator
            VALUE [i]= right_of _comparion_operator
        end loop
```

Figure 4-5 Algorithm for rule parsing

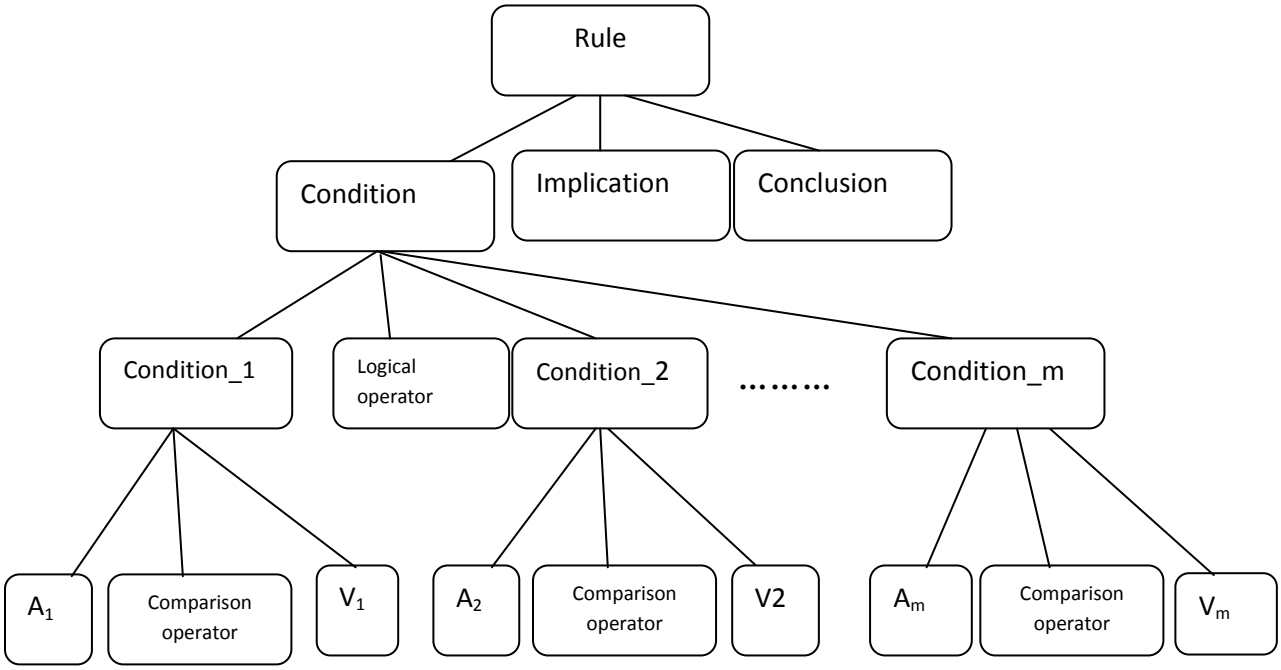

Figure 4-6 Parse tree for JRip rules.

The rationale for reversing here is due to the fact that PROLOG understands the reversed rule format. That means it starts with conclusion and goes for facts that make the conclusion true. Hence the JRip rule must be reversed from the format *(condition*) THEN (*conclusion*) to the format (*conclusion*) CONNECTOR (*condition*). For example:

Rule: (A_1=V_1) and (A_2=v_2)…, and (A_m=V_m) THEN (conclusion)

Reversed rule format: (conlusion):-(A_1=V_1) and (A_2=v_2)…, and(A_m=V_m).

```
Function rule_reversor(line)
  Loop :
     Iterate till end of line
          If token is the last token then
             If token_length not equal to zero then
              Reversed_rule=( token before the last_token and
                                  last_token  and
                                 third token from last_token )
                  break
               end if
            end if
   end loop
loop:
i=0  till LENGTH_OF_LINE-1
if token is not LENGTH_OF_LINE-3
       if token is at first position
                concatenate (Reversed_rule
                             token at first position and opening brace
                             token next to first position token
                             token at third position
                             closing brace
                             space)
        else
                 concatenate (Reversed_rule
                              comma(,)
                              token at first position and opening brace
                               token next to first position token
                               token at third position
                               closing brace
                              space)
                  increment i by 4
        else
                     concatenate (Reversed_rule, current_token
                       jump to next token
       end if
end if
end loop
```

Figure 4-7 Algorithm for rule reverser

After reversing, predicates are added for each rule. The word 'attack' replaced the token 'class=' from the JRip rule to make the predicate more meaningful. The head

'attack(probe)' is more meaningful than 'class(probe)'. In JRip rules the *conclusion* is name of network attacks. But for the conditions, their attribute name is used as predicate. Finally period (.) is should be placed at the end of all reversed rules to tell PROLOG end of statement. The algorithm for reversing rule is depicted in figure 4-7.

Normalize rules:- normalization stage is aimed at changing all tokens in reversed rule into lower case. The conclusions in JRip rule contain U2R, DOS, and R2L, which are in upper case. PROLOG understands a token which starts in or is totally in upper case as a variable. After reversing, the heads of rules should be in the format attack(u2r), attack (dos), and attack(r2l). Hence, the normalization step makes reversed rules in lower case format. The final output has general format shown below.

attack (conclusion):-$a_1(a_1=v_1),a_2(a_2=v_2),\ldots..,a_m(a_m=v_m)$.

4.3 Implementation of Discovered Rules to Knowledge Base Integrator

Having generated rules using JRip classifier, the next task is building or constructing the knowledge base. For this study we devised an automatic construction of knowledge base aligned with the data mining task.

In order to achieve automatic construction of knowledge base, integrator application is developed based on work flow shown in figure 4-3. The overall task of the application is to extract rules from sampled intrusion data set using JRip classifier and mapping JRip rules to PROLOG rules. So as to accomplish the mapping, the integrator is designed and implemented comprising of three modules. The modules are named as; jripMiner, factAndRuleGenerator and rulePreprocessor.

4.3.1 JripMiner module

This module is responsible for undertaking the mining task using the selected classification algorithm. So as to accomplish the mining task, weka.jar is embedded to NetBeans library. Weka.jar contains different mining algorithms including classification, clustering, association rules and others. In addition, it contains other classes used for preprocessing input data such as normalization, re-sampling, SMOTE, etc. This package

contains all the algorithms and preprocessing components that are available in the WEKA 6.8.1 explorer application.

For the integrator application, the weka.classifiers.rules is imported. Under the weka.classifers.rules rules generator algorithms such as DecisionTable, DecisionTableHashing, JRip, M5Rules, OneR, PART, Rule, RuleStates, ZeroR are included as subclass of the weka.classifiers.rules. The weka.classifers.rules.jrip is imported to the class to undertake mining involving JRip which is best performing classifier algorithm (discussed under chapter 3 section 3.4.2.4). The `BufferedReader` opens intrusion dataset file. Then instances are declared from the file read by `BufferedReade`. After that, `data.setClassIndex(data.numAttributes() -1)` makes the last attribute as class attribute. The string array `options` hold set of default option for JRip classifier as illustrated under table 3-3. The command `jrip.buildClassifier(data)` perform the mining task. Following successful completion of rule extraction, the rules are written to file.

4.3.2 rulePreprocessor

After the mining is completed, the result is written as text file (file_name.txt). The rulePreprocessor module is responsible for removing some special characters, removing unwanted tokens, replacing some logical operator by another logical operator and replacing comparison operator with another comparison operator. The replacement and removal of special characters, logical operator and comparison operator is based on the tokenization process illustrated in table 4-2.

Table 4-3 illustrates short listed example rules preprocessed by rulePreprocessor module. The table contains JRip rules after unwanted characters are removed and some characters are replaced by some PROLOG equivalent tokens. All the rules are preprocessed in the same fashion. Then factAndRuleGenerator module continues its task of reversing right hand side to left hand side from the tokenized rules.

Table 4-3 Rules before and after tokenization

NO	Before rule preprocessing	After rule preprocessing
1	(root_shell>=1)and (duration>=25)=>class=U2R (9.0/1.0)	**root_shell>=1,duration>=25:- attack U2R**
2	(num_failed_logins>=1)=>class=R2L(52.0/0.0)	**num_failed_logins >= 1:- attack R2L**
3	(service=imap4)and(dst_host_count<=11)=>class=R2L (11.0/0.0)	**service=imap4,dst_host_count=<11:- attack R2L**
4	duration>=12)and(dst_host_same_src_port_rate<=0)=> class=R2L (5.0/0.0)	**duration>=12, dst_host_same_src_port_rate=<0:- attack R2L**
5	(diff_srv_rate >= 0.37) and (count>=6)=>class=Probe (37.0/0.0)	**diff_srv_rate>=0.37,diff_srv_rate>= 0.37:-attack probe**
6	(count >= 49) and (dst_bytes <=0)=>class=DOS (6977.0/0.0)	**count >= 49,dst_bytes=< 0 :- attack DOS**

4.3.3 factAndRuleGenerator (Rule reverser) module

Once rules are cleaned by using rulePreprocessor module, the next step is integrating rules and facts following syntax of PROLOG for creating the knowledge base needed to enhance reasoning process. This requires reversing the order of the rules from IF…THEN construct to THEN…IF construct for backward chaining. Hence this module exchanges the position of the left hand side and right hand side of JRip rules. For example given JRip rule:

```
(root_shell >= 1) and (duration >= 25) => class=U2R (9.0/1.0)
```

After rule preprocessing it looks like: `root_shell >= 1, duration >= 25:- attack U2R.`

Here, "`root_shell >= 1, duration >= 25`" are the antecedents (left hand side of the rule) and "`U2R`" is the conclusion (the right hand side of the rule). According to Cook **[68]**, prolog rules have both head and body but facts has only heads. Hence, the module first builds the heads of the rules having the format: *'predicat(conclusion):-'* .

The module starts iterating from end of a given preprocessed rule. Then it brings the predicate 'attack' to the beginning and concatenates it with opening brace '(', then follows the conclusion (like U2R in the above preprocessed rule), then closing brace ')'. After that ':-'is concatenated which means *IF* in PROLOG. Therefore, the module finally yields heads such as; **attack (U2R):-, attack (probe):-, attack (DOS):-, attack (R2L):-**. So as to make it complete rule, the body part (antecedents) must be concatenated with the heads.

Predicates for antecedents are the name of the attributes in the rule. For example, given the condition `(root_shell >= 1)`, the attribute `root_shell` is used as a predicate and `root_shell (root_shell >= 1)` is constructed as antecedent or fact.

The body of the rule comes after ':-'. It holds one or more than one facts joined by comma ','. The statement `reverse+=n[i]+"("+n[i]+n[i+1]+n[i+2]+")"+ " "` concatenates antecedent at the beginning of a rule by making name of an attribute as predicate with the head of the rule. The statement `reverse+=","+n[i]+"("+n[i]+n[i+1]+n[i+2]+")"+ " "` is used whenever antecedent or condition of a rule is somewhere in the middle of the rule. This statement first places comma (,) to concatenate previous condition with the current one. Figure 4-8 depicts sample PROLOG rules constructed using the module.

In addition, factAndRuleGenerator module also generates facts. Each rule in the rule base is built as a combination of facts. Hence, while inferencing the inference engine refers to the facts and rules to decide whether a certain condition is true or false in its way of identifying a certain network incident as an "attacks" or "normal" behavior. Hence, fact bases are constructed in line with rule base.

```
attack(u2r):-root_shell(root_shell>=1) ,duration(duration>=25) .
attack(r2l):-num_failed_logins(num_failed_logins>=1) .
attack(r2l):-service(service=ftp_data) ,flag(flag=sf) .
attack(r2l):-service(service=imap4) ,dst_host_count(dst_host_count=<11) .
attack(r2l):-duration(duration>=12) ,dst_host_same_src_port_rate(dst_host
_same_src_port_rate=<0) .
attack(r2l):-num_access_files(num_access_files>=1) ,src_bytes(src_bytes=<
116) .
attack(r2l):-is_guest_login(is_guest_login=1) ,duration(duration=<1) .
```

Figure 4-8 Sample Prolog rules constructed by factAndRuleGenerator module

The statement `newFact+=n[i]+"("+n[i]+n[i+1]+n[i+2]+")"+ ".\n";` collects facts in each iteration. After joining the head of the rules with the body and at the end of all facts and rules period (.) placed to tell PROLOG this is end of statement. Then it normalizes the rule to lower case to avoid the consideration of tokens starting with upper case or which are totally in upper case as variable by PROLOG. The final output of this module is depicted in figure 4-9.

```
root_shell(root_shell>=1).
duration(duration>=25).
num_failed_logins(num_failed_logins>=1).
service(service=ftp_data).
flag(flag=sf).
service(service=imap4).
dst_host_count(dst_host_count=<11).
duration(duration>=12).
dst_host_same_src_port_rate(dst_host_same_src_port_rate=<0).
num_access_files(num_access_files>=1).
src_bytes(src_bytes=<116).
is_guest_login(is_guest_login=1).
duration(duration=<1).
src_bytes(src_bytes=<8).
```

Figure 4-9 Sample facts constructed by factAndRuleGenerator module

While diagnosing a certain network incident, serious of questions are displayed for the user to reply to it considering the attributes and its respective values for the network incident. PROLOG clauses are created which are used for interaction with the user by displaying questions for the user while using the system.

The statement `askerPl+=n[i]+"("+"X"+"):-ask("+n[i]+","+"X )"+".\n"` is used for implementing the question asker. The clause has the format `root_shell(X):-ask(root_shell,X).` Hence, such types of clauses are constructed for all the attributes

in the JRip rules. Figure 4-10 depicts sample final result of clauses used for user interaction.

```
service(X):-ask(service,X ).
flag(X):-ask(flag,X ).
dst_host_count(X):-ask(dst_host_count,X ).
dst_host_same_src_port_rate(X):-ask(dst_host_same_src_port_rate,X ).
num_access_files(X):-ask(num_access_files,X ).
src_bytes(X):-ask(src_bytes,X ).
is_guest_login(X):-ask(is_guest_login,X ).
dst_host_serror_rate(X):-ask(dst_host_serror_rate,X ).
dst_host_rerror_rate(X):-ask(dst_host_rerror_rate,X ).
```

Figure 4-10 Sample asker clauses constructed by factAndRuleGenerator module

After building rules, facts and asker clause (which allows interacting with the user and asks the user to enter yes/no answers for the questions asked), they are separately written by factAndRuleGenerator to files with "file_name.pl" formats so that it is possible to use the files by SWI-Prolog. Figure 4-8, figure 4-9 and figure 4-10 show the result displayed at SWI-prolog interfaces.

The whole program for developing the data mining and knowledge based system integrator using Java NetBeans and Prolog code for constructing is attached in Appendix VI and Appendix VII.

CHAPTER FIVE

Implementation and Experimentation

So far, knowledge is generated from sampled collection of intrusion data set and knowledge base is constructed automatically as rules and facts which can be parsed via selected knowledge representation tool SWI-PROLOG. The system, named as Rule Based-Intrusion Detection and Advising Knowledge Based System (RIDA-KBS), is capable of diagnosing a network incident as normal, probe, DOS, U2R andR2L. In addition, it provides advice for the user about the result of diagnosis. The knowledge base contains validated rules and facts about intrusions and normal behavior of network incidents.

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 question containing attributes with their values. The user is expected to reply for the questions or ask explanation for the questions.

Once RIDA-KBS detects network incident as one of probe, DOS, U2R and R2L attacks, it displays information about the incidents to advise the user in decision making. The decision made by the user is either to allow or deny the incident based on the rules and the displayed information.

5.1 Architecture of RIDA-KBS

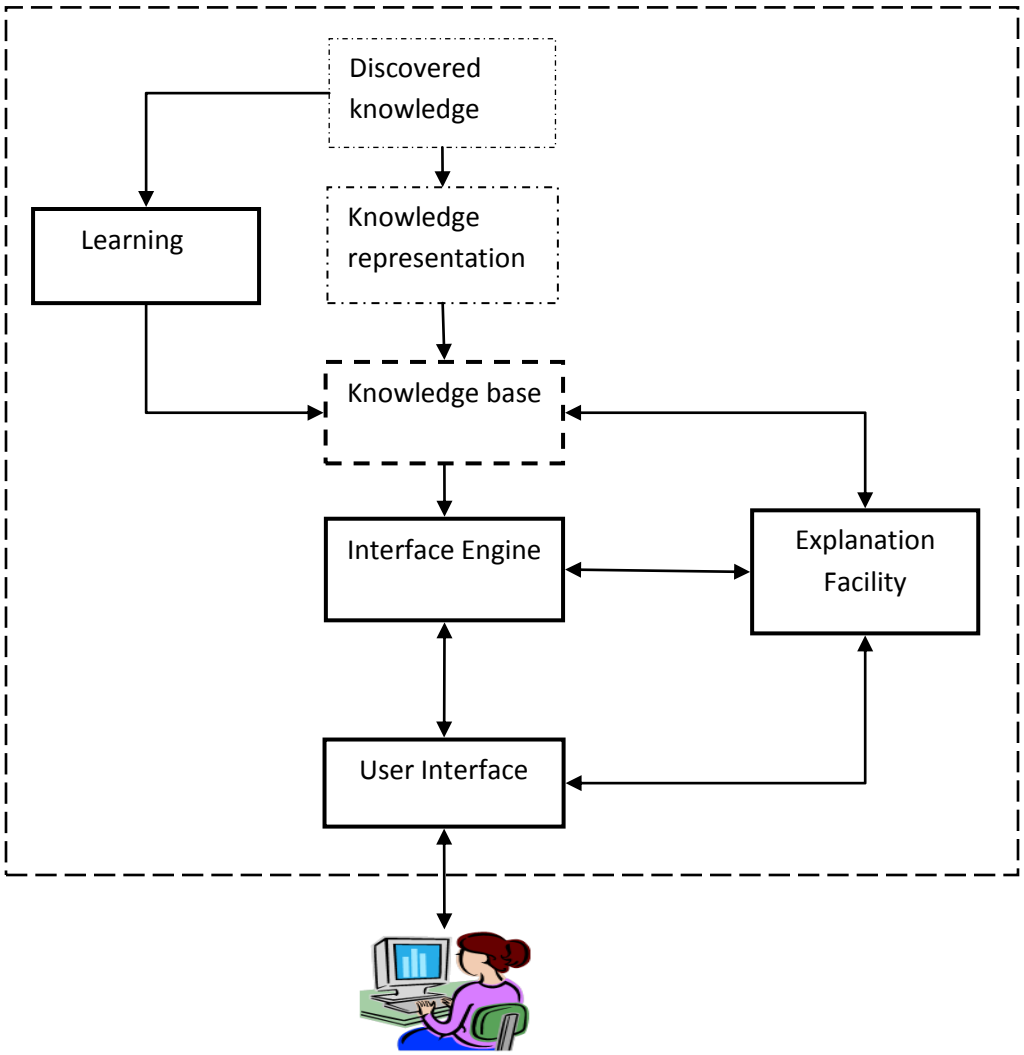

Figure 5-1 Architecture of RIDA-KBS

The architecture of Rule-based Intrusion Detection and Advising Knowledge Based System is shown in figure 5-1. Discovered knowledge (accomplished at knowledge acquisition stage), knowledge representation and knowledge base are drawn as dashed line to portray that they are previously undertaken. The learning component is included in the architecture. Learning is basically the capability of the knowledge based system to incorporate new rules or facts or both into its knowledge based system. The rules change, as the number of instances in the sampled data set changes. The designed knowledge

base must be able to accommodate these changes and use them in its diagnosis of network attacks. This accommodation of new rules by the KBS is learning.

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

Table 5-1 Modules of the RIDA-KBS

No	Module name	Description
1	Rule base	Is container of rules about network intrusion and normal behavior
2	Asker module	Is responsible for presenting questions for the user based on the rules and facts from the knowledge base
3	Attack description module	It gives description or explanation of attack types after detection. It gives general information, damages may caused by the detected attack, recommendation for prevention.

- **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. RIDA-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. A change in

the kind and number of attributes may be caused due to a change in the number of instances and the parameters selected during knowledge acquisition step (during mining employing the selected classifier).

- **Attack description module:-** this module is used following a certain incident is identified as an attack during interaction with the user. The aim of the module is to provide advice and information for the detected attack type. The module also provides recommendations for prevention of such attacks and action to take.

5.2 Network Attack Diagnosis

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. RIDA-KBS 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. For example figure 5-2 is set of sample rules and figure 5-3 is question and answer in RIDA-KBS provided as illustrations of the aforementioned statement.

```
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) .
```

Figure 5-2 Sample rules from the rule base

The above figure is list of three example rules from the rule base. While diagnosing, RIDA-KBS displays question by using body of the rules starting from Rule 1.

```
Is root_shell>=1 :?( what/yes/no) yes.
Is duration>=25: ?(what/yes/no) no.
Is num_failed_logins>=1: ?(what/yes/no) no.
Is service=ftp_data: ? (what/yes/no)  yes.
Is flag=sf:? (what/yes/no) yes.
The type of network attack is r2l (how)?
```

Figure 5-3 Question and answer in RIDA-KBS.

As shown in the above question and answer (figure 5-3), RIDA-KBS starts asking question from first rule. As indicated in figure 5-2 the first rule is a concatenation of two rules joined by and (,). Here the user replied 'yes' for the first question `Is root_shell>=1:?` Following that the next question `Is duration>=25:?` is displayed. This question asks the user the length of connection in seconds.

For Rule 1 to be evaluated as true and the attack type to be identified as U2R, the answer for the two questions should be 'yes'. Since the user replied 'no' for the second question, RIDA-KBS continues asking for the third question `Is num_failed_logins>=1`, which is Rule 2. This rule has only one attribute `num_failed_logins`; telling us about the numbers of failed logins registered for an incident. According to the rule base, the incident is diagnosed as R2L attack if it is greater than or equal to 1 and user replies 'yes' for the question. User replied 'no' therefore, RIDA-KBS proceeds asking the next question from Rule 3. The user replied 'yes' for the questions: `Is service=ftp_data:?` And `Is flag=sf:?.` The first question is about the type of service required from the coming incident and the second one is about flag. According to Rule 3 in rule base, the incident is identified as R2L attack then RIDA-KBS displayed "The type of network attack is r2l (how)?"

The aforementioned paragraphs showed how the diagnosis for network incident is performed. After identifying the type of attack RIDA-KBS has recommendation and information about the detected type of attack.

5.3 Explanation Facility

Explanation facility provides information to user for the questions asked by the RIDA-KBS. The explanation is helpful to have clarity while answering to the questions asked by the system. In its steps towards identifying incidents as an normal, probe,DOS,U2R andR2L, RIDA-KBS displays questions. The user can get explanation of the question asked if he/she has no clarity with it. For example: `Is src_bytes>=21048:?(what/yes/no)` is one of the question asked. The user can get explanation about `src_bytes` attribute by typing 'what'. As indicated in figure 5-4, the

explanation of src_bytes is displayed. It is the amount of data bytes sent from the source to destination. Then RIDA-KBS again asks the user for reply by displaying Is src_bytes>=21048:? (enter yes/no).

```
Is src_byte>=21048:?(what/yes/no)what.
Src_byte : is the number of data bytes from source to destination.
It is basic feature of individual TCP connection.
Is src_byte>=21048:?(enter yes/no)
```

Figure 5-4 Sample explanation facility

5.4 Recommendation for detected attacks

Having detecting the attacks, RIDA-KBS should be able to give advice for the user so as to assist what type of action a network administrator should take. In this case, RIDA-KBS delivers the network administrator the type of attack, description of attack, damages it may cause if it enters to the network and ways for prevention of the mentioned attack.

5.4.1 General Information provider

RIDA-KBS provides general information about the detected attack type. The information includes what the name of attack implies, how it attacks computers and network and its behavior. Figure 5-5 shows prolog code for displaying menu of R2L description.

```
describe_r2l:-write('What do you know about R2L,'),nl,
    write('Please enter your choice of action(1 up to 4'),nl,
    write('1. General Information'),nl,
    write('2. Damages caused'),nl,
    write('3. Prevention'),nl,
    write('4. Exit R2L description'),nl,
read(Reply),(Reply==1>general_info;Reply==2>damages;Reply==3>prevention,Reply==
4->exit_r2l).
```

Figure 5-5 Prolog code for menu of R2L description

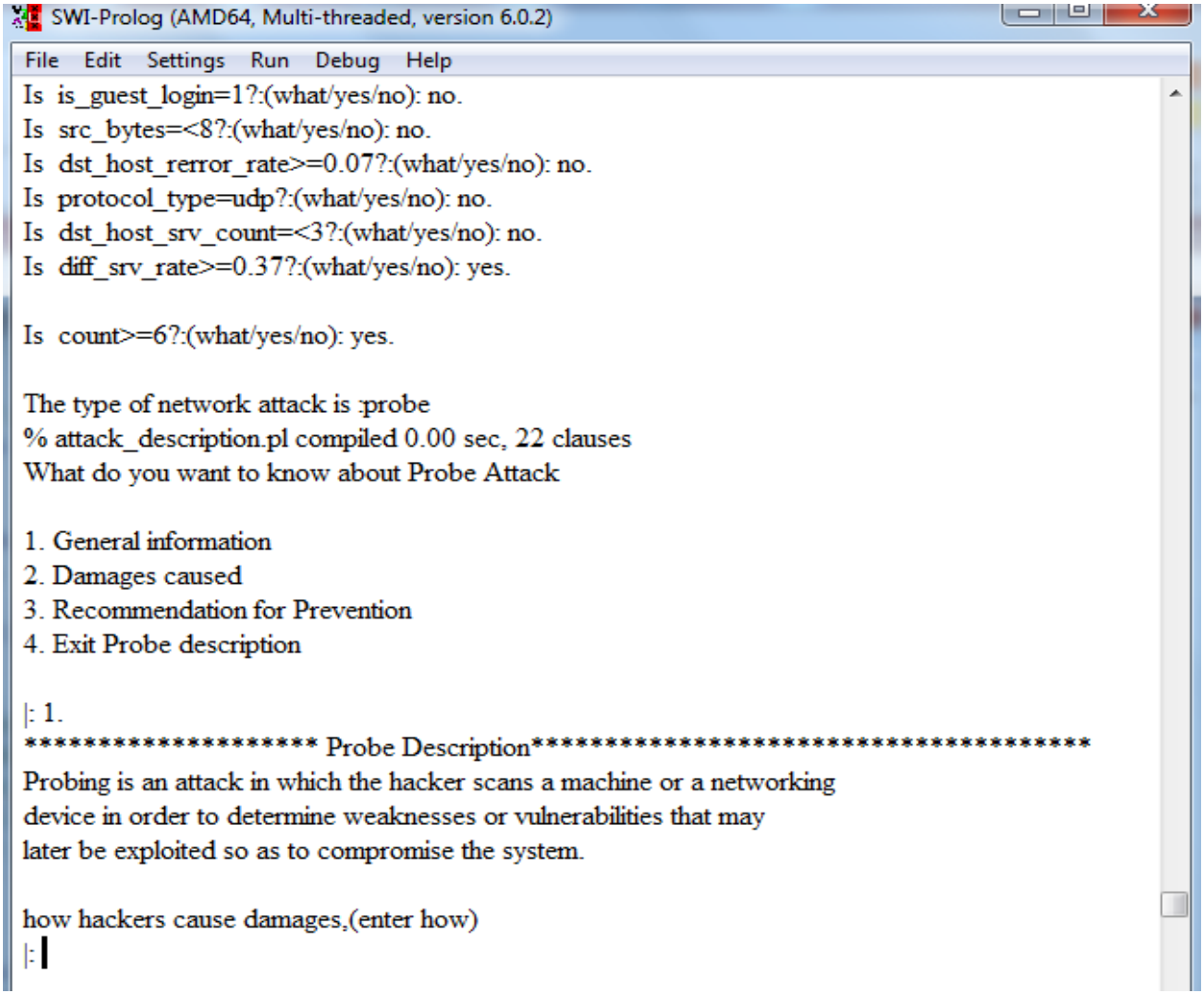

Figure 5-6 Prolog interface for description of probe attack in RIDA-KBS

RIDA-KBS displays menu of options for the user after identifying type of attack of an incident as indicated in the figure 5-6. The figure shows how RIDA-KBS identified the attack type and displayed menu of options for the user, in this case the user has entered a query for general information about the identified attack and the system displayed what probe attack mean. The user can also type 'how' to know more about how probe attack causes damage and create vulnerability to the network. Figure 5-7 shows how probe attacks causes damage and the short listed types of probe attacks.

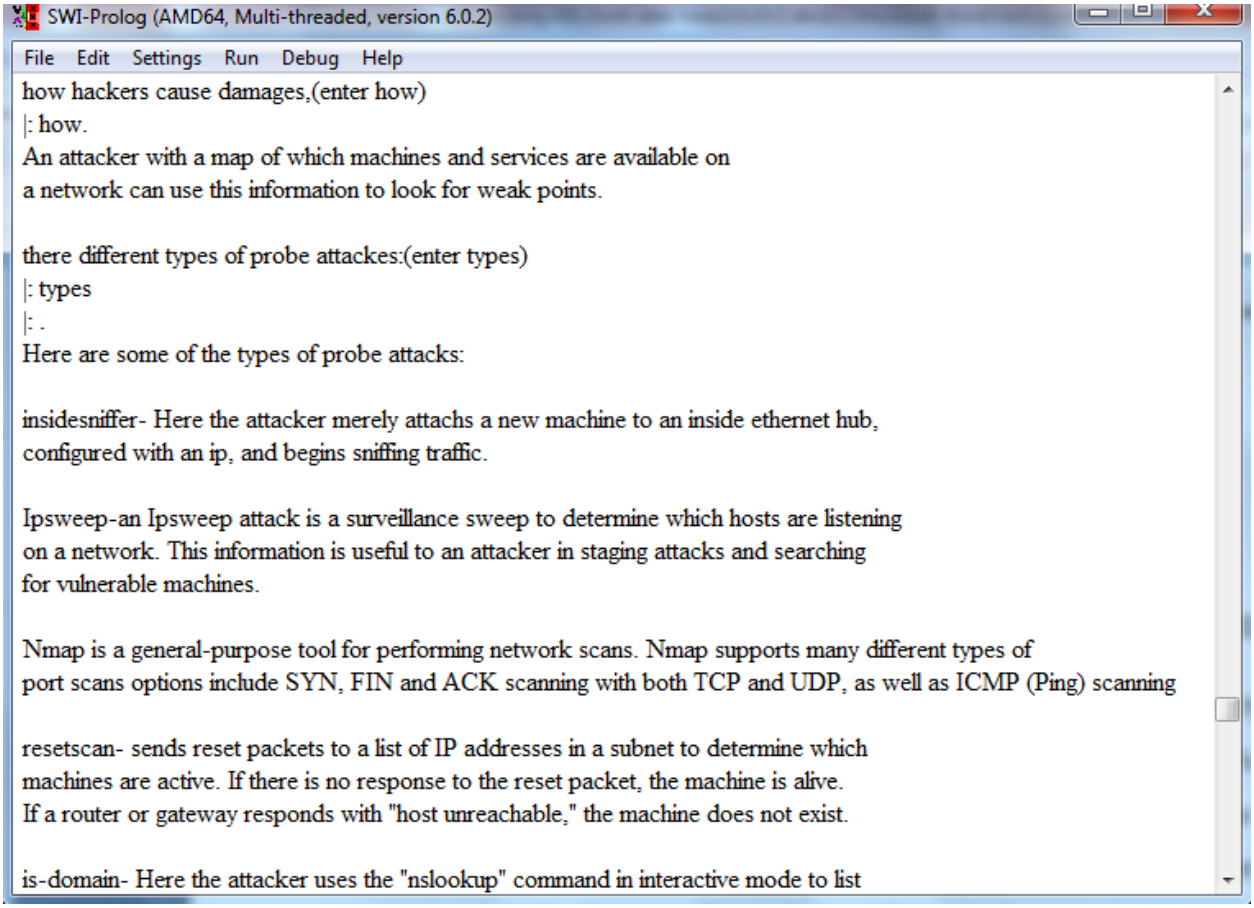

Figure 5-7 RIDA-KBS screen shot showing short listed probe attacks.

5.4.2 Recommendations and prevention

As mentioned in the previous section RIDA-KBS has identified the attack type and provided general information about the attack type. But this is not enough for the user in assisting what actions to take about the incident and prevent possible damages caused by the attack. Hence RIDA-KBS provides recommendations and prevention mechanisms for the detected attack type. Since attacks have different nature, different on the way they enter to the network and the damages they cause is also different, the knowledge based system should be able provide recommendations accordingly. The researcher referred MIT Lincoln Lab DARPA intrusion evaluation [**66**] to acquire codified knowledge about the description of each attack types, behavior of the attacks, recommendation for prevention of each type of attacks. Figure 5-8 indicates list of recommendation of RIDA-KBS for probe attack. The system recommends strengthening configuration of machines

and network devices and to watch out Ping requests. RIDA-KBS also provides this for DOS, R2L and U2R attacks.

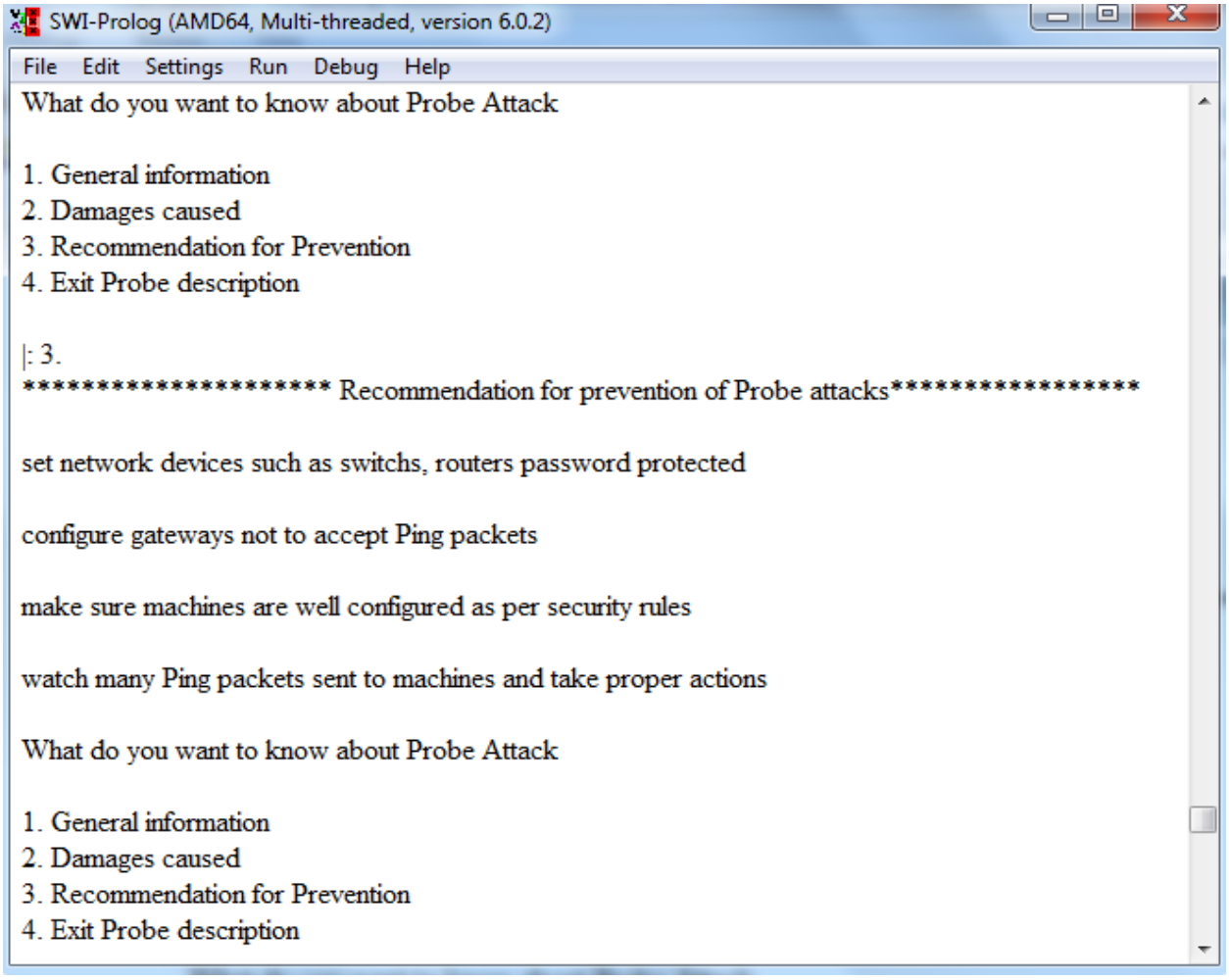

Figure 5-8 RIDA-KBS's screen shot for recommendation of probe attack

5.5 Testing and Evaluation of RIDA-KBS

In order to assure that RIDA-KBS meets the requirement it is developed for, it has to be tested. Testing is aimed at making sure compliance of the system with user expectation, human expert and system functioning [**69**]. It helps to answer the questions is this the right system? And is the system right? For evaluating the performance of the knowledge base system test cases are prepared and given to the system. The outputs of the system are compared against domain area experts' judgment.

User acceptance testing is undertaken following system performance testing basically focusing on testing the behavior of the knowledge based system to check that it is satisfactory in the eyes of the user. However accurate in performance measures is the system, how complete the knowledge based system is, it will be difficult if the system doesn't meet user requirements or is not accepted by users. User acceptance testing allows assuring the system's behavior in realistic environment. It does not take into consideration the internal mechanics of the system and tends to be subjective.

So as to make sure the RIDA-KBS meets performance requirement and user expectation, the system performance is evaluated, followed by user acceptance testing.

5.5.1 System Performance Testing

System performance testing is done by preparing test cases. The test cases include samples of intrusion instances taken from KDDcup'99 intrusion data set. The instances include 20 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 numbers of attributes and the time it consumes to label it manually, the researcher prepared only 20 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. The same set of test instances are provided to RIDA-KBS and the outputs are compared to the domain experts' judgment.

Confusion matrix is used for comparing the performance of RIDA-KBS with domain experts' judgment. In confusion matrix, the entries in the matrix indicate the number of attacks labeled as let's; say attack X by domain experts and detected as attack X or attack Y by RIDA-KBS. System performance testing basically used to measure how accurate the system is. Precision, Recall, F-measure, True Positive rate measure how accurate the system is.

Table 5-2 Confusion matrix for evaluation of RIDA-KBS compared to experts' judgment

	RIDA-KBS Recommendation						
Domain experts suggestion	**Attack Class**	**Normal**	**Probe**	**DOS**	**R2L**	**U2R**	**Total**
	Normal	4	0	0	0	3	**7**
	Probe	0	4	0	0	0	**4**
	DOS	0	1	3	0	0	**4**
	R2L	0	0	0	4	0	**4**
	U2R	0	0	0	0	1	**1**
	Total	**4**	**5**	**3**	**4**	**4**	**20**

The confusion matrix on table 5-2 shows matrix of test cases evaluation by RIDA-KBS and domain experts' suggestion. The rows illustrate evaluation of domain expert and the columns illustrate result of RIDA-KBS.

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

The entries for *DOS* and *R2L* columns show that the system has correctly identified three instances as DOS and four instances R2L attack types respectively. The entries in the confusion matrix under *U2R* column depict that three instances out of four are incorrectly identified by the system as Normal and one instance is correctly classified as U2R. With regard to U2R attack type diagnosis the system achieved the lowest result as compared to the others.

It is clear that each type of attacks has their own way of attacking and causing damages to the victim computer. Identification of each class of attacks to their correct class is important to provide proper advice to network administrators so that they can take appropriate measures. But as shown in the confusion matrix, 3 normal instances are incorrectly identified as U2R attacks and 1 DOS is identified as probe attack. The researcher believes that the system's identification of these instances as an attack as strength though it is not to their correct class. The problem would have been they were identified as normal instances.

The system has correctly detected or identified 16 test instances out of 20 to their correct class. This means the system has 80% detection accuracy. And four instances out of twenty are incorrectly classified which is 25%. But, this measure alone is not enough to measure performance of the knowledge base system since it only tells us the overall performance. Hence Precision and Recall are employed to evaluate 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 **[70]**.

As clearly illustrated in table 5-3, the system's performance is evaluated in terms of TP rate, Precision, Recall and F-measure which enables us to view in detail how accurate is the system in identifying network attacks and normal incidents. Hence, RIDA-KBS has registered a TP rate of 57% for identifying normal network incidents correctly as normal but 43% of them are incorrectly identified as an attack.

Besides, RIDA-KBS scored 100%, 75%, 100% and 25% TP rates for Probe, DOS, R2L and U2R attacks respectively.

According to the rule: **attack(u2r):-root_shell(root_shell>=1) ,duration(duration>=25)** in the knowledge base of RIDA-KB, row 16 from the test case is labeled as U2R by both RIDA-KBS and domain experts. But this rule also applies for instances in row 13 and 15 which the domain experts labeled them as normal though they are U2R attack types

based on the above rule. This indicates that there is inconsistency in domain experts in labeling the instances correctly. Because of this the highest FP rate is registered for normal network incidents in that 3 normal instances are misclassified as U2R attack.

Table 5-3 Performance evaluation based on Precision, Recall, TP rate and F-measure

	TP Rate	Precision	Recall	F-Measure	Attack Class
	0.57	1	0.57	0.57	Normal
	1	0.8	1	0.89	Probe
	0.75	1	0.75	0.86	DOS
	1	1	1	1	R2L
	0.25	0.25	1	0.375	U2R
Weighted average	0.799	0.71	0.73	0.72	

Apart from True Positive rate, the system is also evaluated using its Precision and Recall. The system has registered its highest 100% Recall for R2L, U2R and Probe attacks and a 75% for DOS attack and 25% for U2R.

But since RIDA-KBS is expected of identifying each network incidents correctly so that relevant advice is delivered to the user accordingly, its precision is expected to be high. As compared to recall, the system has performed well in its precision for identifying network incidents accurately. Therefore, it scored 80% and 25% precision for, Probe and U2R network incidents respectively. The system has 100% precision for Normal DOS and R2L attacks.

To sum up, RIDA-KBS has an accuracy of 80%. This result is encouraging for using the system for intrusion detection and providing respective advice for users.

5.5.2 User Acceptance Testing

The aim of undertaking user acceptance testing is that to make sure how well RIDA-KBS 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. The experts, after providing training how RIDA-KBS works, are given test cases/ instances to use and evaluate the system. The evaluators assessed RIDA-KBS by using the following standards.

- 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
- Inclusion of suggestion and important advice about intrusion detection.
- The ability of the system to make right conclusion and recommendation
- Importance of the KBS in the domain area

Evaluation of the system using user acceptance testing is making sure how the users or domain experts' view the system on the base of the aforementioned evaluation standards. Different researchers have used different types of user acceptance testing evaluation criteria. But for this study, the evaluation criteria suggested by Solomon [**63**], Pu et al. [**71**], Seblewongel [**72**] and Rediet [**73**] customized and used to ease the evaluation process, the weights scale Suggested by Solomon [**63**] has been used such that Excellent = 5, Very Good =4, Good =3, Fair =2 and Poor =1

Table 5.4 depicts that summary of domain experts' evaluation of the system. The values indicate numbers of evaluator who evaluated the system as poor, fair, good, very good and excellent with respect to evaluation criteria.

Thus, 20% of evaluators replied poor for **simplicity to use and interact with the system**. And the same percentage, 40%, of respondents replied Fair and Good for it. This is due to

the command line interface used for interaction with the user such that users are expected to type commands and replies. With regard to the second question, that is **Attractiveness of the system** 40%, 20%, 40% of evaluators replied fair, good and Very Good.

The third question is about **Efficiency of the system in time**. All of the evaluators (100%) agreed that, the efficiency of the system is excellent in replying for their request. The fourth evaluation criteria is about **the accuracy of the system in reaching a decision to identify the types of network attacks** and 60% of evaluator scored very good and 40% of them as excellent. Whereas 60%, which is the majority, of the evaluator scored the system as very good and 20% as good and excellent for **Inclusion of suggestion important advise about intrusion detection.**

The other evaluation criterion is **the ability of the system to make right conclusions and recommendation.** Among the evaluators**,** 60% replied very good for RIDA-KBS and 40% of them replied excellent for KBS.

Table 5-4 User acceptance evaluation

No	Criteria of evaluation	Poor	Fair	Good	Very Good	Excellent	Average
1.	Simplicity to use and interact with the system	1	2	2	0	0	**5.5**
2.	Attractiveness of the system	0	2	1	2	0	**3**
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	0	3	2	**4.5**
5	Inclusion of suggestion and important advice about intrusion detection.	0	0	1	3	1	**4**
6	The ability of the system to make right conclusions and recommendation	0	0	0	3	2	**4.5**
7	Importance of the KBS in the domain area	0	0	0	3	2	**4.5**
		Total Average					**4.43**

The final criterion is i**mportance of the KBS in the domain area.** This criterion is included to measure how does RIDA-KBS is important in the area of network intrusion detection. And 60% of evaluators replied that very good and 40% replied good. This implies that the contribution of developing KBS like RIDA-KBS is important in the area of intrusion detection and advising of users after detecting network incidents as an attack or normal. Finally, according to the evaluation filled by domain experts', RIDA-KBS has registered 4.43 out of 5(88.6%), which is taken as very good achievement.

CHAPTER SIX

Conclusion and Recommendation

6.1 Conclusion

In this study, the possibility of integrating data mining models with knowledge based system is realized and explored. The integration process begun by taking samples of KDDcup'99 intrusion dataset. The dataset is preprocessed and made suitable for mining steps. Due to several limitations in acquiring knowledge for knowledge base from domain experts in the area of network intrusion detection, an automatic knowledge acquisition mechanism is proposed in this study. Data mining has proven to induce hidden knowledge from large collection of dataset. Hence, data mining classifier, JRip is employed for knowledge acquisition step since it has performed best among the selected classifiers with an accuracy of 99.88%.

The induction of network attack signatures and automatic knowledge acquisition for constructing knowledge base are performed simultaneously by automatic integrator application. Given rules generated, the integrator application has enabled building signature based intrusion detection and advising knowledge based system. The signatures are a set of rules describing about the types of network attacks. Besides, as the number of instances in the dataset changes it is apparent that the number of rules, the attribute combination in each rule and the values for the attributes also changes. The application has the ability of accommodating these changes and update the knowledge based system up to date.

Following the successful integration of induced knowledge with the knowledge based system, the rule based intrusion detection and advising KBS is built. System performance testing is undertaken to make sure that the right RIDA-KBS has been built. Hence the testing disclosed that the system has 80% of accuracy with very good Precision and Recall.

User acceptance testing is performed based on seven criteria of evaluation. Selected domain experts are trained and used the system to evaluate how much the KBS meets their requirements. The system on average scored 80% based on user acceptance evaluation. The system has registered 80.5% overall accuracy according to the system performance and user acceptance tests.

The performance analysis shows that RIDA-KBS registered acceptable performance. In addition, the study has proven that possibility of updating both rule base and fact base of the knowledge base system whenever the data size changes. And then the knowledge base system also provides advice and information based on the new changes yielding up-to-date knowledge base system. This characteristic distinguishes this study in that it somehow alleviated problems of signature based intrusion detection system which are claimed to have limitation in detecting new types of attacks.

However, further exploration and study has to be done to refine and yield a better knowledge based system which can be deployed in real network and provide advice to network administrators so that they can take timely and appropriate actions for a certain network incident.

Moreover, this study has paved the way for local researchers on using automatic knowledge acquisition techniques for the development of knowledge based system and motivates them to apply this approach than the conventional knowledge acquisition approach.

6.2 Recommendations

In this study promising result is achieved in integrating machine learning induced patterns with knowledge based system for detecting network attacks and providing advice to network administrators. Some challenges have been encountered which hinder the system from scoring a better achievement.

The first one is in the course of integration, two interfaces namely; graphical user interfaces (for the integrator) and command line interface for RIDA-KBS has been used. A

challenge has encountered in bringing the integrator and RIDA-KBS together under one interface. This is reflected on user acceptance test in that evaluator rated the simplicity to use and interact with the system below very good.

The other challenge encountered is on using knowledge which the KBS has already used previously before re-running the integrator application following a change in the numbers of dataset. In addition, the designed prototype KBS supports four types of attacks namely; probe, DOS, U2R and R2L. But as depicted in Appendix I, each type of classes of attacks are divided into specific attacks. Apart from this, some types of attacks are tailored to specific operating system.

JRip classifier has incorrectly classified 3 UR2 instances out of 9 as normal. These types of attacks are very dangerous therefore, a classifier which can better classify each instances to their correct class should be further explored.

Hence the researcher believes further researches have to be done to boost the benefits of integration of data mining with knowledge based system and the following are recommended for future study.

- Building hybrid knowledge based system which is capable of employing rule based reasoning and case based reasoning with integrated data mining techniques.
- Building KBS with graphical user interface which is simple to use and attractive to users.
- RIDA-KBS detects and advises for Probe, DOS, U2R and R2L attacks. But each type of attacks is name of set of attacks under them as indicated in Appendix I. For future work, it is recommended to design knowledge base system which directly detect attacks separately and provide the necessary information for it.
- Some of the attacks cause damages on specific operating system. In addition in reality most computer networks in organizations run specific network operating system (Linux, SunOs or Microsoft operating system). Therefore, designing platform specific knowledge based system is also recommended.

- To apply integration of machine learning with knowledge based system in other domain areas than intrusion detection, especially in areas where there is shortage of domain expert to acquire knowledge.

References

[1] Ali A.Wei Lu, Mahbod Tavallaee Ghorbani, *Network Intrusion Detection and Prevention concepts andtechniques*. New York: Springer, 2010.

[2] D Nagaraju, P Ramesh Kumar, and K Nageswara Rao P Srinivasulu, "Classifying the Network Intrusion Attacks using Data Mining Classification Methods and their Performance Comparison," *IJCSNS International Journal of Computer Science and Network Security*, vol. 9, pp. 11-18, June 2009.

[3] Fashoto S.G., Ojesanmi O.A. and Makinde O.E. Oyebode E.O., "Intrusion Detection System for Computer Network Security," *Australian Journal of Basic and Applied Sciences,* vol. 5(12), no. 1991-8178, pp. 1317-1320, 2011.

[4] P.R. Devale and G.v. Garje Snehal A. Mulay, "Intrusion Detection System using Support Vector Machine,"*International Journal of Computer Applications*, vol. 3 , no. 0975 - 8887, pp. 40-43, June 2010.

[5] Mihaela Oprea, "On the Use of Data-Mining Techniques in Knowledge-Based Systems," *Economy Informatics*, vol. 6, pp. 21-24, April 2006.

[6] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques*.: Morgan Kaufmann Publishers, 2000.

[7] Pedro Domingos, "Toward Knowledge-Rich Data Mining," *Data Mining and Knowledge Discovery,Springer*, vol. 15, no. 1, pp. 21-28, April 2007.

[8] SEO & PPC Management Solution. (2012, December) SEO & PPC Management Solution. (2013, Feb 23). [Online]. http://www.theecommercesolution.com/usefull_links/KBS_Data_Mining.php

[9] S TERRY BRUGGER, "Data Mining Methods for Network Intrusion," *ACM Journal* , vol. V, pp. 1-35.

[10] Jiawei Han and Micheline Kamber, *Data mining: concepts and techniques*, 2nd ed. San Francisco: Morgan Kaufmann Publishers, 2006.

[11] Theodoros Lappas and Konstantinos Pelechrinis, Data Mining Techniques for (Network) Intrusion Detection Systems, 2007, Department of Computer Science and Engineering.

[12] R. Shanmugavadivu N.Nagarajan, "Network intrusion detection system using fuzzy logic," *Indian Journal of Computer Science and Engineering (IJCSE),* vol. 2, no. 0976-5166, pp. 101-111, Feburary 2011.

[13] Ryszard S. Michalski, "Knowledge Mining: A Proposed New Direction," in *Sanken Symposium on Data Mining and Semantic Web*, Osaka University, 2003.

[14] Adamu T., "Computer Network Intrusion Detection: Machine Learning Approach," School of Information Science, Addis Ababa University, Addis Ababa, M.Sc Thesis 2010.

[15] Zewdie M., "Optimal feature selection for Network Intrusion Detection: A Data Mining Approch,"School of Information Science, Addis Ababa University, Addis Ababa, M.Sc Thesis 2011.

[16] Tigabu Dagne Akal, "Constructing Predictive Model for Network Intrusion Detection ," Addis Ababa University, Addis Ababa, M.Sc Thesis 2012.

[17] Jay R. Aroson and Ting-Peng Liang Efraim Turban, *Decision Support Systems and Intelligent Systems*, 7th ed. New Delhi, India: Prentice Hall of India, 2007,

[18] M. Mehdi Owrang O, "Database systems techniques and tools in automatic knowledge acquisition for rule-based expert system," in *Knowledge-Based Systems Vol 1*. Washington, United States of America:Acadamic Press, 2008, ch. 8, pp. 201-248.

[19] Gregory Piatetsky-Shapiro, and Padhraic Smyth Usama Fayyad, "From Data Mining to Knowledge Discovery in Databases," *American Association for Artificial Intelligence.*, vol. 17, no. 0738-4602-1996, pp. 37--54, 1996.

[20] Association ofr Computing Machinery. (2013, March 25) ACM KDD CUP. [Online]. http://www.sigkdd.org/kddcup/index.php?section=1999&method=data

[21] Ajith Abraham, "Rule-based Expert System," in *Handbook of Measuring System Design*, Peter H. Sydenham and Richard Thorn., Ed. Oklahoma, USA: John Wiley & Sons, Ltd. , 2005, ch. 130, pp. 909-919.

[22] R.P. Datta and Sanjib Saha, "An Empirical comparison of rule based classification techniques in medical data bases," in *2nd International Congress on Pervasive Computing and Management* , Sydney, Australia, 2009, pp. 1-15.

[23] Donald Nute, Andre Vellino Micheal A. Covington, *Prolog Programming In Depth*. New Jersey, USA: Prentice Hall, Upper Saddle River, New Jersey, 1997.

[24] SANS Institute InfoSec Reading Room, Intrusion Detection System: Definition, Need and Challenges,2001.

[25] Rafeeq Ur Rehman, *Intrusion Detection System with Snort : Advanced IDS Techniques Using Snort , Apache, MySQL, PHP and ACID*. Upper Saddle River, New Jersey, United States of America: Prentice Hall PTR, 2003.

[26] Julie Greensmith and Uwe Aickelin, "Firewalls, Intrusion Detection Systems and Anti-Virus Scanners,"School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, NOTTINGHAM NG8 1BB, UK, Computer Science Technical Report NOTTCS-TR-2005-1, 2005.

[27] Robert J. Shimonski. (2013, Feburary 13) WindowsSecurity.com. [Online]. http://www.windowsecurity.com/articles-tutorials/intrusion_detection/What_You_Need_to_Know_About_Intrusion_Detection_Systems.html

[28] Hervé Debar, "An Introduction to Intrusion-Detection Systems," in *Proceedings of Connect'2000*, Zurich, 2002, pp. 1-18, IBM Research, Zurich Research Laboratory.

[29] Marc Dacheir and Andreas Wespi Herve Debar, "A revised taxonomy of intrusion-detection systems,"IBM Research, Zurick, Research Report pp. 361-3782000.

[30] Tony Bradley. (2013, Feburary 16) About.com. [Online]. http://netsecurity.about.com/cs/hackertools/a/aa030504.htm

[31] Herve Debar. (2013, May 8) SANS. [Online]. http://www.sans.org/security-resources/idfaq/knowledge_based.php

[32] Charles Youman and Duminda Wijesekera Steven Noel, "Modern Intrusion Detection, Data Mining, And Degress Of Attack Guilt," in *Applications of Data Mining in Computer Security*, March 2002, pp. 2-25.

[33] Qiang Wang, A Clustering Algorithm for Intrusion Detection, Department of Computer and Information Sciences, Temple University, Philadelphia, USA.

[34] Joyce Jackson, "DATA MINING: A CONCEPTUAL OVERVIEW," *Communications of the Association for Information Systems*, vol. Volume 8, no. ISSN: 1529-3181 , pp. 267-296, 2002.

[35] 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.

[36] Duminda Wijesekera,Steven Noel Charles Youman, "Modern Intrusion Detection, Data Mining, And Degress Of Attack Guilt," March 2013.

[37] Yongjian Fu, "Data Mining: Tasks, Techniques and Applications," in *Introduction to Data Mining and its Applications*. Berlin, Germany: Springer Berlin Heidelberg, 2006, ch. 7, pp. 195-216.

[38] Max Bramer, *Principles of Data Mining*. Portsmouth, UK: Springer, 2007.

[39] Thair Nu Phyu, "Survey of Classification Techniques in Data mining," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, March 18-20,2009, pp. 978-988.

[40] Adriane B.S. Serapiao and Antonio C. Bannwart, "Knowlede Discovery for Classification of Three-Phase Vertical Flow Patterns of Heavy oil from Pressure Drop and Flow Rate Data," *Jornal of Petroleum Engineering* , vol. 2013, pp. 1-8, August 2010.

[41] Data mining: Rule based classifiers, 2013, Introduction to Data mining course slides, Avialable at: http://staffwww.itn.liu.se/~aidvi/courses/06/dm/lectures/lec4.pdf, Accessed 5/30/2013.

[42] William W. Cohen, "Fast Effective Rule Induction ," in *Machine Learning: Proceedings of the 12th International Conference (ML95)*, 1995, pp. 115--123.

[43] J. Rose Quinlan, *C4.5 Programs for Machine Learning*, Pat Langley, Ed. San Mteo, California, United States of America: Morgan Kaufmann publishers, 1993.

[44] M. M.,Ali, A.B.M.S, Tickle, K.S. Mazid, "A Comparison Between Rule Based and Association Rule Mining Algorithms," in *Third International Conference on Network and System Security*, Gold Coast, 2009, pp. 452-455.

[45] Priti Sajja Rajendra Akerkar, Knowledge-Based Systems, (2013, Feb 18). [online] http://books.google.com.et/books?id=mQZnd4zmZsoC&printsec=frontcover#v=onepage&q&f=false.

[46] Priti Srinivas Sajja and Rajendra Akerkar, *Advanced Knowledge Based Systems: Model, Application and Research*, vol. 1, pp. 1-11, 2010.

[47] Inc BookRags. (2013, Feburary 8) BookRags. [Online]. http://www.bookrags.com/research/knowledge-based-systems-csci-03/

[48] Cornelius T. Leondes, *Knowledge Based System Techniques and applications* , 1st ed. San Diego, United States of America: Accadamic Press, 2005.

[49] E De Kock, Chapter 6 - Expert systems and knowledge acquisition, 2003, University of Pretoria etd.

[50] John Platt, Rule-Based System, Available at: http://www.icsd.aegean.gr/lecturers/konsterg/teaching/KE/Rules.ppt, Accessed: Feb 20, 2013.

[51] Stuart Russel and Peter Norvig, *Artificial Intelligence : A modern Approach*, 3rd ed. Upper Saddle River, New Jersey, United States of America: Pearson Education, Inc., 2010.

[52] C.S Krishnamoorth and S. Rajeev, *Artificial Intelligence and Expert Systems for Engineers*.: CRS PRESS, 96.

[53] S. Ramani, S Muthu Raman, KSR Anjaneyulu, R. Chandrasekar M. Saskumar, *A Practical Introduction to Rule Based*. New Delhi, India: Norosa Publishing House, , 2007.

[54] Indika. (2011, May 20) differencebetween. [Online]. http://www.differencebetween.com/difference-between-prolog-and-lisp/

[55] David Hemmendinger. (2013, June 20) ENCYCLOPAEDIA BRITANICA. [Online]. http://www.britannica.com/EBchecked/topic/130670/computer-programming-language/248125/Visual-Basic

[56] Kindie Alebachew, "Designing a Hybrid Classifier for Network Intrusion Detection System," School of Information Science, Addis Ababa University, Addis Ababa, M Sc Thesis 2011.

[57] Mary Matthews, Anupam Joshi, Tim Finin Sumit More, "A Knowledge-Based Approach To Intrusion Detection Modeling," in *IEEE Symposium on Security and Privacy Workshops (SPW)*, San Francisco, CA, 2012, pp. 75 - 81, Computer Science and Electrical Engineering, University of Maryland, Baltimore county, Baltimor, MD, USA.

[58] Johnny S. K. Wong, Vasant Honavar, and Les Miller Guy G. Helmer, "Intelligent Agents for Intrusion Detection," in *In Proceedings, IEEE Information Technology Conference*, Syracuse, New York, 1998, pp. 121--124, Iowa State University, Ames, Iowa 50011.

[59] Idris Bharanidharan Shanmugam, and Abdul Manan Ahmed Norbik Bashah, "Hybrid Intelligent Intrusion Detection System," *World Academy of Science, Engineering and Technology 11* , pp. 23-26, May 2005.

[60] P. Yogesh, and A. Kannan S. Ganapathy, "Intelligent Agent-Based Intrusion Detection System Using Enhanced Multiclass SVM," *Computational Intelligence and Neuroscience*, vol. 2012, Article ID 850259, pp. 1-10, July 2012.

[61] M. Sadiq Ali Khan, " Rule based Network Intrusion Detection using Genetic Algorithm," *International Journal of Computer Applications, Published by Foundation of Computer Science*, vol. 18, no. 8, pp. 26-29, March 2011.

[62] Hong LIAN,GuYu HU ,GuiQiang NI ZhiSong PAN, "An Integrated Model of Intrusion Detection Based on Neural Network and Expert System," in *17th IEEE International Conference on Tools with Artificial Intelligence, 2005. ICTAI 05.*, Hong Kong, Novenmber 2005, pp. 1-2.

[63] Solomon Gebremariam, "Self-Learning Knowledge Based System for Diagnosis and Treatment of Diabets," School of Information Science, Addis Ababa University, Addis Ababa, M Sc Thesis 2013.

[64] Xueqiao Huang and John R. Jensen, "A machine-Learning Approach to Automated Knowledge-Based Building for Remote Sensing Imange Analysis with GIS Data," *Photgrammetric Engineering and Remote Sensing, American Society for Photogrammetry and Remote Sensing*,vol. 63, pp. 1185-1194, October 1997.

[65] 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.

[66] MIT. (2013, May 8) Lincoln Laboratory. [Online]. http://www.ll.mit.edu/index.html

[67] wikipedia. (2013, June 21) Wikipedia. [Online]. http://en.wikipedia.org/wiki/Parsing

[68] Diane J. Cook. (2013, May 8) Washington State University. [Online].

http://www.eecs.wsu.edu/~cook/ai/lectures/prolog/node3.html

[69] Hassan M. Ghaziri Elias M. Awad. (2013, May 26) Google Books. [Online]. http://books.google.com.et/books?id=CI63F2n4N7AC&pg=PA264&lpg=PA264&dq=user+acceptance+testing+for+knowledge+based+systems&source=bl&ots=vy5jTbheAg&sig=A8PSoV6UWAwcMHrQm34nNk2wDU0&hl=en&sa=X&ei=PnqTUfCDKMjWOZLCgfgH&redir_esc=y#v=onepage&q=user%20acceptanc

[70] 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.

[71] 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.

[72] Seblewongel E., "Prototype of Knowledge Based System for Axiety Mental Disorder Diagnosis," School of Informatin Science, Addis Ababa University, Addis Ababa, Msc Thesis.

[73] Rediet A., "Design and Development of a Prototype Knowledge-Based System for HIV Pre-Test Counseling," School of Information Science, Addis Ababa University, Addis Abab, Msc Thesis 2006.

Appendix I

Class of attack	Attack type	description
R2L	ftp_write	The Ftp-write attack is a Remote to Local User attack that takes advantage of a common anonymous ftp mis-configuration. The anonymous ftp root directory and its subdirectories should not be owned by the ftp account or be in the same group as the ftp account. If any of these directories are owned by ftp or are in the same group as the ftp account and are not write protected, an intruder will be able to add files (such as an rhosts file) and eventually gain local access to the system
	warezmaster	
	guess_passwd	
	imap	The Imap attack exploits a buffer overflow in the Imap server of Redhat Linux 4.2 that allows remote attackers to execute arbitrary instructions with root privileges. The Imap server must be run with root privileges so it can access mail folders and undertake some file manipulation on behalf of the user logging in. After login, these privileges are discarded.
	multihop	
	phf	The Phf attack abuses a badly written CGI script to execute commands with the privilege level of the http server. Any CGI program which relies on the CGI function escape_shell_cmd() to prevent exploitation of shell-based library calls may be vulnerable to attack. In particular, this vulnerability is manifested by the "phf" program that is distributed with the example code for the Apache web server
	spy	
	warezclient	
DOS	back	In this denial of service attack against the Apache web server, an attacker submits requests with URL's containing many frontslashes. As the server tries to process these requests it will slow down and becomes unable to process other requests
	Neptune	For each half-open connection made to a machine the tcpd server adds a record to a data structure describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections. The half-open connections data structure on the victim server system will eventually fill; then the system will be unable to accept any new incoming connections until the table is emptied out. Normally there is a timeout associated with a pending connection, so the half-open connections will eventually expire and the victim server system will recover. However, the attacking system can simply continue sending IP-spoofed packets requesting new connections faster than the victim

		system can expire the pending connections. In some cases, the system may exhaust memory, crash, or be rendered otherwise inoperative.
	pod	
	smurf	In the "smurf" attack, attackers use ICMP echo request packets directed to IP broadcast addresses from remote locations to create a denial-of-service attack. There are three parties in these attacks: the attacker, the intermediary, and the victim (note that the intermediary can also be a victim)
	teardrop	
	land	Some implementations of TCP/IP are vulnerable to packets that are crafted in a particular way (a SYN packet in which the source address and port are the same as the destination--i.e., spoofed). Land is a widely available attack tool that exploits this vulnerability.
Probe	satan	Network probing tool which looks for well known security vulnerabilities.
	portsweep	Surveillance sweep through many ports to determine which services are supported on a single host. Portsweeps can be made partially stealthy by not finishing the 3-way handshake that opens a port (ie. FIN scanning).
	Nmap	Network mapping using the nmap tool. Mode of exploring network will vary--options include SYN,FIN and ACK scanning with both TCP and UDP, as well as ICMP (Ping) Scanning.
	Ipsweep	Surveillance sweep on a network to determine what machines are on a network, as well as what services these machines are running.
U2R	buffer_overflow	
	loadmodule	SunOS 4.1.x) The loadmodule program is used by the xnews window system server to load two dynamically loadable kernel drivers into the currently running system and to create special devices in the /dev directory to use those modules. Because of the way the loadmodule program sanitizes its environment, unauthorized users can gain root access on the local machine. A script is publicly available and has been used to exploit this vulnerability.
	rootkit	Rootkit is a scenerio in which an attacker breaks into and then installs a rootkit on a target machine. A rootkit is a collection of programs that are intended to help a hacker maintain access to a machine once it has been compromised.

Appendix II

Attributes relation and data declaration

```
@relation 'kdd-slkbsdm'
@attribute 'duration' real
@attribute 'protocol_type' {'tcp','udp', 'icmp'}
@attribute 'service' {'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'daytime', 'discard', 'domain', 'domain_u', 'echo',
'eco_i', 'ecr_i', 'efs', 'exec', 'finger', 'ftp', 'ftp_data', 'gopher', 'harvest', 'hostnames', 'http', 'http_2784', 'http_443',
'imap4', 'IRC', 'iso_tsap', 'klogin', 'kshell', 'ldap', 'link', 'login', 'mtp', 'name', 'netbios_dgm', 'netbios_ns', 'netbios_ssn',
'netstat', 'nnsp', 'nntp','ntp_u', 'other', 'pm_dump', 'pop_2', 'pop_3', 'printer', 'private', 'remote_job', 'rje',
'shell','smtp', 'sql_net', 'ssh', 'sunrpc', 'supdup', 'systat', 'telnet', 'time','uucp', 'uucp_path', 'vmnet', 'whois', 'X11',
'Z39_50'}
@attribute 'flag' { 'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH' }
```

@attribute 'src_bytes' real
@attribute 'dst_bytes' real
@attribute 'land' {'0', '1'}
@attribute 'wrong_fragment' real
@attribute 'urgent' real
@attribute 'hot' real
@attribute 'num_failed_logins' real
@attribute 'logged_in' {'0', '1'}
@attribute 'num_compromised' real
@attribute 'root_shell' real
@attribute 'su_attempted' real
@attribute 'num_root' real
@attribute 'num_file_creations' real
@attribute 'num_shells' real
@attribute 'num_access_files' real
@attribute 'num_outbound_cmds' real
@attribute 'is_host_login' {'0', '1'}
@attribute 'is_guest_login' {'0', '1'}
@attribute 'count' real
@attribute 'srv_count' real
@attribute 'serror_rate' real
@attribute 'srv_serror_rate' real
@attribute 'rerror_rate' real
@attribute 'srv_rerror_rate' real
@attribute 'same_srv_rate' real
@attribute 'diff_srv_rate' real
@attribute 'srv_diff_host_rate' real
@attribute 'dst_host_count' real
@attribute 'dst_host_srv_count' real
@attribute 'dst_host_same_srv_rate' real
@attribute 'dst_host_diff_srv_rate' real
@attribute 'dst_host_same_src_port_rate' real
@attribute 'dst_host_srv_diff_host_rate' real
@attribute 'dst_host_serror_rate' real
@attribute 'dst_host_srv_serror_rate' real
@attribute 'dst_host_rerror_rate' real
@attribute 'dst_host_srv_rerror_rate' real
@attribute 'class' {'normal', 'Probe', 'DOS', 'U2R', 'R2L'}
@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,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,1,1,1,0,0,3,3,1,0,0.33,0,0.33,0.33,0.67,0.67,R2L
0,tcp,telnet,RSTO,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,0,4,4,1,0,0.25,0,0.25,0.25,0.75,0.75,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,125,179,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,1,1,1,0,0,6,6,1,0,0.17,0,0.17,0.17,0.83,0.83,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

0,tcp,private,SH,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,121,1,0.01,0.92,0.93,0,0.93,1,0,0,Probe
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,281,10,1,1,0,0,0.04,0.06,0,255,10,0.04,0.08,0,0,1,1,0,0,DOS
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,282,11,1,1,0,0,0.04,0.06,0,255,11,0.04,0.08,0,0,1,1,0,0,DOS
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,283,12,1,1,0,0,0.04,0.06,0,255,12,0.05,0.08,0,0,1,1,0,0,DOS

Appendix III

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

Table 1: Basic features of individual TCP connections.

feature name	description	type
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

Table 2: Content features within a connection suggested by domain knowledge.

feature name	**description>**	**type**
count	number of connections to the same host as the current connection in the past two seconds	continuous
Note: The following features refer to these same-host connections.		
serror_rate	% of connections that have "SYN" errors	continuous
rerror_rate	% of connections that have "REJ" errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
Note: The following features refer to these same-service connections.		
srv_serror_rate	% of connections that have "SYN" errors	continuous
srv_rerror_rate	% of connections that have "REJ" errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 3: Traffic features computed using a two-second time window.

Appendix IV

Dear Evaluator,

This evaluation form is prepared aiming at measuring to what extend does RIDA-KBS is useable and acceptable by end users in the area of network administration. Therefore, you are kindly requested to evaluate the system by labeling (√) symbol on the space provided for the corresponding attribute 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 rated 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	Inclusion of suggestion and important advice about intrusion detection.						
6	The ability of the system to make right conclusions and recommendation						
7	Importance of the KBS in the domain area						
		Total					

Appendix V

Evaluation test cases/ instances

Duration	protocol_type	service	flag	src_bytes	dst_bytes	wrong_fragment	num_failed_logins	root_shell	num_access_files	is_guest_login	count	dst_host_count	dst_host_srv_count	dst_host_same_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate
60	tcp	telnet	S3	125	179	0	1	0	0	0	1	1	1	1	1	0	1
0	tcp	telnet	RSTO	125	179	0	1	0	0	0	2	2	2	1	0.5	0	0.5
0	tcp	telnet	RSTO	125	179	0	1	0	0	0	2	3	3	1	0.33	0	0.33
0	tcp	telnet	RSTO	125	179	0	1	0	0	0	1	4	4	1	0.25	0	0.25
0	tcp	private	S0	0	0	0	0	0	0	0	93	255	9	0.04	0	0	1
0	tcp	private	S0	0	0	0	0	0	0	0	94	255	10	0.04	0	0	1
0	tcp	private	S0	0	0	0	0	0	0	0	95	255	11	0.04	0	0	1
0	tcp	private	S0	0	0	0	0	0	0	0	86	255	13	0.05	0	0	1
0	tcp	private	RSTR	0	0	0	0	0	0	0	1	255	1	0	0.4	0	0
0	tcp	private	RSTR	0	0	0	0	0	0	0	1	255	1	0	0.41	0	0
5	tcp	csnet_ns	RSTR	0	0	0	0	0	0	0	1	255	1	0	0.41	0	0
0	tcp	private	RSTR	0	0	0	0	0	0	0	2	255	1	0	0.42	0	0
25	tcp	telnet	SF	269	2333	0	0	1	0	0	1	69	2	0.03	0.01	0	0
150	tcp	telnet	SF	1587	6707	0	0	0	0	0	1	1	1	1	1	0	0
103	tcp	telnet	SF	302	8876	0	0	1	1	0	1	1	1	1	1	0	0
184	tcp	telnet	SF	1511	2957	0	0	1	0	0	1	1	3	1	1	0.67	0
0	tcp	http	SF	237	909	0	0	0	0	0	9	255	255	1	0	0	0
0	tcp	http	SF	226	1484	0	0	0	0	0	10	255	255	1	0	0	0
0	tcp	http	SF	231	1600	0	0	0	0	0	11	255	255	1	0	0	0
0	tcp	http	SF	230	1651	0	0	0	0	0	12	255	255	1	0	0	0

Appendix VI

Code for the integrator application

```
public class Slkb2 {
      public static String mining_result="";
    public  int counter=0;
    public String Data="top_goal(X):-attack(X).\n" +
"\n" +
"%% Here comes the rules for identification or diagnosis of attacks\n" +
" :-reconsult('ask.pl').\n";
    public static String file_path="";
    public String tempArr="";
    public static void main(String[] args) throws FileNotFoundException,
IOException, Exception {
           System.out.print(file_path);
        //Slkb2 nw= new Slkb2();
        slkbsUImain Ui= new slkbsUImain();
        Ui.show();
       // nw.jripMining(null);
        //nw.factAndRuleGenerator(null);
    }
    public void jripMining(String path) throws FileNotFoundException,
IOException, Exception{
        file_path=jFileChooser1.getSelectedFile().getAbsolutePath();
        //BufferedReader reader= new BufferedReader(new
FileReader("C:\some\where\fole\path.arff"));
         BufferedReader reader= new BufferedReader(new FileReader(file_path));
        // TODO code application logic here
        Instances data = new Instances(reader);
        reader.close();
        data.setClassIndex(data.numAttributes() -1);
        //Instances labeled = new Instances (data);
        //PART part= new PART();
        JRip jrip = new JRip();
        String [] options = new String[8];
        options[0]="-F";
        options[1]="3";
        options[2]="-N";
        options[3]="2.0";
        options[4]="-O";
        options[5]="2";
        options[6]="-S";
        options[7]="1";
        jrip.setOptions(options);
        jrip.buildClassifier(data);
        mining_result=jrip.toString();
       System.out.print(jrip);
       System.out.print("Mining done.....");
       jTextArea2.setText(jrip.toString());
    }
   public void factAndRuleGenerator(String jRules) throws
FileNotFoundException{
  String newFact="";
  String askerPl="";
  FileReader fr = new FileReader("C:som\where\file\path.txt");
  BufferedReader bf= new BufferedReader( fr);
  /*
```

```
  String Data="top_goal(X):-attack(X).\n" +
"\n" +
"%% Here comes the rules for identification or diagnosis of attacks\n" +
" :-reconsult('ask.pl').\n";
  */
  int numOfLines=0;
  try
  {
   //RemoveString r= new RemoveString();
   String x="";
  //public int j=0;
   while ( (x= bf.readLine())!=null)
   {
         if( x.trim().length()!=0 )
         {
              if(!(x.contains("Number of
Rules")||x.contains("normal")||x.contains("JRIP
rules:")||x.contains("==========")))
           {
         String result = rulePreprocessor(x);
         String[] n= result.split(" ");
          String reverse="";
           //LineNumberReader lineNum = new LineNumberReader(bf);
           numOfLines+=1;
        for ( int i=n.length-1; i>=0; i--)
        {  if(i==n.length-1)
        {
              if (n[i].length()>0)
              {
          reverse+="attack("+n[i].toLowerCase() +"):-";
        break;
              }
        }
        }//end of for loop for attack type
              {
              for ( int i=0; i<n.length-1;)
              {
                        reverse+=n[i]+"("+n[i]+n[i+1]+n[i+2]+")"+ " ";
                                    }
                              else
                              {
                               reverse+=","+n[i]+"("+n[i]+n[i+1]+n[i+2]+")"+ "
";
                              }
                            newFact+=n[i]+"("+n[i]+n[i+1]+n[i+2]+")"+ ".\n";
                            askerPl+=n[i]+"("+"X"+"):-
ask("+n[i]+","+"X )"+".\n";

                              i+=4;
                    }
                    else
                    {
                          reverse+=n[i];
                                  i+=1;
                    }
              }//end of for loop for rule and fact base extractor
        }
        reverse.trim();
```

```
          Data+=reverse.toLowerCase()+".\n";
        }//end of fact base writter

        }
    }
 public String rulePreprocessor(String txt)
{
String fin= "";
    if (txt.trim().length()>0)
    {
 //  fin= txt.replace("JRIP rules:", "");
   //fin= fin.replace("===========","");
   fin= txt.replace("(", "");
   fin= fin.replace("and", ",");
   fin= fin.replace("<=", "=<");
   fin= fin.replace(")","");
   fin= fin.replace("=>", ":-");
   fin= fin.replace("class=", "attack");
    }
   return fin;
}
public void askerBuilder() throws FileNotFoundException, IOException
 {
  String asker_path="C:\some\where\file\path\askerPl.pl";
  String asker_w="C:\some\where\file\path\askerProlog.pl";
  int counter2=0;
  FileReader askerfr = new FileReader(asker_path);
  BufferedReader askerbf= new BufferedReader(askerfr);
  String[] X=new String[100];
  String read="";
  while((read=askerbf.readLine())!=null)
  {
          if(!(Arrays.asList(X).contains(read)))
          {
          X[counter2]=read.toString();
          counter2+=1;}
  }
  askerfr.close();
      try
     {
      BufferedWriter askerPrologWriter =new BufferedWriter(new
FileWriter(asker_w));
      for (int j=0;j<counter2;j++)
      {
      askerPrologWriter.write(X[j]);
      askerPrologWriter.newLine();
      askerPrologWriter.flush();
      }
      askerPrologWriter.close();
      }catch (Exception e){
           System.out.println(e);   }
```

Appendix VII

Prolog code for RIDA-KBS+

```
go:-
        greeting,
        load_kb,
        %solve,
        repeat,
        write('Enter choice:'),nl,
        write('first type load(to load the KB'),nl,
        write('type consult(to identify an attack)and quit(to exit from the SLKBS)'),nl,
     write('and remember to write your choices and/or answers in lower case or small letters'),nl,
        read(X),((X==load)->do(load);(X==consult)->do(consult);(X==quit)->do(quit)).
    % write('Please enter yes/no for the quesitons'),nl,nl.
greeting:-
        write('Welcome to self-learning knowledge based system'),nl,
        write('for indetifying network intrusion types and advising system'),nl,
        write('************************************************'),nl,
        write('     The system is designed                     '),nl,
        write('   and developed by Abdulkerim Mohammed        '),nl,
        write('************************************************'),nl,nl.
do(load):-load_kb,!.
do(consult):-solve,!.
do(quit):-!.
do(X):-
        write(X),
        write('This is not correct command:'),greeting,nl,
        fail.
load_kb:-
write('files loaded...'),nl.
solve:-
        reconsult('mainrules.pl'),
        top_goal(X),
        %cls,
        write('The type of network attack is :'),
        write(X),
        /*
        write('   /how?/'),nl,
        read(How),(How==how,
        write('It is called as '),
        write(X),write('  attack according to the rule(s):'),nl,
    % write(Attr),
        attack(X)),
        */
        reconsult('attack_description.pl'),
```

```
        (X=='r2l'->describe_r2l;X=='u2r'->describe_u2r;X=='dos'->describe_dos;X=='probe'-
>describe_probe),
        nl,
        abolish(known,3).
        %define(known,3).
solve:-
   write('This is not an attack, it is a normal network behavior'),nl.
describe_r2l:-write('What do you know about.....,'),nl,
   write('Please enter your choice of action(1 up to 4'),nl,
   write('1. General Information'),nl,
   write('2. Damages caused'),nl,
   write('3. Prevention'),nl,
   write('4. Exit R2L description'),nl,
   read(Reply),(Reply==1->general_info;Reply==2->damages;Reply==3->prevention,Reply==4->exit_r2l).
general_info:-write('A Remote to User attack occurs when an attacker who has the ability to send'),nl,
            write('packets to a machine over a network but who does not have an account on that
machine'),nl,
            write('and exploits some vulnerability to gain local access as a user of that machine.'),nl,
            write('how/done'),nl,
            read(Rep),(Rep=='how',(
write('Here are possible ways an attacker can gain unauthorized access to a local account on a machine:'),nl,
                  write('1.Buffer overflows in network server software (imap, named, sendmail).'),nl,nl,
                  write('2.The Dictionary, Ftp-Write, Guest and Xsnoop attacks all attempt to exploit'),nl,
                  write('  weak or misconfigured system security policies'),
write('3.The Xlock attack, a remote attacker gains local access by fooling a legitimate user'),nl,
                  write('  who has left their X console unprotected, into revealing their
password'))),nl,describe_r2l.
damages:-write('After the attacker login by unautorized account, he/she changes'),nl,
        write('the remote user''s computer.'),nl,describe_r2l.
prevention:-write('1.Set the Xconsole protected'),nl,
            write('2.Correclty configure system security'),nl,
            write('3.make passwords which are not easy to guess'),nl,
            write('4.Be carefull while opening Powerpoint macros'),nl,describe_r2l.
 exit_r2l:-!.
describe_u2r:-write('What do you want to know about User to Root (U2R) attacks'),nl,
   write('Please enter your choice of action(1 up to 4'),nl,
   write('1. General Information'),nl,
   write('2. Damages caused'),nl,
   write('3. Prevention'),nl,
   write('4. Exit U2R description'),nl,
   read(Reply),(Reply==1->u2r_general_info;Reply==2->u2r_damages;Reply==3-
>u2r_prevention;Reply==4->exit_u2r),nl.
u2r_general_info:-write('User to Root are types of attack in which the attacker attaks'),nl,
write('with access to normal user account of the system'),nl,nl,
                  write('Types of U2R:(types)'),nl,
                  read(Types),(Types=='types',
```

```
                write('1.Buffer overflows occur when a program copies too much data into a static
buffer'),nl,
                write(' without checking to make sure that the data will fit.')),nl,
                write('2.loadmodule attack exploits programs that make assumptions about'),nl,
                write('  the environment in which they are running.'),nl,
                write('3. anypw is s a Console User to Root attack that allows the attacker to logon to'),nl,
          write('  the system without a password. A boot disk is used to modify the NT authentication'),nl,
                write('  package so that a valid username can login with any password string.'),nl,
                write('  Logins via telnet also work with any password.'),nl,
   write('4.ntfsdos This console-based attack reboots the system from a floppy disk containing
NTFSDOS.EXE.'),nl,
                write('5.Perl exploits a bug in some Perl implementations'),nl,describe_u2r.
u2r_damages:-write('U2R attackers like perl and Xterm abuse vulnerabilities in the'),nl,
            write('system in order to gain super user privileges'),nl,describe_u2r.
u2r_prevention:-write('******** U2R attack Prevention Recomendations*********'),nl,nl,
                write('1.Make sure the availability of enough space while copying to static buffer'),nl,
                write('2.Make sure passwords are not easy to guess'),nl,
                write('3.Ensure maximum care on programs that manage temporary files'),nl,
                write('4.Be carefull while running two or more programs running
simultaneously'),nl,nl,describe_u2r.
exit_u2r:-!.
describe_dos:-write('What do you want to know about Denial of Service(DOS) Attack'),nl,nl,
             write('1. General information'),nl,
             write('2. Damages caused'),nl,
             write('3. Recommendation for Prevention'),nl,
             write('4. Exit DOS description'),nl,nl,
             read(Ans),(Ans==1->dos_general_info;Ans==2->dos_damages;Ans==3-
>dos_prevention;Ans==4->exit_dos),nl,nl.
dos_general_info:-write('****************General Information about DOS******************'),nl,nl,
               write('A denial of service attack is an attack in which the attacker makes'),nl,
                  write('some computing or memory resource too busy or too full to handle legitimate'),nl,
                  write('requests, or denies legitimate users access to a machine'),nl,describe_dos.
dos_damages:-write('************************DOS damages***********************************'),nl,
            write('mailbomb, neptune, or smurf attack abuse a perfectly legitimate feature.'),nl,
            write('teardrop, Ping of Death create malformed packets that confuse the TCP/IP'),nl,
            write('stack of the machine that is trying to reconstruct the packet.'),nl,nl,
            write('apache2, back, syslogd take advantage of bugs in a particular network
daemon.'),nl,describe_dos.
dos_prevention:-write('*****************Recommendations for Prevention of DOS attacks************'),nl,
                write('1.Install and maintain anti-virus software '),nl,
                write('2.Install a firewall, and configure it to restrict traffic coming into and leaving your
computer '),nl,
                write('3.Follow good security practices for distributing your email address '),nl,
                write('4.Apply email filters, it may help you manage unwanted traffic'),nl,nl,describe_dos.
exit_dos:-!.
describe_probe:-write('What do you want to know about Probe Attack'),nl,nl,
```

```
            write('1. General information'),nl,
            write('2. Damages caused'),nl,
            write('3. Recommendation for Prevention'),nl,
            write('4. Exit Probe description'),nl,nl,
            read(Ans),(Ans==1->probe_general_info;Ans==2->probe_damages;Ans==3-
>probe_prevention;Ans==4->exit_probe),nl,nl.
probe_general_info:-write('******************* Probe Description*************************************'),nl,
             write('Probing is an attack in which the hacker scans a machine or a networking'),nl,
                 write('device in order to determine weaknesses or vulnerabilities that may'),nl,
                 write('later be exploited so as to compromise the system.'),nl,nl,
                 write('how hackers cause damages,(enter how)'),nl,
                 read(How),How==how,
                 (write('An attacker with a map of which machines and services are available on'),nl,
                  write('a network can use this information to look for weak points.'),nl,nl),
                 write('there different types of probe attackes:(enter types)'),nl,
                 read(Types2),(Types2==types,
             write('Here are some of the types of probe attacks:'),nl,nl,
                 write('insidesniffer- Here the attacker merely attachs a new machine to an inside ethernet
hub,'),nl,
                 write('configured with an ip, and begins sniffing traffic. '),nl,nl,
                 write('Ipsweep-an Ipsweep attack is a surveillance sweep to determine which hosts are
listening'),nl,
                 write('on a network. This information is useful to an attacker in staging attacks and
searching'),nl,
                 write('for vulnerable machines.'),nl,nl,
                 write('Nmap is a general-purpose tool for performing network scans. Nmap supports
many different types of'),nl,
            write('port scans options include SYN, FIN and ACK scanning with both TCP and UDP, as well
as ICMP (Ping) scanning'),nl,
            nl,
            write('resetscan- sends reset packets to a list of IP addresses in a subnet to determine which'),nl,
            write('machines are active. If there is no response to the reset packet, the machine is alive.'),nl,
           write('If a router or gateway responds with "host unreachable," the machine does not exist.'),nl,nl,
            write('is-domain- Here the attacker uses the "nslookup" command in interactive mode to list'),nl,
            write('all machines in a given DNS domain from a mis-configured primary or secondary DNS
server.'),nl,
            write('Thus the attacker can learn what machines (IP addresses) belong to (and perhaps exist in)
the domain.'),nl,nl,
            write('SAINT- gathers information about the presence of various network information
services'),nl,
            write('as well as potential security flaws'),nl,nl,describe_probe).
probe_damages:-write('**********************Damages caused by Probe attacks*************************'),nl,
             write('Under here are some the damages caused by probe attacks.....'),nl,nl,
             write('probes attacks conjust the attacked network by sending too many Ping packets'),nl,
             write('attack DNS servers'),nl,
```

```
            write('look for mis-configured machine in a network and cause attack on
it.'),nl,nl,describe_probe.
probe_prevention:-write('********************** Recommendation for prevention of Probe
attacks*****************'),nl,nl,
                write('set network devices such as switchs, routers password protected'),nl,nl,
                    write('configure gateways not to accept Ping packets'),nl,nl,
                    write('make sure machines are well configured as per security rules'),nl,nl,
                    write('watch many Ping packets sent to machines and take proper
actions'),nl,nl,describe_probe.
exit_probe:-!.
```