Intrusion Detection Using Genetic Algorithms

Developed by

Fauzia Batool

338-FBAS/MSCS/F07

Supervised by

Mr.Qaisar Javaid

(Assistant Professor)

Department of Computer Science & Software Engineering

Faculty of Basic and Applied Sciences

International Islamic University Islamabad

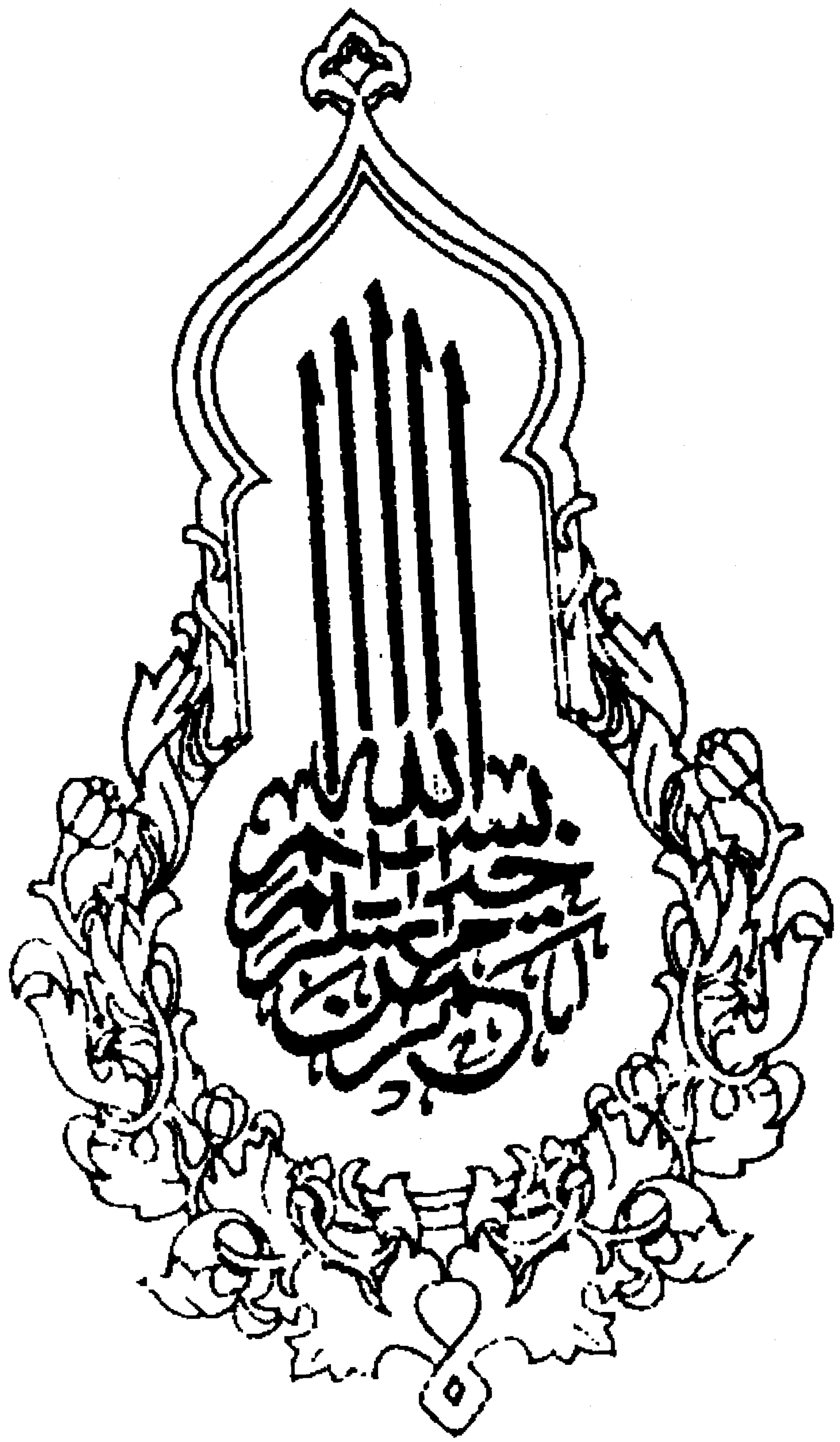
(2012)

Accession No TH-9653

MS
005.3
FAI

1 Software Engineering

2 Data Mining Techniques


بسم الله الرحمن الرحيم

Department of Computer Science & Software Engineering

International Islamic University, Islamabad

Date: ____________________

Final Approval

This is to certify that we have read the thesis submitted by Fauzia Batool,**338-FBAS/MSCS/F07.**It is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic University, Islamabad for the degree of **MSCS.**

Committee:

External Examiner:

Dr.Khalid Rashid [illegible]

Senior Advisor

COMSATS Institute of Information Technology ,Park Road,Chak Shahzad,Islamabad

Internal Examiner:

Dr.Zunera Jalil [illegible]

Assistant Professor

DCS &SE(FC),FBAS,IIUI

Supervisor:

Mr. Qaisar Javaid [illegible]

Assisstant Professor

DCS&SE,FBAS, IIUI

Dedication

I dedicate my work to **The Holy Prophet Muhammad** (Allah's grace and peace be upon him) lord of the world and the city of knowledge

&

To my parents **Abid Hussain Jaffer** and **Sakina Bibi** who are

the epitome of my existence and whose continuous

motivation and prayers

made me able to complete my thesis

&

To my parents- in- law **Muhammad Bashir** and **Shahnaz Begum**

who supported me in making this dream come true

&

To my husband **Muhammad Hafeez**

&

To my lovely daughters **Mumtahina Batool** and **Tatheer Batool.**

A dissertation Submitted To

Department of CS & SE,

Faculty of Basic and Applied Sciences,

International Islamic University, Islamabad

As a Partial Fulfillment of the Requirement for the Award of the

Degree of MSCS

Declaration

I hereby declare that this Thesis ***"Intrusion Detection Genetic Algorithms"*** neither as a whole nor a part thereof has been copied out from any source. It is further declared that we have done this research with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers especially my supervisor ***Mr.Qaisar Javaid***. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from any of the training institute or educational institutions, I shall stand by the consequences.

Fauzia

Fauzia Batool

338-FBAS/MSCS/F07

Acknowledgement

First of all I am obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this thesis..

I would like to pay my thanks to my supervisor **Mr.Sir.Qaisar Javaid, Assistant Professor, DCS&SE, IIUI** who guided me throughout my thesis work. And I would like to express my gratitude to **Dr.Muhammad Sher, HOD, DCS&SE, IIUI** who helped me choosing my topic of research.

Finally I would like to thank my father **Mr.Abid Hussain Jaffer** and My husband **Muhammad Hafeez** whose motivation was the main reason for completing my thesis.

Fauzia

Fauzia Batool

338-FBAS/MSCS/F07

Thesis

~~Project~~ In Brief

Project Title:	Intrusion Detection Using Genetic Algorithms
Undertaken By:	**Fauzia Batool**
	338-FBAS/MSCS/F07
Supervised By:	**Mr.Qaisar Javaid**
	Assistant Professor, DCS&SE,IIUI
Start Date:	March,2010
Completion Date:	February,2012
Tools and Technologies:	Java™ SE Development Kit 7
Documentation Tools	Microsoft Office 2007
	Microsoft Visio
	IEEE 829-1983 for testing documentation
Operating System:	Microsoft® Windows® XP Professional
System Used:	Dual core
	Intel Core I3

Abstract

Machine learning as well as data mining techniques is being actively used for implementation of Intrusion Detection System. But recently idea of using Genetic Algorithm based techniques has evolved. Some research towards this direction has been made. Up to some extent Genetic Algorithm based techniques have been used to solve data mining tasks as well.

In this Project emphasis was to use Genetic Algorithm based technique to detect the misuse. We will apply Genetic Algorithm's application in Data Mining in general and intrusion detection specifically

A system for intrusion detection has been proposed which is able to learn using existing attacks and then detect unseen attacks having a slight tilt of the existing attacks. Our system will detect DOS, Probe, R2L and U2R attacks. This system will save resources as the system will not have to calculate all features. It uses KDD data set and use 8 out of 41 features of this dataset.

Better accuracy is achieved compared to the systems using 5 features and those of six features, however a little more resources are required as compared to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abbreviatons

FD	Ficher linear discriminant
DDoS	Distributed denial of service
DoS	Denial of service
GA	Genetic Algorithm
HMM	Hidden Markov model
IDS	Intrusion Detection System
KDD	Knowledge discovery and data mining
KNN	K-nearest neighbour
LPM	Linear programming machine
MLP	Multi layer perception
RDA	Regularized discriminant analysis
R2L	Root to local
SVM	Support vector machine
U2R	User to root

Chapter 1
Introduction

1. INTRODUCTION

Today computer networks around the world are facing heightened challenges regarding security. These networks mostly belong to big corporate organizations, universities, strategic government departments and ISP's. There are daily hundreds of thousands of malicious attacks carried out by trouble-makers, using a variety of techniques: like Worms, Trojans, hacker intrusion into networks, denial-of-service (DoS) attacks by flooding the server and lately a more complicated version; the Distributed DoS (DDoS) attacks. Coping up with such kind of attacks is becoming a big problem in the field of Computer Sciences. Lately there has been a great trend of using machine learning and data mining techniques for Intrusion Detection Systems (IDS).

Intrusion Detection techniques can be classified into to two types. ***Misuse detection*** and ***Anomaly detection***. Both techniques have their pros and cons. *Misuse detection* have low false alarms but vulnerable to new attack which are not seen before. On the other hand *Anomaly detection* can detect the unseen attacks as it classifies anything deviant from the normal behavior as an intrusion. However it may cause problems in a scenario where the system encounters a new behavior which is not an intrusion. An IDS based on anomaly detection would alarm it as an intrusion. So this results in a high number of false alarms. Recently efforts are being made to implement a hybrid architecture which eliminates the negative points of the both types of modules and keep their benefits.

Genetic Algorithm(GA) based techniques are often used when the search space is very large. Genetic Algorithm is a guided random search technique. It has been derived from Darwin's theory of evolution. The basic principle of Genetic based techniques is the "*survival of the fittest*".

Machine learning as well as data mining techniques is being actively used for implementation of IDS. But recently idea of using Genetic Algorithm based techniques as evolved. Some research towards this direction has been made. Up to some extent Genetic Algorithm based techniques have been used to solve data mining tasks as well.

In this Project our emphasis was to use Genetic Algorithm based technique to detect the misuse. Next section presents the over view of Genetic Algorithm, its life cycle and Genetic Algorithm's application in Data Mining in general and intrusion detection specifically.

Information systems are a very important part for the infrastructure of almost all sectors of society. For this purpose they share information. Most of the users of this information are legitimate users but along with them, unauthorized users can also access the systems This makes network security the most famous problem nowadays. In spite of implementation of firewalls, prevention and intrusion detection systems, still a lot of intrusions go undetected.

There are several malicious attacks carried out by trouble-makers, using a variety of techniques: like Worms, Trojans, hacker intrusion into networks, denial-of-service (DoS) attacks by flooding the server and lately a more complicated version; the Distributed DoS (DDoS) attacks. Cop up with such kind of attacks is becoming a big problem in the field of Computer Sciences.

The mechanism that is used to find the difference or we can say is expected to differentiate between authorized and unauthorized users is called intrusion detection system. Intruder can attack through internet or an intruder can even be the authorized user of a system trying to get additional privileges or authorized user who is misusing a privilege given to him is also an intruder.

1.1. INTRUSION DETECTION SYSTEM

Due to increase in number of security threats there should be some intrusion detection system running as defense for each entity. So, company should have intrusion detection system.

Several reasons that caused use of IDS more secures and protects data on computers [7] are:

1. There should be penalty for the intruders who misuse or abuse the system and there should be safety measures to detect misuse.
2. Attacks and other violations that are not detected by the existing security systems should be detected.
3. There should be measures to detect and deal with the preambles to attacks.
4. The existing threats to the organization should be documented.

5. For larger organizations, it should be like a control for quality for administration and design.
6. Useful information should be provided about the misuse that happens and this should result in diagnosis, corrections of the problems and recovery of the system.

Intrusion detection systems are normally divided into two categories;
Anomaly detection and misuse detection. Some advanced techniques are being used to predict future intrusions. However artificial intelligence approaches are being applied to both types of intrusion detection systems.

1.1.1. Anomaly intrusion detection Approach

Anomaly detection can detect the attacks that are unknown or unseen as it shows anything that is deviant from the normal behavior as an intrusion. However problems can be caused in a scenario where the system faces a new behavior which is not an intrusion. Anomaly detection IDS would take it as an intrusion. So this causes high number of false alarms.

1.1.2. Misuse Intrusion Detection Approach

Misuse detection has low false alarms but it is vulnerable to new attack that is not seen before.
In misuse detection IDS matches a predefined pattern of known attack for an event or set of events. It tries to catch the intruders who intrude the system with some known technique.

Intrusion detection systems are also classified according to the resources they are monitoring. So, there are two categories: host based and network based. Host based intrusion detection systems are used for monitoring the host activities and network based intrusion systems are used for monitoring any intruder activity in the network data. Nowadays both are used together as hybrid systems.

1.2. GENETIC ALGORITHM OVERVIEW

Computer Scientist utilized miracles of genetics like neural networks and genetic algorithm. Both gained lots of success in problem solving.

John Holland at University of Michigan, introduced genetic algorithms for the first time in United States in 1970's[3].As more work is done on genetic algorithms in computational systems, they have become more attractive for problems of optimization. Specifically, genetic algorithms are more efficient on mixed (continuous and discrete) problems. They get less 'stuck' at local optima as compared to gradient search methods.

The idea of genetic algorithm is derived from Darwin's theory of evolution. The basic principle is called "survival of the fittest".

Genetic algorithms can evolve simple rules for network traffic which can differentiate between normal network connections and anomalous ones.

In evolutionary algorithms, initially we take a population that consists of all possible solution on which the principle of survival of the fittest is applied so that better approximations to the solution are produced. At every next generation, new approximations are selected according to their level of fitness in the problem domain by selecting new individuals and breeding them together using operators borrowed from natural genetics. With this process new individuals are created that are best fit according to their environment than the parent individuals just as it happens in natural adaptation.

1.2.1. The advantages of using GAs for Intrusion Detection

Deployment of genetic algorithms for intrusion detection has number of advantages, [9] namely:

• A genetic algorithm creates many offspring, so they are intrinsically parallel, they can explore the solution domain in various directions at the same time. If one path appears as a dead end, they can easily eliminate it and keep on working on new avenues, providing them a greater chance by finding the optimal solution.

• Because of parallelism that makes them to implicitly evaluate many options at once, they are particularly fit for solving the problems probability of potential solutions is very large - too huge to search in any reasonable amount of time, as network data is.

• GAs can cope well with attribute interactions and avoid getting stuck in local maxima because they work with populations of candidate solutions instead of single solution and employ stochastic operators to guide the search process permit, which jointly make them very suitable for dealing with classifying rare class, as intrusions are.

• There is a big possibility of evolving new rules for intrusion detection because system based on GAs can easily be re-trained. This property makes a GA system adaptable.

Figure [4 Pohlheim, Hartmut. - *Genetic and Evolutionary Algorithms:Principles, Methods and Algorithms.* Genetic and Evolutionary Algorithm Toolbox. (2001) from: http://www.geatbx.com/docu/algindex.html.] shows the structure of a simple genetic algorithm. Starting by a random generation of initial population, then evaluate and evolve through selection, recombination, and mutation. Finally, the best individual (chromosome) is picked out as the final result once the optimization meets its target.

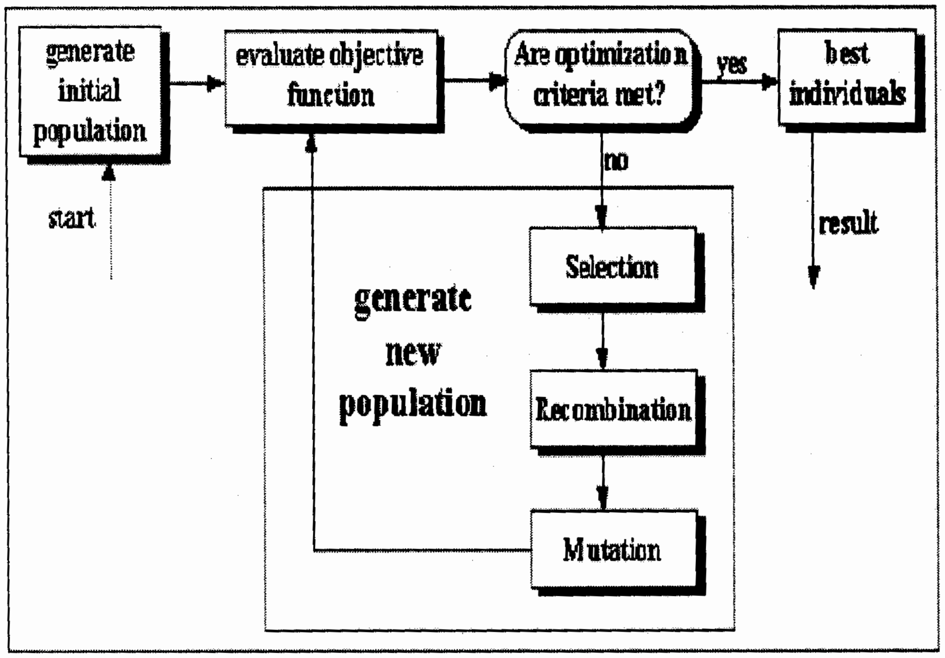


Figure 1: An overview of Genetic Algorithm

1.2.2. Process of Genetic Algorithms

GA consists of chromosomes that reproduced over many generations with respect to their fitness in a particular environment. Most fit chromosomes are most likely to *survive, mate, and bear children* [W. Spears, and V. Anand, _A Study of Crossover Operators in Genetic Programming", In Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems, Charlotte, NC.1991, pp. 409-418.7].Genetic algorithms end with fixed number of maximal generations or if required fitness level is achieved or if there is no advantage of running the algorithm till certain fixed generations.

1.2.3. Parameters for a Genetic Algorithm

Key points about genetic algorithms are:

1.2.4. Fitness Function

Fitness function is a basic element for a genetic algorithm. It calculates the goodness of an individual for achieving the objective of the problem. The result of fitness function is called a fitness value which is normally a real valued number. Fitness is calculated for all individuals who are the part of the population.

1.2.5. Genetic Algorithm operators

Selection, mutation and cross over are the operators for process of genetic algorithms. They have to participate in each generation for every population.

1.2.5.1. Selection operator

Selection operator prefers the better individuals as compared to the other ones and allows the individuals to inherit their genes to the next generation. Fitness is the criteria for an individual to be better than the other one which can be obtained by objective function or subjective judgment.

1.2.5.2. Crossover operator

This is the factor of genetic algorithms that make them different from other optimization techniques. Two individuals are selected from the population using selection operator. A cross over site is chosen within a bit string randomly. The values of these strings are exchanged. Then the new two offsprings are inherited into the next generation. By combining different portions of good individuals, better offsprings are achieved.

1.2.5.3. Mutation operator

This operator is used for maintaining diversity of one generation of a population from the other generation.

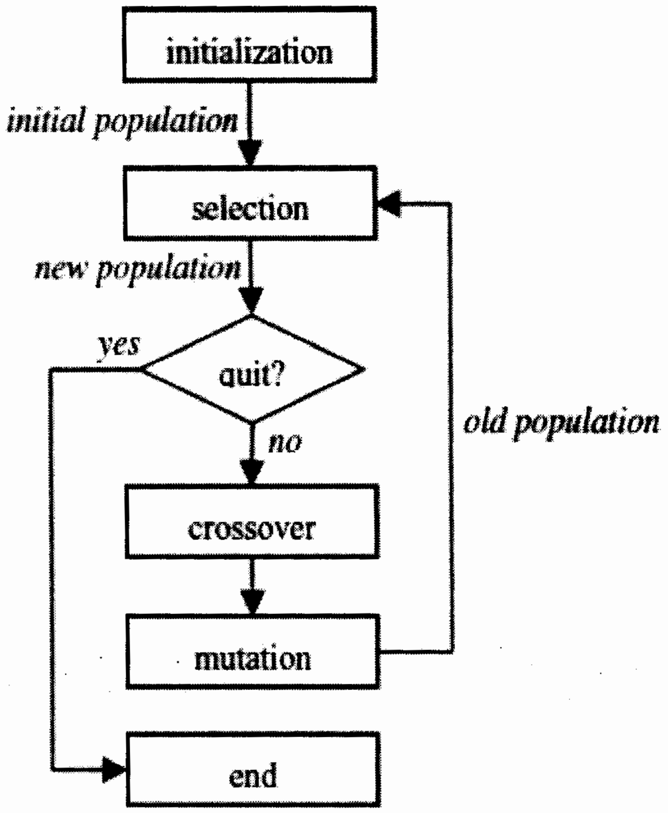


Figure 2: Genetic Algorithm representation

1.2.6. Genetic Cycle

1. **[Start]** Random population of n number of chromosomes is generated (they are basically possible solutions for the problem)
2. **[Fitness]** The fitness f(x) is calculated or evaluated for each chromosome present in the population.

3. **[New population]** keep on creating new population until the new population is complete. And for that, repeat all the following steps.
4. **[Selection]** Two parent chromosomes are selected according to their fitness from the whole population.(the better fitness, the bigger chance to be selected)
5. **[Crossover]** With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
6. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
7. **[Accepting]** Place new offspring in the new population
8. **[Replace]** Use new generated population for a further run of the algorithm
9. **[Test]** If the end condition is satisfied, **stop,** and return the best solution in current population
10. **[Loop]** Go to step **2**

1.3. Types of Attacks

There are four categories of attack type:

1	Probe --surveillance and other probing--
2	Denial of service (DOS)
3	User-to-root (U2R) --unauthorized access to local superuser--
4	Remote-to-local (R2L) -unauthorized access from a remote machine-

Table 1: four categories of attack type

Four broad categories which have few sub categories.

Probe	**DOS**	**U2R**	**R2L**
Ipsweep	Back	Buffer_overflow	Ftp_write
Nmap	Land	Loadmodule	Guess_passwd
Portsweep	Neptune	Perl	Imap
Satan	Pod	Rootkit	Multihop
	Smurf		Phf
	Teardrop		Spy
			Warezclient
			warezmaster

Table 2: Sub-categories of attacks

Our purpose is to generate an intrusion detection system with the help of genetic algorithms that can generate set of rules from a large population of rules. These rules generated will be the best among the population. And they will be created using 8 attributes out of 41 of data attributes. This is for reducing cost and each attack type will have different 8 attributes that will be most relative to that attack type.

1.4. Organization Of Thesis

Rest of the thesis is organized as follows.

Chapter 2: Literature review is given in order to review the existing related work in detail. Important findings are extracted.

Chapter 3: Requirement analysis gives an analysis of existing Intrusion detection approaches that use artificial intelligence techniques to find intrusions. Formulate the problem and includes the problem statement and proposes the solution in detail.

Chapter 4.This chapter explains a brief introduction of intrusion detection using genetic algorithm. Implementation details of proposed solution are also described.

Chapter 5: This chapter contains different test scenarios in the form of test cases and graphs that compare the results with previous works done.

Chapter 6: Presents conclusion drawn from the work done and scope for work in future.

Chapter 2
Literature Review

2. LITERATURE REVIEW

2.1. Introduction

The main focus of this chapter is to provide detail of literature survey that explains what we are actually trying to investigate. We will investigate how different types of artificial intelligence techniques are used to develop intrusion detection systems and how we find genetic algorithms better than the other techniques. And at the end we will see the ranking of attributes of KDD data set.

2.2. Related Research

According to Kim and Bentley [3], no existing IDS model yet satisfies the requirements completely. We summaries these requirements here in order to analyze whether the existing AIS-based IDSs reviewed in this paper have provided some of these functions. The seven requirements reported in [8] are as follows:

- Robustness: it should have less failure rates.
- Configurability: According to the local requirements, it should be able to configure itself easily
- Extendibility: scope of IDS should be easy to extend.
- Scalability: it should be scalable to cope with high volume data.
- Adaptability: As intruders keep on doing new kind of intrusions, it should be adaptable to changes.
- Efficiency: it should be low cost and have low overhead for the system.

2.2.1. Artificial intelligence techniques used for Intrusion Detection

Due to use of large amount of data in the network nowadays, artificial intelligence techniques are being applied to detect intrusions within the network. Most reliable and efficient intrusion detection systems are constructed with artificial intelligence techniques [10].[11],[12],[13],[14], [15], [16], [17], [18], [19] used kdd data for developing intrusion detection systems with different machine learning techniques. The results of all of them are represented in the following table:

Technique	Detection rate(%)	False positive rate(%)
C4.5	95	1
Support vector machin8e (SVM)	95.5	1
Multi layer perception (MLP)	94.5	1
k-nearest neighbor (k-NN)	92	1
Linear programming machine (LPM)	94	1
Regularized discriminant analysis (RDA)	94	1
Ficher linear discriminant (FD)	89	1
γ-algorithm	80	1
k-means clustering	65	1
Single leakage clustering	69	1
Quarter-sphere SVM	65	1
Y-means clustering	89.89	1
Genetic programming ensemble for distributed intrusion detection (GEdIDS)	91	0.43
SVM+ GA	99	-
SVM+ FuzzyLogic	99.52	0.44
Neural Networks + PCA	92.22	-
C4.5 + PCA	92.16	-
GA	92.47	0.69
C4.5+Hybrid neural networks	93.26	0.2
Hidden Markov model (HMM)	79	-

Table 3: Different Artificial Intelligence Techniques used for intrusion detection

2.2.2. Genetic Algorithms and KDD data used for intrusion detection

[4] has used genetic algorithm for intrusion detection. He has used KDD data to run the algorithm. KDD data consist of 41 attributes. In this paper he has discovered that if we used 8 attribute we can identify the attack. The result of his experiments is then sent to KDD cup. If we use all 41 attribute it will take more computation power. So, more time is required to detect the attack.

2.2.3. Ranking Of Data Attributes

[5] worked on KDD data .They used KNN classifier to rank all the attributes for each attack type. After this, they determined 5 most important features for each type of attack for kdd cup 99 data. In this paper they present the 5 attributes to detect the intrusion. By using 5 attributes their result is very much close to Dong song that used 8 attributes. In this paper he did not test his algorithm on real data. He test his algorithms on KDD Data .They made a rule generator that used these high ranked top most 5 features for each packet. Their classification of attributes with respect to each attack type is represented in the following table

Feature name	Rank	DOS	R2L	U2R	Probe
Duration	I	23	33	33	2
Protocol	2	34	3	6	37
Service	3	1	12	31	35
Flag	4	11	32	41	3
Sec_byte	5	24	36	17	6
Dst_byte	6	32	22	19	20
Land	7	33	I I	20	40
Wrong_fragment	8	12	6	1	26
Urgent W Count	9	16	13	3	39
Hot X Sev_count	IO	6	I	15	29
Num_failed_login	11	9	39	26	IO
Logged_in	12	17	5	30	16
Num_comprised	13	18	18	2	34
Root_shell	14	22	25	14	25
Su_attempted	15	25	26	22	31
Num_root	16	36	IO	35	I
Num_file_creations	17	3	23	31	18
Num_shells	18	1	35	38	19
Num_access_files	19	13	31	I	8
Num_cutbounds_cmds	20	15	1	8	17
Is_host_login	21	19	9	9	12

Is_guest_login	22	30	19	11	36
Count	23	35	38	21	5
Sev_count	24	40	40	12	21
Serror_rate	25	2	4	16	24
Sev_serror_rate	26	8	20	24	38
Rerror_rate	27	10	21	25	22
Srv_rerror_rate	28	14	11	27	7
Same_srv_rate	29	26	29	32	9
Diff_srv_rate	30	31	8	4	II
Srv_diff_host_rate	31	38	14	29	33
Dst_host_count	32	39	16	IO	32
Dst_host_srv_count	33	41	2	I8	23
Dst_host_same_srv_rate	34	20	30	23	13
Dst_host_diff_srv_rate	35	21	15	40	15
Dst_host_same_src_port_rate	36	29	31	13	14
Dst_host_srv_diff_host_rate	31	27	41	28	28
Dst_host_server_rate	38	31	24	36	21
Dst_host_srv_serror_rate	39	4	21	34	4
Dst_host_rerror_rate	40	5	28	5	41
Dst_host_srv_rerror_rate	41	28	34	39	30

Table 4: All attributes of KDD data

Most of the work done on intrusion detection systems nowadays proves their classification to be near to the best results. [21] also used their ranking to develop intrusion detection system and they used five features.

This ranking can be used to select the top most features for each attack type because as we go down the features don't count to be important. So, they can be discarded because functionality of the algorithm can be attained by using most important features to save computational cost they also used five features for intrusion detection.

Kim and Bentley (2007) worked with artificial immune systems for intrusion detection for improving problems of adaptability and effectiveness. They used AIS for making the IDS adaptable and effective. [3]

[9] used six features for intrusion detection. They also used genetic algorithms for their problem. Using less features reduced the efficiency of the intrusion detection system by 8 %.

For using genetic algorithms as our approach to find intrusion detections, we need to calculate the cost of each rule generated by the system.[20]created a cost based matrix for normal,DOS,Probe,U2R and R2L attacks. The same matrix is also used in our system to calculate the cost of rules generated in our system.

[22] rules created for detecting intrusions are presented in the following format,

(If {condition} then {act})

Where condition makes a comparison between the network connection and the rules that are specified and act is basically the action performed after the intrusion is caught This researcher used 9 basic features for intrusion detection

[23], used features and they used both offline training and online testing. Their method yielded good detection rate when generated rules were used to classify the data itself but when these rules were used to classify the testing data set the detection rates decreased up to 20%.And the rules can either be used to detect network intrusions or classify the type of intrusion detection depending upon selection of fitness function.

[24] identified those important features that are efficient and effective in building an IDS computationally. Using 41 input variables FNT was able to reduce the number of variables to 4, 12,12,8 and 10 and 9, 10, 9, 8 and 8 using 12 input variables for attack classes that were investigated. For most of the classes, the best accuracy was obtained using 41 variables. But the direct NN classifier outperformed for U2R attack. But the 12 variable dataset the direct NN approach outperformed the FNT model for DOS, U2R and R2L attacks.

[25] Improved the Support vector machine based intrusion detection systems with the help of genetic algorithms. He combined GA and SVM to improve the performance of IDS. An optimal detection model□ for SVM classifier was determined. As the result of the fusion, SVM based IDS did not only select optimal parameters for SVM but also optimal feature set among the whole feature set.

SVM got improved by applying genetic algorithm in such a way that number of features was minimized and detection rates were maximized. The detection rates were much higher than the systems that used only SVM for detection.

2.3. Findings

Following are the findings from the above literature:

- ❖ No existing IDS model yet satisfies the requirements completely.
- ❖ Most of the artificial intelligence techniques give good results for intrusion detection.
- ❖ Genetic Algorithms give a reasonable results and also low false alarm rate than other techniques.
- ❖ KDD data is the data mostly used as test data for developing intrusion detection systems.
- ❖ The ranking of attributes done by [5] is mostly used by researchers because it gives right results.
- ❖ Using 6 features reduced the efficiency of the algorithm.

2.4. Limitations

Artificial intelligence techniques are nowadays used for intrusion detection because domain is very large.

Genetic Algorithm is a good choice to develop intrusion detection system.It give low false alarm rates. But decreasing the number of features to decrease computation cost also decreases efficiency of the system. Recent researches use six features for finding intrusion with genetic algorithms. So, we decide to use eight features in order to keep system efficient and also to save computational cost.

Chapter 3
Proposed Technique

3. Proposed Technique

The algorithm is required to generate a set of rules for misuse detection. These rules are generated by learning the system on the training data. We have used only attack data for training. Our algorithm detects the types of attacks enabling network administrator to take appropriate measures as well. We used Michigan approach which means that one rule is represented by a single chromosome.

The algorithm modeling is done as following .

3.1. Representation

The basic step is how to present the problem in a form so that we can apply genetic algorithm on it to solve the problem. There are two types of approaches for rule generation. Michigan and Pittsburgh approach. In Pittsburgh each individual encodes a set of rules whereas in Michigan each individual encodes a single rule. Pittsburgh approach makes a larger individual because of which the fitness function becomes slow. Instead, Michigan approach creates smaller individual and makes fitness function calculations faster. But by adopting the Michigan approach we judge the quality of individual rules and hence we cannot determine the quality of the whole set which is obviously not the case in Pittsburgh approach.

In Michigan approach the rule representation can be divided into two parts Antecedent and Consequent. For Antecedent we will proceed as follows for each categorical attribute we consider its possible values. Let us suppose it has k possible values. We will allocate k bits to represent the attribute. For each possible value the corresponding bit will be ON if the rule contains the attribute has that value. The interesting point is that we can have multiple bits on which means that we can have OR operator in the rule for the possible values of the one attribute. For example: IF (Marital Status = "married" OR "divorced"). In machine learning terms this behavior is called Conjunctions of Disjunctions. If all the bits corresponding to one attribute are 1 then effectively that attribute plays no part in the rule. For this we can also keep a present/absent bit for an attribute. This representation is for categorical attributes. For numeric attributes we can keep binary values or we can use high level representations as well.

The Consequent part can be to encode in the genome of an individual and making it subject to evolution. The second possibility is that we keep the consequent value constant for a set of attributes. Now let us suppose we have m possible values for classification so we can now run over algorithm m times for each possible value and generating the best rules. The third approach can be selecting a more appropriate predicted class. This can be done as soon as the rule is generated the consequent part that will be the class that maximizes the individual's fitness

3.2. Fitness Function Since our training data consist of only attack types so we cannot use the prediction and accuracy. Our fitness function is as follows.

Fitness Function $F = P / \Sigma\, P_i$ -----------------> Equation No.1

Here P is the number of predicted attacks of a class which has the highest score and Pi is the number of predicted attacks for class i.

3.3. Selection The initial selection for the rules can be done in several ways. For example each example in the training data can vote for a rule which covers it. In a probabilistic way we can select the rules. Hence we get the rules covering different example. And which cover the most examples are selected. Alternatively we can randomly select the rules as well.

3.4. Cross Over The crossover is an important step in application of Genetic algorithms. There are two types of crossovers, 1-point and 2-point. We use 2-point crossover because it makes the position of each attribute in chromosome irrelevant. For selection of the chromosomes which take part in the crossover. We can simulate Roulette Wheel. According to Roulette wheel principle, each chromosome is given a weight according to their fitness function. Chromosomes with high fitness function have more chances to be selected. After that we randomly hit the numbers and select the chromosome.

The crossover can be used to specialize a rule or generalize a rule depending upon whether it is over fitting or under fitting the problem. For example:

Parents	Children produced by generalizing crossover	children produced by specializing crossover
0 (1 0) 1	0 (1 1) 1	0 (0 0) 1
1 (0 1) 0	1 (1 1) 0	1 (0 0) 0

3.5. Mutation Mutation step can be performed by randomly adding or subtracting small values from the values of attribute and hence generalizing or specifying it.

Brute force technique is a simple method and it always gives you the right solution to the problem. But the cost increases very quickly as the size of the problem increases.It is a good choice to use brute force approach when the problem domain is small but normally it is not used for large problems.

Using brute force is not a good approach as the domain becomes very large when we use it in the real world problem. Whereas we find genetic algorithms to be a good choice as it is proved to be a good technique for finding optimized solutions in a large domain. It gives nearest approximate results for the problems.

3.6. Problem scenarios

Z.Bankovic [9] used six features for intrusion detection .This reduced the efficiency by 8%. Its efficiency was reduced from 99% to 92.5%.

As discussed in literature survey genetic algorithms have been used for feature selection problem but their work was not for real data they worked just on test data. And their models could not provide the solution for new incoming intrusions in the network. The disadvantage with the use of AIS-based IDSs is that, at present, there is no way to know exactly what type of attacks will pass through undetected [6]. V.D.Kotov [7]Worked for adaptable intrusion detection systems using AIS but they mentioned that AIS would be only useful for LAN intrusions.

How to automatically and systematically build adaptable and extensible intrusion detection system using eight features of network data

We seek an *automatic* approach so that we can eliminate the manual and ad hoc elements from the development process of IDSs. We also seek a *systematic* approach so that the same set of development tools can be readily applied to different audit data sources. We evaluate the utility of our development approach using not only the effectiveness of the resultant IDSs, but also their *adaptability* and *extensibility.*

3.7. Problem Definition

Nowadays almost all the organizations have problems regarding intrusions in their systems. Which is a big threat for them i.e DOS, Probe, R2L etc. So, our basic purpose is to differentiate the attacks from the normal traffic and the method should be cost effective and should have good detection rates as the native systems are.

3.8. Focus of research

As we see decreasing number of features reduces the accuracy of the results and increasing them will increase the cost of algorithm .We want to develop an algorithm with more than six features and have opted to select eight features to go with my algorithm. So, a balance can be kept between efficiency and cost of the algorithm.

Rank	DOS	R2L	U2R	Probe
I	23	33	33	2
2	34	3	6	37
3	1	12	31	35
4	11	32	41	3

5	24	36	17	6
6	32	22	19	20
7	33	II	20	40
8	12	6	1	26

Table 5: Eight Attributes that are used in this project

3.9 System Design

Our system should be able to detect intrusions of all four attack type that are DOS, Probe, U2R, and R2L.

Genetic Cycle will be used to identify different type of attacks. A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems.

3.9.1 Genetic Cycle

1. **[Start]** Random population of n number of chromosomes is generated (they are basically possible solutions for the problem)
2. **[Fitness]** The fitness f(x) is calculated or evaluated for each chromosome present in the population.
3. **[New population]** keep on creating new population until the new population is complete .And for that, repeat all the following steps.
4. **[Selection]** Two parent chromosomes are selected according to their fitness from the whole population.(the better fitness, the bigger chance to be selected)
5. **[Crossover]** With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
6. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).

7. **[Accepting]** Place new offspring in the new population
8. **[Replace]** Use new generated population for a further run of the algorithm
9. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
10. **[Loop]** Go to step **2**

Our objective is to use Genetic Algorithm to detect intrusions in networks. We will try to implement Genetic Algorithm using Data Mining techniques. The resultant intrusion detection system should be effective and adaptable.

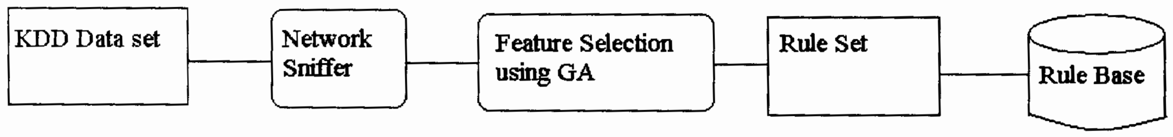


Figure 3: How the IDS will be implemented

We need to collect enough historical data that includes both normal and anomalous network connections. So we use KDD(Knowledge discovery and data mining) data set that is being used worldwide. This data set is analyzed by the network sniffers and results are fed into GA for fitness evaluation and feature selection. Then the GA is executed and the rule set is generated. These rules are stored in a database to be used by the IDS. The resultant rule set is with 8 selected features that will be efficient and reduce computation cost.

KDD Datasets contain 41 basic features (attributes), which helps to detect all kinds of intrusions on network. Thus, we will implement Genetic Algorithm using 8 attributes for each attack type.

3.9.2. Algorithm and functions

The algorithm we have used for intrusion detection is Genetic algorithm. The implementation of this algorithm is challenging one for us.

The algorithm as basic genetic algorithm with some modification which are required for our problem solving;

The algorithm has following steps:

```
Take R input for number of rules.
Generate randomly R number of rules according to number of bins of corresponding columns

FOR each K number of iterations
    DO
        READ each line from training set
        Compare it for each rule
            IF
                rule satisfies
               Then Check does it belong to same class label
                    IF
                    it is then ok
                    ELSE
                    Add penalty to rule cost
            ELSE
               Then check its class
    IF it belongs to same class then
        Add penalty to rule cost

    SORT rules in ascending order according to their cost
    Discard lower half of rules
    Randomly pick the two rules and cross them to each other and these yields two more
    chromosomes (rules).
    Add them to the rules.
WRITE the best rules of the iteration on the file along with their cost

END
```

Check all the refined rules on the corrected dataset.

Compute the cost of these rules

WRITE the best rules on the file along with their cost

The function used in the implementation of this algorithm is as follow.

sort (...): sort the chromosomes(rules) according to their penalty(cost). It takes the 2- D array of chromosomes and their respective cost and sorts them in ascending order.

procedure (...) In this function we read a packet from file and check one by one to each rule and gave them plenty or appraise according to their correctness. As its functionality suggests it takes chromosomes (rules) and for computation of cost fitness (cost) array.

iterative (...): It is heart of the program, where almost all the logic is implemented. It will take number of iterations, Rules, fitness array and run it as many times as user gave number of iterations. Actually it is the training step of our program. At the end of this function we will achieve sorted array of rules (Which are best up to given iterations) which will be passed to evaluate function.

evaluation (...): It will receive rules and run the test data which in our case is *corrected dataset* . And gave us output on the given parameters and write them to file;

writeToFile (...) : The function used to write Group of rules on each iterations to file along with their iteration number and their respective cost.

Chapter 4
Implementation

4. Implementation

4.1. Rule Generator for Misuse Detection (Algorithm)

The requirement of the algorithm is to generate a set of rules for misused detection. These rules are generated by learning the system on the training data. We used only attack data for training. Over algorithm detects the types of the attacks enabling network administrator to take appropriate measures as well. We used Michigan approach which means that one rule is represented by a single chromosome.

4.1.1. Start

We generated random population on *n* chromosomes, *n* is the number entered by the user.

4.1.2. Fitness

Fitness function is not much complex we have embedded and fitness value is calculated on the bases of the entries in the **cost Matrix**.

4.1.3. Selection

On the bases of Darwin's theory of survival of the fittest, half population is removed and the best half is then responsible of generating the other half.

4.1.4. Crossover

Crossover is totally randomized. Two random parents are selected and random amount of their portion is crossed-over and other half of the population is generated.

4.1.5. Mutation

Mutation was implemented with one random bit..

4.1.6. Accepting

The offspring are made part of population that are generated by crossover. This step chooses the best of the bests.

4.1.7. Loop Back

This process is repeated for number of iterations specified and then it is tested on corrected data to compare the results at the end.

Here how we check that whether its attack or not.

We check for any of these rules in code like that

DOS:

if((connection_length =0 or connection_length = 58329) And

(#OfFailedLoginAttemps =0 or #OfFailedLoginAttempts =5) And

(#ofConnectionsToSameHost= 2..24 or #ofConnectionsToSameHost= 25..146 or #ofConnectionsToSameHost= 147..271 or #ofConnectionsToSameHost= 271...510 or #ofConnectionsToSameHost>510) And

(#ofConnectionsToSameService= 2..7 or # ofConnectionsToSameService = 8..14 or # ofConnectionsToSameService = 15..23 or # ofConnectionsToSameService = 23...510 or # ofConnectionsToSameService>510) And

(CurrPercentageOfConnections<3 or CurrPercentageOfConnections= 3 . . 7 or CurrPercentageOfConnections>7)And

(Rule's representation corresponding to that is == 1)) || ((Count of the number of connections to the same host as the current connection>= 0 || <=255)And

(Rule's representation corresponding to that is == 1)) || ((number of connections to the same service as the current connection>=0||<=255)And

if ((successful login or not == 0 || 1)

then

Attack_type= DOS

If the output of above statement will come out true then we will declare that is intrusion otherwise we will say that it's not this type of attack.

Then we will check for do we got it wrong or correct. If we do have wrong then we got punishment of cost.

PROBE:

if ((Type of Protocol == "TCP" || "ICMP" || "UDP") &&

(Rule's representation corresponding to that is == 1)) || ((Number of data bytes from destination to source >= 0 || <=5155468)&&

(Rule's representation corresponding to that is == 1)) || ((Percentage of connections to different services >=0||<=100) &&

(rule's representation corresponding to that is == 1)) || ((percentage of connections to different hosts >=0 || <= 100) &&

(rule's representation corresponding to that is == 1)&&

(Rule's representation corresponding to that is == 1)) || ((number of outboundcommands>= 0 || <=16) &&

(Rule's representation corresponding to that is == 1)) || ((Percentage of connections to the current host that have "RST" error >=0 || <= 100) &&

(Rule's representation corresponding to that is == 1)) || ((Percentage of connection that have "SYN" error>=0||<=100) &&

Then it means that this rule says that it is **Probe**.

R2L

if ((successful login or not == 0 || 1) &&

(Rule's representation corresponding to that is == 1)) || ((Count of the number of connections to the same host as the current connection>= 0 || <=255)&&

(Rule's representation corresponding to that is == 1)) || ((number of connections to the same service as the current connection>=0||<=255) &&

(rule's representation corresponding to that is == 1)) || ((percentage of connections to a particular port from the same source >=0 || <= 100) &&

(rule's representation corresponding to that is == 1))&&

(#OfFailedLoginAttemps =0 or #OfFailedLoginAttemps =5)

(Rule's representation corresponding to that is == 1)) || ((Number of data bytes from destination to source >= 0 || <=5155468)

if ((is_guest_login == 0 || 1)

)

U2R

if ((Data Bytes from Connection to source >= 0 || <= 5155468) &&

(Rule's representation corresponding to that is == 1)) || ((number of file creation operations >= 0 || <=16)&&

(Rule's representation corresponding to that is == 1)) || ((Percentage of connection to different host >=0||<=100) &&

(Rule's representation corresponding to that is == 1)) || ((number of connections to the same service as the current connection >=0 || <= 255) &&

(Rule's representation corresponding to that is == 1)) || ((Percentage of connections that have "RST" error >=0 || <= 100) &&

(Rule's representation corresponding to that is == 1)) || ((number of accessfiles>= 0 || <=16)&&

(Rule's representation corresponding to that is == 1)) || ((number of outboundcommands>= 0 || <=16)&&

(connection_length =0 or connection_length = 58329

Then it means that this rule says that it is **U2R.**

4.2. KDD Cup 99 Information

KDD data set is the most commonly used for intrusion detection projects nowadays. It has 41 attributes.9 of which are basic and rest are derived ones.

4.2.1. Intruder Detection Learning

Our basic goal is to make software that should be able to distinguish between "normal" and "bad"connections.
we are using a training data that is 10% kdd data and a test data that is corrected.gz. Attacks fall into four main categories:

- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local super user (root) privileges, e.g., various ``buffer overflow" attacks;
- Probing: surveillance and other probing, e.g., port scanning.

TH 9653

The test data doesn't have the same probability of intrusions as in training data.and it includes specific attack types not in the training data. The task is more realistic because of this difference intrusion experts sometimes say that most novel attacks are a slightly varied form of known attacks and the "signature" of known attacks should be sufficient to catch new variants. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.

- Back dos,
- buffer_overflow
  u2r
- ftp_write r2l
- guess_passwd r2l
- Imap r2l
- Ipsweep probe
- land dos
- loadmodule u2r
- Multihop r2l
- Neptune dos

- nmap probe
- perl u2r
- phf r2l
- pod dos
- portsweep probe
- rootkit u2r
- satan probe
- smurf dos
- spy r2l
- Teardrop dos
- warezclient r2l
- warezmaster r2l

4.2.2. Features of Data:

KDD dataset has a total of 41 attributes that are shown as follows in the following table:

Feature	Description	Type
1. Duration	Duration of the connection	Conti nual
2. Protocol type	Connection protocol (udp, tcp, icmp)	Discr ete
3. Service	Destination service (e.g. telnet, ftp)	Discr ete
4. Flag	Status flag of the connection	Discr ete
5. Source bytes	Bytes sent from source to destination	Conti nual
6. Destination bytes	Bytes sent from destination to source	Conti nual

Feature	Description	Type
7. Land	1 if connection is from/to the same host/port; 0 otherwise	Discrete
8. Wrong fragments	Number of wrong fragments	Continual
9. Urgent	Number of urgent packets	Continual
10. Hot	Number of hot indicators	Continual
11. Failed logins	Number of failed logins	Continual
12. Logged in	1 if successfully logged in; 0 otherwise	Discrete
13. # Compromised	Number of "compromised" conditions	Continual
14. Root shell	1 if root shell is obtained; 0 otherwise	Discrete
15. su attempted	1 if "su root" command attempted; 0 otherwise	Discrete
16. # Root	Number of "root" accesses	Continual
17. # File creations	Number of file creation operations	Continual
18. # Shells	Number of shell promts	Continual
19. # Access files	Number of operations on access control files	Continual
20. # Outbound cmds	Number of outbound commands in an ftp sessions	Continual
21. Is hot login	1 if the login belongs to the "hot" list; 0 otherwise	Discrete
22. Is guest login	1 if the login is a "guest" login; 0 otherwise	Discrete
23. Count	Number of the connections to the same host as the current	Conti

Feature	Description	Type
	connection in the past two seconds	nual
24. srv count	Number of connections to the same service as the current connection in the past two seconds	Conti nual
25. serror rate	% of connections that have "SYN" errors	Conti nual
26. srvserror rate	% of connections that have "SYN" errors	Conti nual
27. rerror rate	% of connections that have "REJ" errors	Conti nual
28. srvrerror rate	% of connections that have "REJ" errors	Conti nual
29. Same srv rate	% of connections to the same service	Conti nual
30. Diff srv rate	% of connections to different services	Conti nual
31. srv diff host rate	% of connections to different hosts	Conti nual
32. dst host count	count of connections having the same destination host	Conti nual
33. dst host srv count	count of connections having the same destination host and using the same service	Conti nual
34. dst host same srv rate	% of connections having the same destination host and using the same service	Conti nual
35. dst host diff srv rate	% of different services on the current host	Conti nual
36. dst host same src port rate	% of connections to the current host having the same src port	Conti nual
37. dst host srv diff host rate	% of connections to the same service coming from different host	Conti nual
38. dst host srvserror rate	% of connections to the current host that have S0 error	Conti nual
39. dst host srvserror rate	% of connections to the current host and specified service that have an S0 error	Conti nual

Feature	Description	Type
40. dst host rerror rate	% of connections to the current host that have RST errors	Continual
41. dst host srvrerror rate	% of connections to the current host and specified service that have an RST error	Continual

Table 6: Description of All attributes of KDD data

4.2.3. Datasets

We have used KDD-99.our task was to build a network intrusion detection system, a model that can predict and is capable of distinguishing between "good" normal connections and "bad" connections, called intrusions or attacks. This database contains a standard set of data to be used, which includes a wide variety of intrusions simulated in a military network environment.

The data is available in full (743M of network connections), or in a number of smaller data sets. We have sampled data from the set containing 10% of the original data set (kddcup.data_10_percent). The reason we choose this data set was because it is a rich dataset with hundreds of thousands of records and 41 attributes which include numeric and categorical types. Due to these silent features this dataset is becoming an international standard to evaluate IDS algorithms.

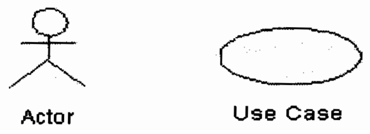
4.2.4. Input and Output Format

There are basically four Project files, one for each attack type. Each attack type is implemented alone to distribute the work load and achieve efficiency.

- Algorithm prompts for number of iterations and number of rules to be generated for single iteration.
- Algorithm reads 10% data sample file and start leaning.
- After completion of the specified number of iterations, Chromosomes are then tested on corrected data file.
- All these entries are written in output files (one for each attack type)

4.3. Use Case Diagram

A use case diagram consists of an actor and the use case. Actor is the person performing the action and the use case is the action itself



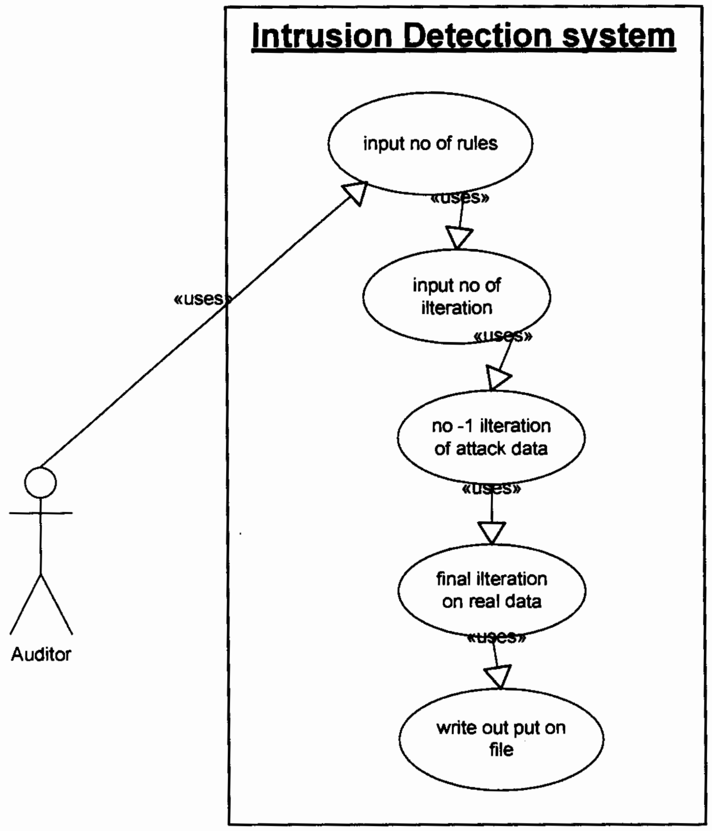


Figure 4: Use case diagram demonstrating system interaction

4.4 Sequence Diagram

These diagrams are used to show the flow of the system. There can be more than one sequence diagrams for one use case diagram For different kind of actions there are different sequence diagrams. Sequence diagrams tell which operation will be executed first .the messages being sent from one object to the other. The following are sequence diagrams.

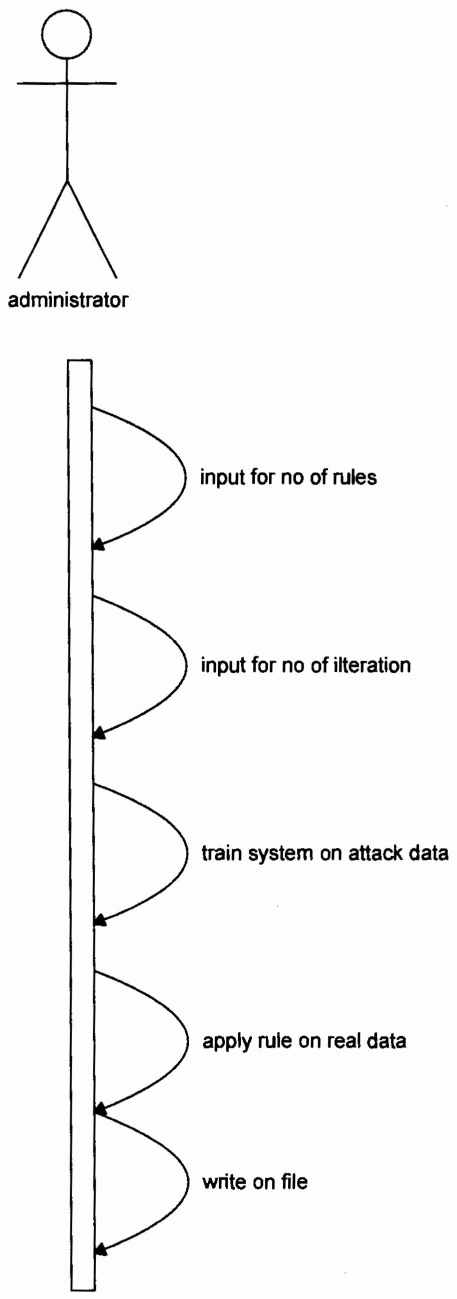


Figure 5: Sequence diagram demonstrating flow of functionality

4.5. Activity Diagrams

They represent the activities present in use cases. It is actually the procedural design in UML. Operations in use cases in sequence are represented in activity diagram. Activity diagram are useful when we want to describe a parallel behavior, to show how behaviors in several use cases interact. The activity diagrams are described as follows.

4.5.1. Activity Diagram of DOS

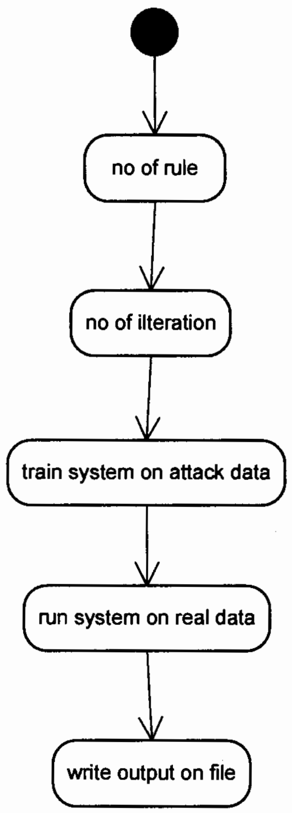


Figure 6: Activity diagram of DOS

4.5.2. Activity Diagram of U2RFinal

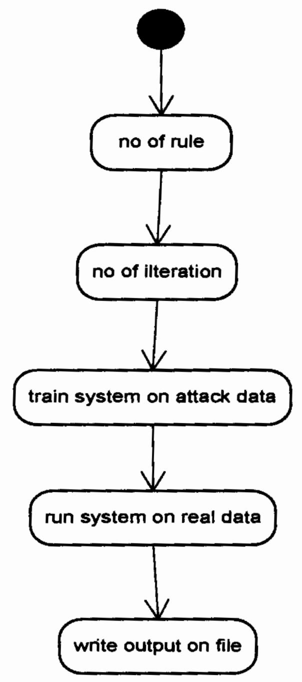


Figure 7: Activity Diagram of U2RFinal

4.5.3. Activity Diagram of R2L

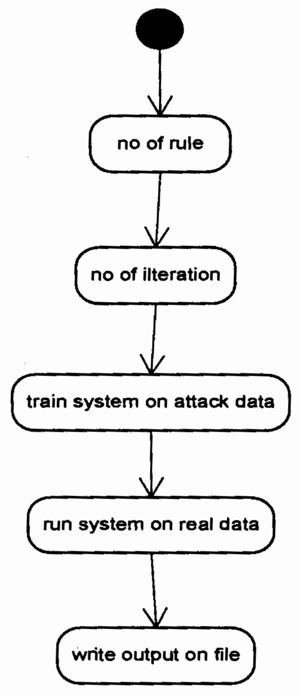


Figure 8: Activity Diagram of R2L

4.5.4. Activity Diagram of Probe

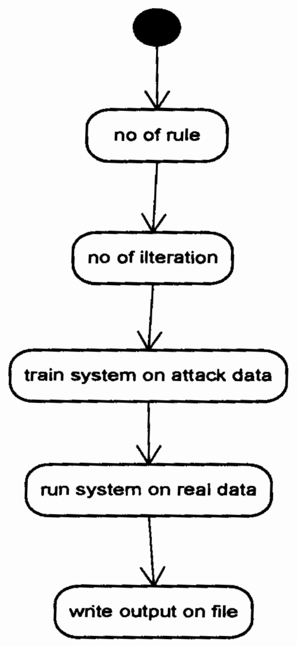


Figure 9: Activity Diagram of Probe

4.6. Flow Diagrams

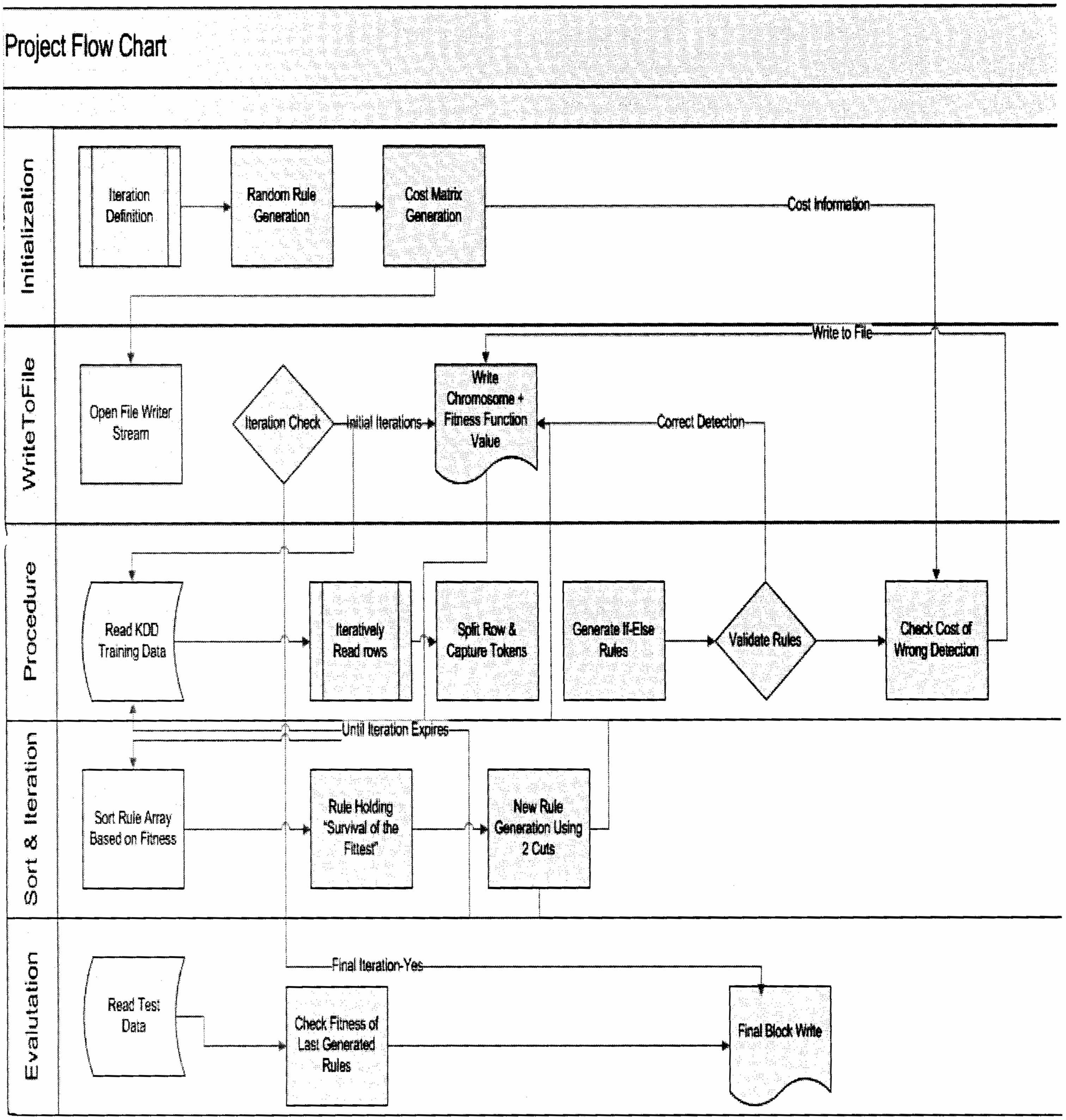


Figure 10: Project Flow Diagram

Chapter 5
Experimental Results

5. Experimental Results

The overall o1bjective of the testing process is to identify the maximum number of errors in the code with a minimum amount of efforts. Finding an error is thus considered a success rather than failure. On finding an error, efforts are made to correct it.

5.1. Test Plan

5.1.1. Introduction

In this section we are going to perform different tests on our system in order to make it sure if the system is working properly or not. We have used IEEE 829-1983 for Software Test Documentation for testing of our system.

5.1.2. Test Items

We will test the following items for our testing:

- DOS module
- Probe module
- R2L module
- U2R module
- Input dialog box for each module
- Output files for each module

Features to be tested:

They will be described on later.

Features not to be tested:

We cannot test the following features:

- We are unable to test the rules if they are really the best rules or not.

- Comparison testing can't be performed for the items. Because comparable software are not available.

5.1.3. Item Pass/Fail Criteria

A test case is considered to be pass if the test case exhibits an expected result. OtherwiseIt's considered to be fail.

5.1.4. Environment in which test cases will run

1. A Pentium® IV or higher machine.
2. Microsoft® Windows XP

5.1.5. Test Design specification

Following features will be tested in our testing according to the standard we have chosen:

Input Dialog for DOS Rule generator module

Output file for DOS Rule generator module

Input Dialog for Probe Rule generator module

Output file for Probe Rule generator module

Input Dialog for U2R Rule generator module

Output file for U2RRule generator module

Input Dialog for R2L Rule generator module

Output file for R2L Rule generator module

Number of iterations and number of rules in output files

Stress testing for each module

5.1.6. Test Case Specification

Following are the test cases to be performed on our system and the body of test case is also defined:

5.1.6.1. Input Dialog for DOS Rule Generator

Test Items

Dialog that asks for # of rules for DOS module

Dialog which asks for # of iterations for DOS module

Input Specifications

Run the system for DOS module.

Output Specifications

A dialog box should appear that should ask the user to enter the number of rules. After that it should ask for the number of iterations for DOS module.

Environmental Needs

3. A Pentium® IV or higher machine.
4. Windows 2000 (Family)

Special Procedural Requirements

It should be able to take the number of rules and number of iterations.

Inter-case Dependencies

None

5.1.6.2. Output file for DOS Rule generator module

Test Items

Output file that results after running the DOS file

Input Specifications

Run the system for DOS module.

Look for the iterations to complete.

Output Specifications

An output file should be created that tells about all the iterations performed according to input given at the start. And this is the raw representation of the rules generated as a result of our genetic algorithm.

Environmental Needs

5. A Pentium® IV or higher machine.
6. Windows 2000 (Family)

Special Procedural Requirements

An output file should be generated that contains our results.

Inter-case Dependencies

None

5.1.6.3. Input Dialog for Probe Rule Generator

Test Items

Dialog that asks for # of rules for Probe module

Dialog which asks for # of iterations for Probe module

Input Specifications

Run the system for Probe module.

Output Specifications

A dialog box should appear that should ask the user to enter the number of rules. After that it should ask for the number of iterations for Probe module.

Environmental Needs

7. A Pentium® IV or higher machine.
8. Windows 2000 (Family)

Special Procedural Requirements

It should be able to take the number of rules and number of iterations.

Inter-case Dependencies

None

5.1.6.4. Output file for Probe Rule generator module

Test Items

Output file that results after running the Probe file

Input Specifications

Run the system for Probe module.

Look for the iterations to complete.

Output Specifications

An output file should be created that tells about all the iterations performed according to input given at the start. And this is the raw representation of the rules generated as a result of our genetic algorithm.

Environmental Needs

9. A Pentium® IV or higher machine.
10. Windows 2000 (Family)

Special Procedural Requirements

An output file should be generated that contains our results.

Inter-case Dependencies

None

5.1.6.5. Input Dialog for U2R Rule Generator

Test Items

Dialog that asks for # of rules for U2R module

Dialog which asks for # of iterations for U2R module

Input Specifications

Run the system for U2R module.

Output Specifications

A dialog box should appear that should ask the user to enter the number of rules. After that it should ask for the number of iterations for U2R module.

Environmental Needs

11. A Pentium® IV or higher machine.
12. Windows 2000 (Family)

Special Procedural Requirements

It should be able to take the number of rules and number of iterations.

Inter-case Dependencies

None

5.1.6.6. Output file for U2R Rule generator module

Test Items

Output file that results after running the U2R file

Input Specifications

Run the system for U2R module.

Look for the iterations to complete.

Output Specifications

An output file should be created that tells about all the iterations performed according to input given at the start. And this is the raw representation of the rules generated as a result of our genetic algorithm.

Environmental Needs

13. A Pentium® IV or higher machine.
14. Windows 2000 (Family)

Special Procedural Requirements

An output file should be generated that contains our results.

Inter-case Dependencies

None

5.1.6.7. Input Dialog for R2L Rule Generator

Test Items

Dialog that asks for # of rules for R2L module

Dialog which asks for # of iterations for R2L module

Input Specifications

Run the system for R2L module.

Output Specifications

A dialog box should appear that should ask the user to enter the number of rules. After that it should ask for the number of iterations for R2L module.

Environmental Needs

15. A Pentium® IV or higher machine.
16. Windows 2000 (Family)

Special Procedural Requirements

It should be able to take the number of rules and number of iterations.

Inter-case Dependencies

None

5.1.6.8. Output file for R2L Rule generator module

Test Items

Output file that results after running the R2L file

Input Specifications

Run the system for R2L module.

Look for the iterations to complete.

Output Specifications

An output file should be created that tells about all the iterations performed according to input given at the start. And this is the raw representation of the rules generated as a result of our genetic algorithm.

Environmental Needs

17. A Pentium® IV or higher machine.
18. Windows 2000 (Family)

Special Procedural Requirements

An output file should be generated that contains our results.

Inter-case Dependencies

None

5.1.6.9. Number of iterations and number of rules in each output file

Test Items

# of iterations and # of rules in each file.The data is given in raw form

Input Specifications

Input system with # of iterations and # of rules.

System will create an output file

Manually check the # of iterations and # of rules for each module.

Output Specifications

The number of iterations and number of rules should be same as entered by the user.

Environmental Needs

19. A Pentium® IV or higher machine.

20. Windows 2000 (Family)

Special Procedural Requirements

The count of number of iterations and rules should be the same as entered in the start.

Inter-case Dependencies

None

5.1.6.10. Stress testing for each module

Test Items

DOS, Probe, U2R, R2L modules

Input Specifications

Input system with a large # of iterations and # of rules.

Look for the performance.

Output Specifications

The number of iterations and number of rules should be same as entered by the user.

Performance of the system got slow.

Environmental Needs

21. A Pentium® IV or higher machine.
22. Windows 2000 (Family)

Special Procedural Requirements

The count of number of iterations and rules should be the same as entered in the start.

Inter-case Dependencies

None

We had run our application for different number of parameters. And this leads us to different types of results. And if we judge the correlation between given or input parameters then we can also got bit of idea of the algorithm efficiency. Like we can input our program different number of iterations as well as different number of total rules; Conducted test yield that if we increase any of these then it will have a positive effect on the algorithm accuracy as well as quality. As describe earlier that we have actually two dataset one *(kddcup.data_10_percent)* on which we supervise or perform learning of our algorithm and second one (*corrected*) on which we test our data. There some interesting result shown below.

5.2. Results For Training Dataset with 5 iterations

Number of Iterations=5														
# of Rules	**Corresponding Rule**												**Total Cost**	**% Correct**
10	0	1	0	0	0	0	1	1	0	1	0	0	391458	20.76
150	1	0	1	0	1	1	1	1	1	1	1	0	90243	71.73

200	1	0	1	0	0	1	1	1	1	1	1	1	49629	89.95
500	1	0	1	0	1	1	1	1	1	1	1	1	49604	89.995

Table 7: percentage of correctness for training dataset when number of iterations=5

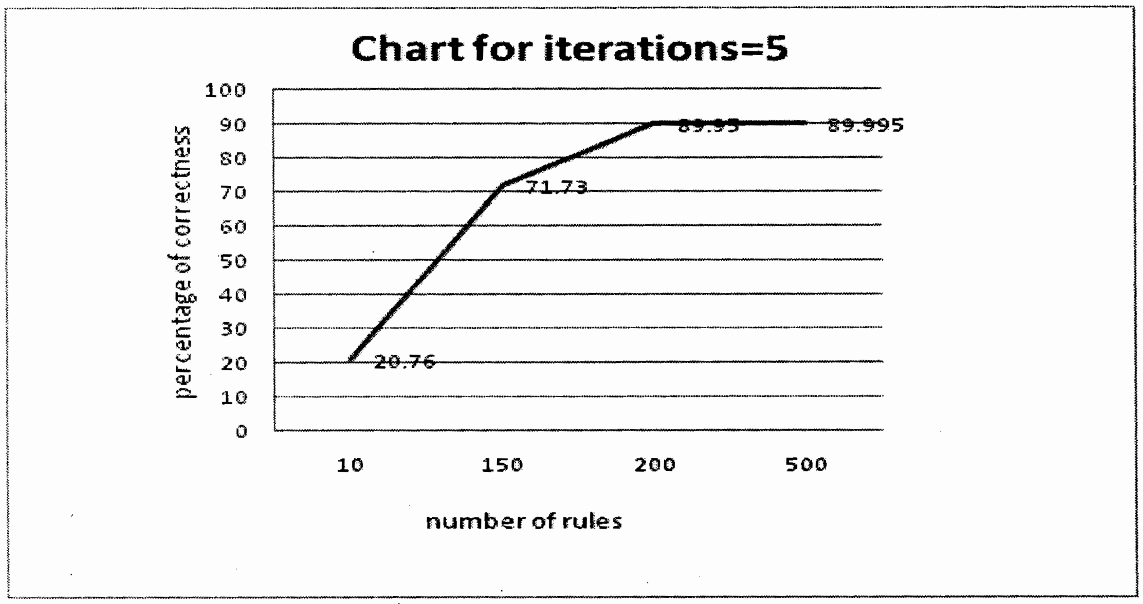


Figure 11: Graph for training data when number of iterations are 5

As we see as the number of rules increases the percentage of correctness also increases. But as going in ascending order the number of rules the rapid increase in correctness percentage becomes slower. Another way to represent above scenario is as follows

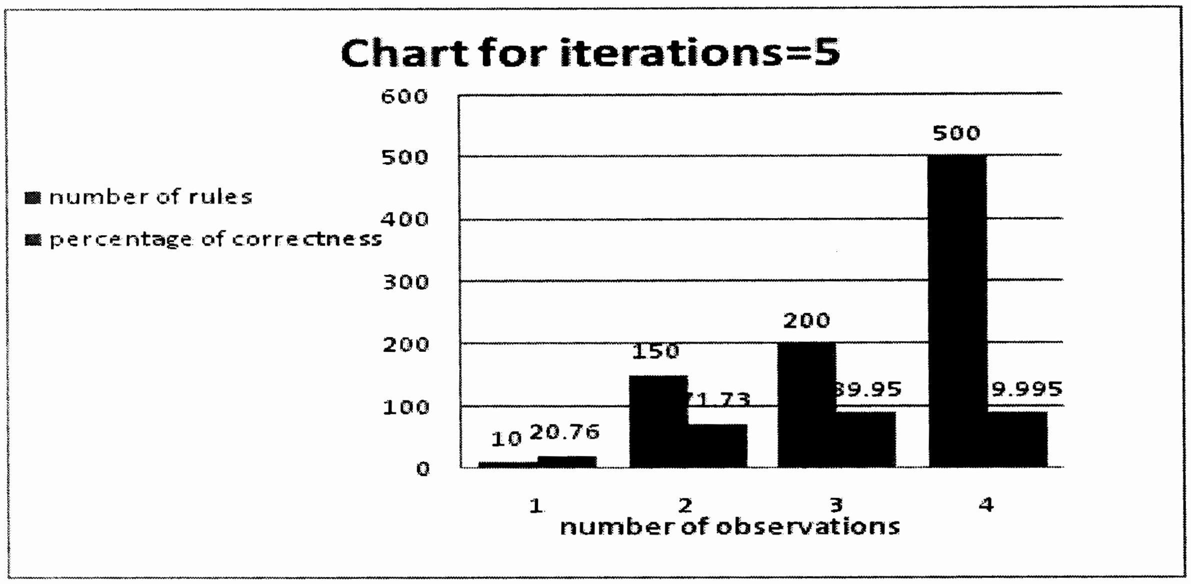


Figure 12: Another representation of above graph

5.3.Result For Corrected Dataset when iterations are 5

Number of Iterations=5														
# of Rules	Corresponding Rule												Total Cost	% Correct
10	0	1	0	0	0	0	1	1	0	1	0	0	223298	28.20
150	1	0	1	0	1	1	1	1	1	1	1	0	61738	80.15
200	1	0	1	0	0	1	1	1	1	1	1	1	53633	82.77
500	1	0	1	0	1	1	1	1	1	1	1	1	52999	82.96

Table 8: Percentage of correctness for corrected dataset when iterations =5

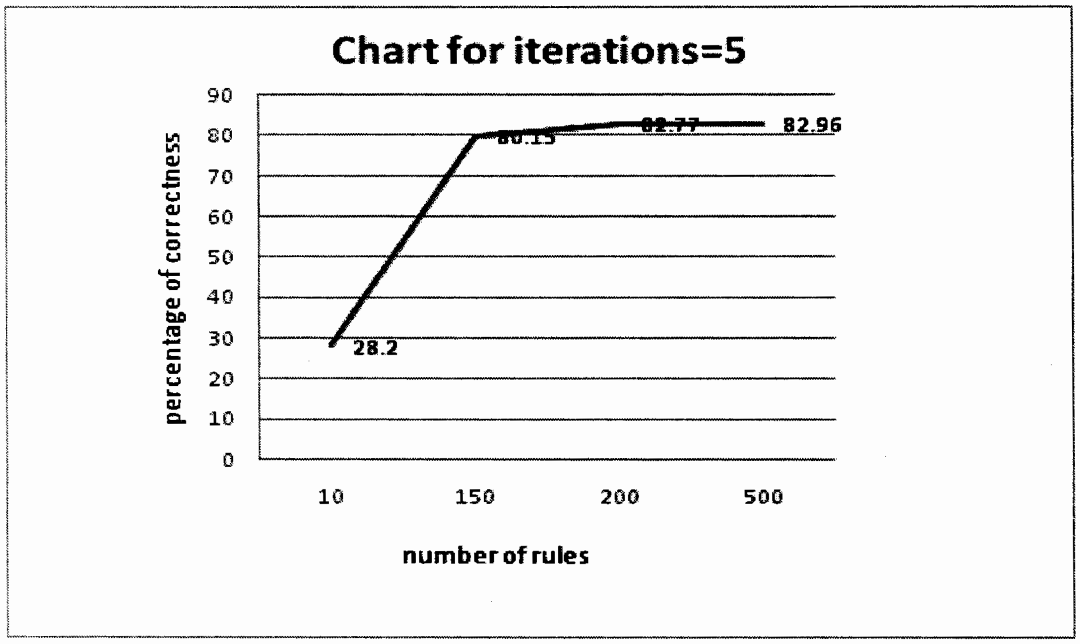


Figure 13 : Graph for percentage of correctness for corrected dataset when iterations=5

It also depicts the same relationship as in the training dataset.

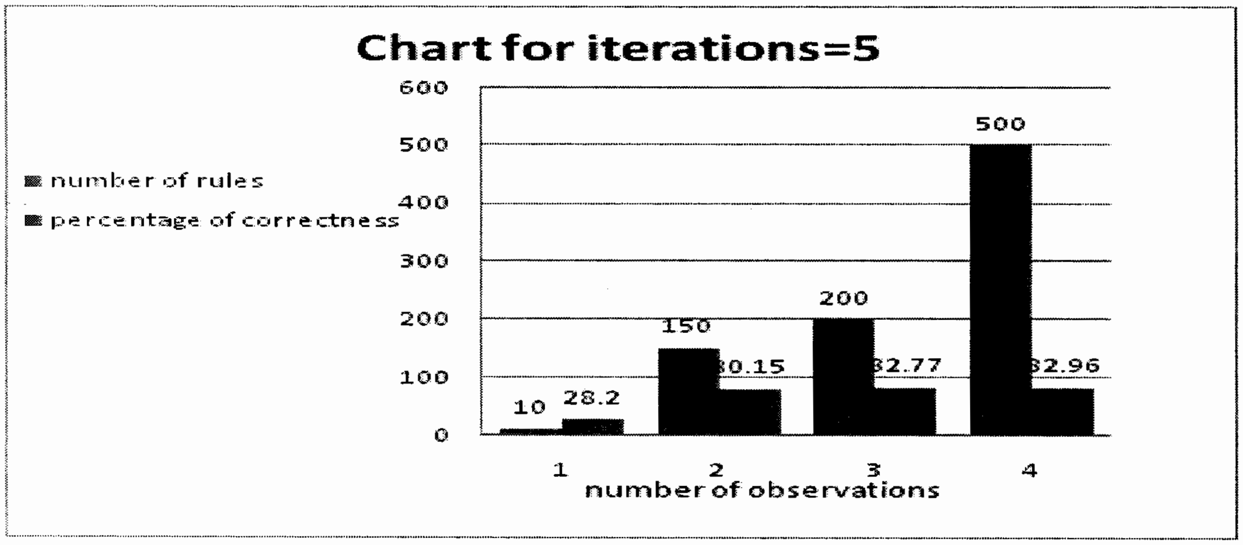


Figure 14 : Another representation of percentage of correctness for corrected dataset when iterations=5

5.4. Result For Training Dataset when iterations are 10

Number of Iterations=10														
# of Rules	Corresponding Rule												Total Cost	%Correct
10	1	1	1	0	0	0	1	1	1	1	1	1	165141	66.65
50	1	1	1	0	1	1	1	0	0	1	1	1	164899	66.62
150	1	0	1	0	0	1	1	0	0	1	1	1	88269	82.13
200	1	1	1	1	1	1	1	0	1	1	1	1	97418	80.28
500	1	1	1	1	0	1	1	1	1	1	1	0	49674	89.94

Table 9: Percentage of correctness for training dataset when iterations =10

Now we have increased the number of iterations too from 5 to 10. And the result above and graph below speaks itself that degree of correctness of finding an intrusion is better than previous. But we will also see that same behavior will be true also for this dataset for number of iterations

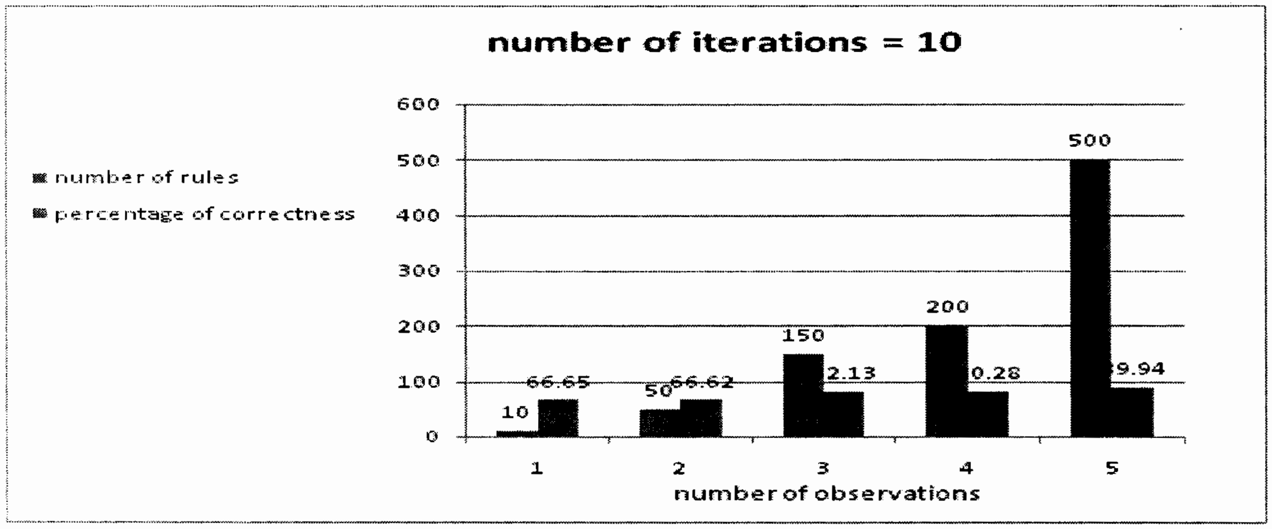


Figure 15 : Graph representation of percentage of correctness for training dataset when iterations are 10

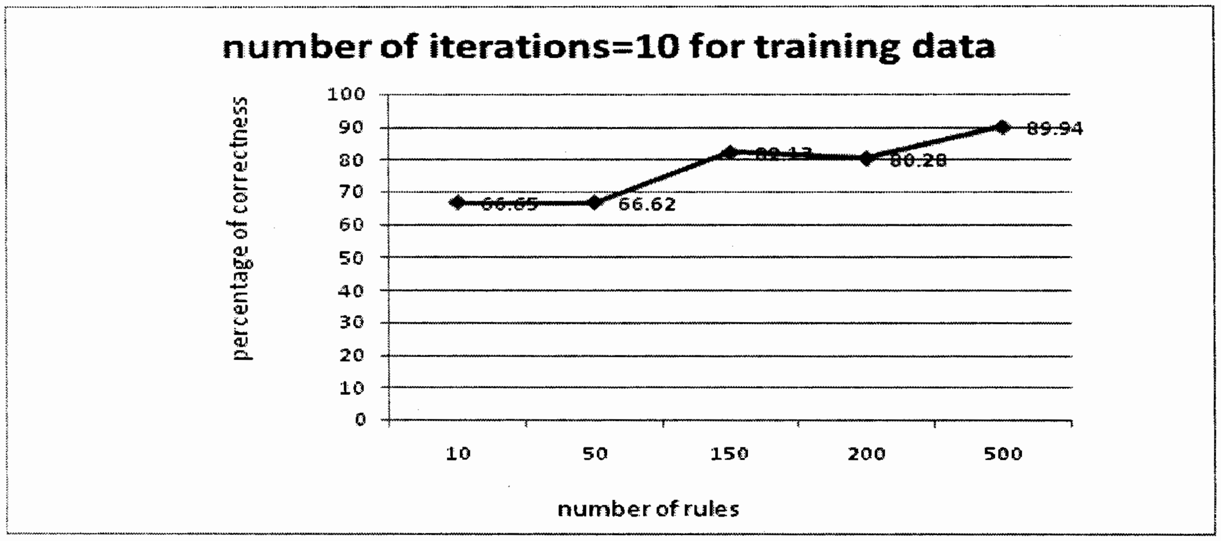


Figure 16 : Graph representation of percentage of correctness for training dataset when iterations are 10

5.5.Result For Corrected Dataset when iterations are 10

Number of Iterations=10														
# of Rules	**Corresponding Rule**												**Total Cost**	**% Correct**
10	1	1	1	0	0	0	1	1	1	1	1	1	122213	60.71
50	1	1	1	0	1	1	1	0	0	1	1	1	121977	60.78
150	1	0	1	0	0	1	1	0	0	1	1	1	72441	76.70
200	1	1	1	1	1	1	1	0	1	1	1	1	57327	81.56
500	1	1	1	1	0	1	1	1	1	1	1	0	35553	88.56

Table 10: Percentage of correctness for corrected dataset when iterations =10

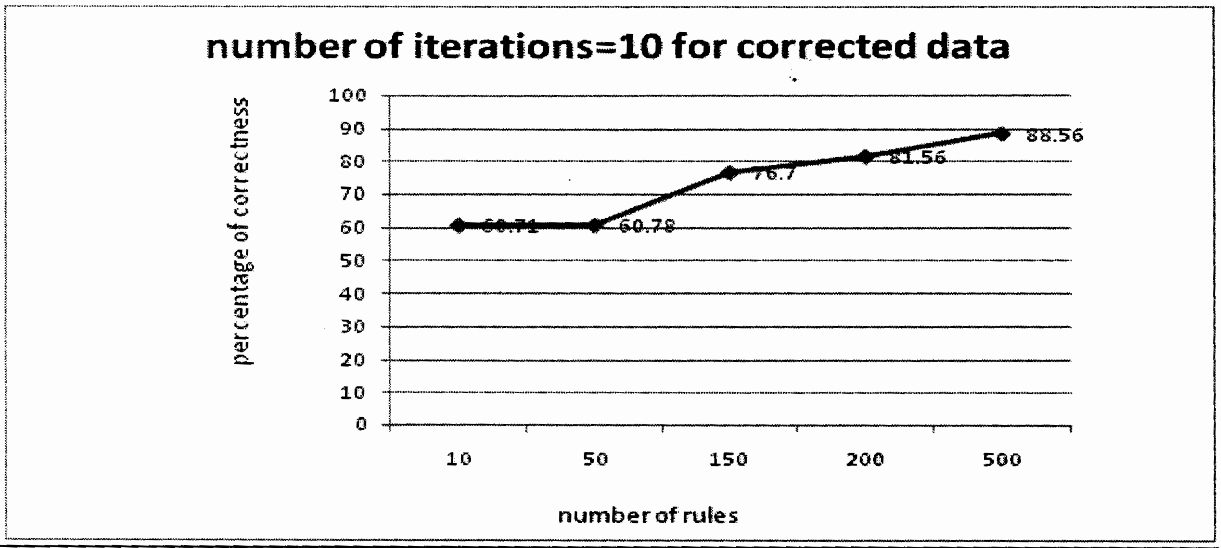


Figure 17 : Graph representation of percentage of correctness for corrected dataset when iterations are 10

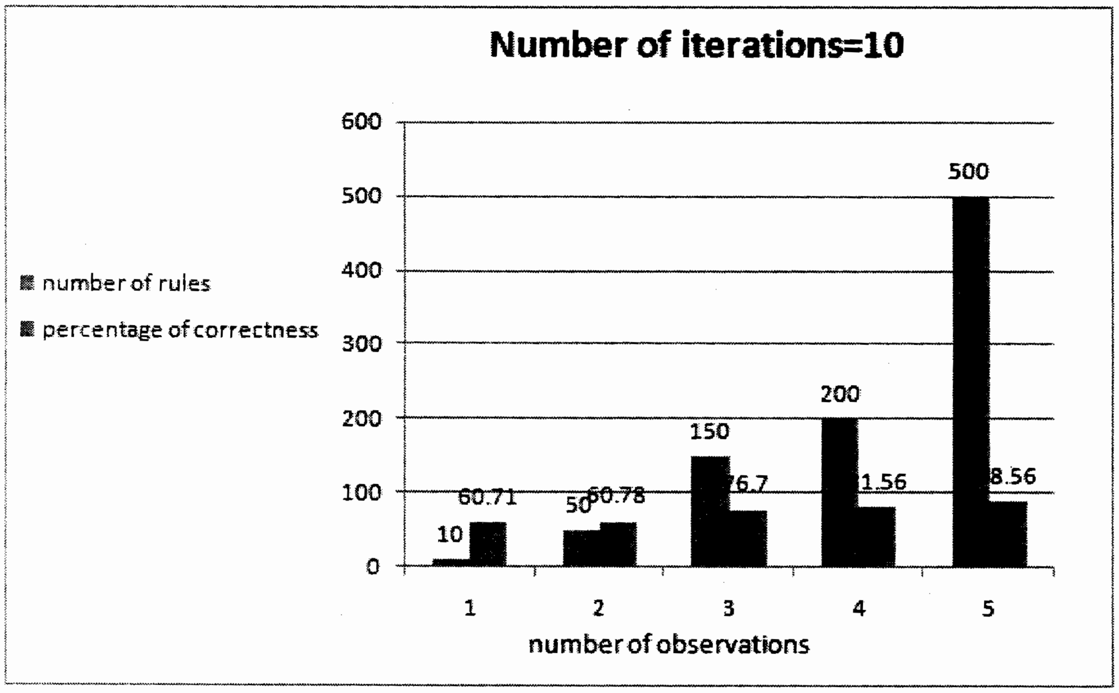


Figure 18 : Graph representation for percentage of correctness for corrected dataset when iterations are 10

5.6.Results For DOS

As we seen both in graph as well as in table format that all of them follow the same pattern that if we incr eas

	DOS	Not DOS
DOS	348319	11848

e the number of rules then it's more likely that we will detect more attack.

Here how we check that whether its attack or not.

If the output will come out true then we will declare that is intrusion otherwise we will say that it's not this type of attack.

Then we will check for do we got it wrong or correct. If we do have wrong then we got punishment of cost.

5.7. Results For Probe

Iterations = 50		
# of Rules	Total Cost	% Correct
200	For Training data	97.55
200	For Corrected data	97.66

Table 11 : Percentage of correctness for Probe when iterations are 50

5.8. Results For R2L

Iterations = 50		
# of Rules	Total Cost	% Correct

200	For Training data	97.79
200	For Corrected data	96.44

Table 12: Percentage of correctness for R2L when iterations are 50

5.9. Results For U2R

Iterations = 50		
# of Rules	**Total Cost**	**%Correct**
200	For Training data	97.88
200	For Corrected data	97.99

Table 13: Percentage of correctness for U2R when iterations are 50

5.10. Results with Respect to Time

It will take four times of the time of our system, if all attributes are considered and four times more if all the types of attacks are detected using one project file. Thus, overall we have reduced the time eight times as compared to 41 attributes and all files handled in same project file.

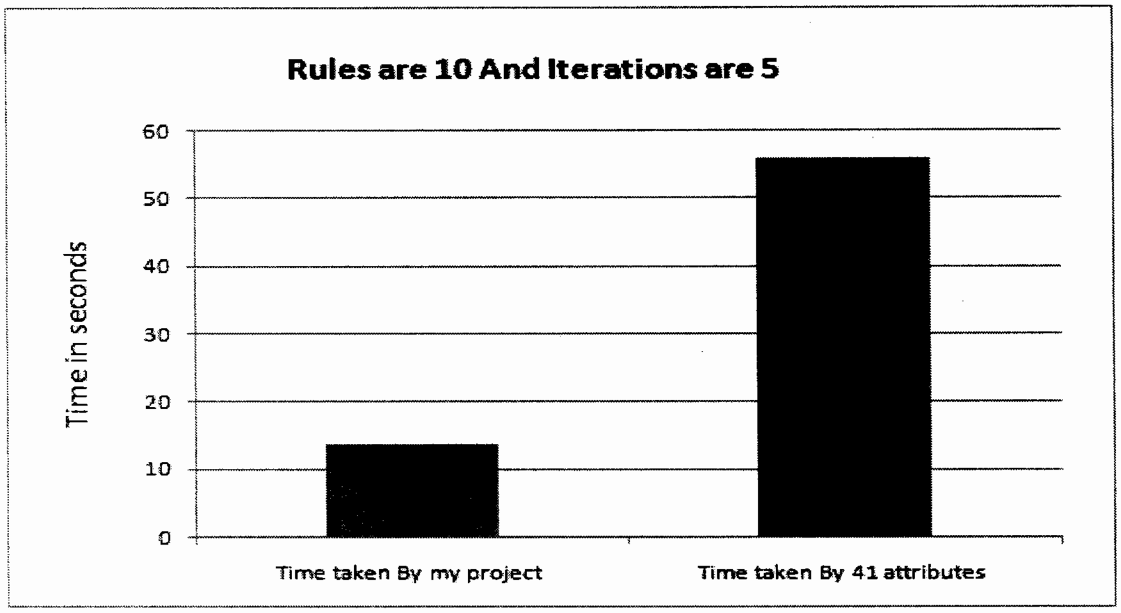


Figure 19: Time taken by my approach and time taken by all 41 attributes when rules are 10 and iterations are 5

Chapter 6
Conclusion

6. Conclusion

I have got promising results for misuse detection by using genetic algorithms, although few results were not expected. By using 8 features in our program, we have got better results than those of 5 features but it takes more cost.

Genetic Algorithm is based on extensive learning technique so more the learning and more appropriate are the results and for this reason, high value of iteration results in better performance. Number of Rules is another factor more the rules more generations will be generated and results in more chances of choosing of best results.

DOS has most number of attacks in given dataset and that's why its stats represent more realistic approaches.

6.1. Future Work

This offline rule generator should run at the front end of an hybrid implementation of Anomaly based intrusion detection and Misuse detection. Generating the training data form anomaly based detector is also an important part of the task. By doing so we can update our misuse detector such that it can detect new types of attacks as well in such a way that one may benefit the strong points of anomaly based detection as well.

6. Conclusion

I have got promising results for misuse detection by using genetic algorithms, although few results were not expected. By using 8 features in our program, we have got better results than those of 5 features but it takes more cost.

Genetic Algorithm is based on extensive learning technique so more the learning and more appropriate are the results and for this reason, high value of iteration results in better performance. Number of Rules is another factor more the rules more generations will be generated and results in more chances of choosing of best results.

DOS has most number of attacks in given dataset and that's why its stats represent more realistic approaches.

6.1. Future Work

This offline rule generator should run at the front end of an hybrid implementation of Anomaly based intrusion detection and Misuse detection. Generating the training data form anomaly based detector is also an important part of the task. By doing so we can update our misuse detector such that it can detect new types of attacks as well in such a way that one may benefit the strong points of anomaly based detection as well.

REFERENCES

[1] G. Weiss, "*Mining with rarity: A unifying framework*", SIGKDD Explorations ,2004

[2].D. E. Goldberg, "*Genetic algorithms for search, optimization, and machine learning.*" Addison-Wesley91, 1989

[3] J. Kim and P. Bentley. "*Immune system approaches to intrusion detection –a review*", Aachen: Germany, 2007.

[4] D. Song, Malcolm I. Heywood, A. Nur Zincir-Heywood" *Training Genetic Programming on Half a Million Patterns: An Example From Anomaly Detection*" IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 9, NO. 3, JUNE 2005

[5] M. J. Middlemiss. Grant Dick, "*Weighted Feature Extraction using a Genetic Algorithm for Intrusion Detection*",2003

[6]. *Vulnerability analysis of immunity-based intrusion detection systems using genetic and evolutionary hackers*

[7]. V. D. Kotov ,Vladimir I. Vasilyev *Artificial Immune Systems Based Intrusion Detection System*"Ufa State Aviation Technical University,2009

[8] J. W. Kim. "*Integrating Artificial Immune Algorithms for Intrusion Detection.*" PhD thesis, University College London, 2002.

[9]. Z. Bankovic, J. M. Moya, Álvaro Araujo, Slobodan Bojanic and Octavio Nieto-Taladriz "*A Genetic Algorithm-based Solution for Intrusion Detection*" 2006

[10] .Z. Bankovic, J. M. Moya, Álvaro Araujo, Slobodan Bojanic and Octavio Nieto-Taladriz." *Improving network security using genetic algorithm approach*, 2007.

[11] A.Chittur " *Model Generation for an Intrusion Detection System Using Genetic Algorithms",* 2006.

[12] Z.Pan , S.Chen , Hu G, Zhang D. *Hybrid Neural Network and C4.5 for Misuse Detection. In: Proceedings of the second international conference on machine learning and cybernetics,* vol. 4. 2003

[13] S.Folino , Pizzuti C, Spezzano G. GP ensemble for distributed intrusion detection systems. In ICAPR 2005, 3rd *international conference on advances in pattern recognition*, LNCS, Springer Verlag, 3686/2005, Bath, UK, August 2005.

[14] P. Laskov, P. Düssel, C. Schäfer and K. Rieck, "*Learning intrusion detection: supervised or unsaupervised?, CIAP: international conference on image analysis and processing"* Cagliari : Italy (2005) September.

[15] Y.Guan , Ghorbani AA, Belacel N, Y-means. *"A Clustering method for Intrusion Detection. In Canadian conference on electrical and computer engineering"*, IEEE CCECE, vol. 2. 2003

[16] D.S. Kim, Ha-Nam Nguyen, Jong Sou Park. "*Genetic algorithm to improve SVM based network intrusion detection system. In: Proceedings of the 19th international conference on advanced informational networking and applications"*, vol. 2. 2005

[17] J.T.Yao, S.L.Zhao, l.V.Saxton. "*A study on fuzzy intrusion detection. data mining, intrusion detection, information assurance, and data networks security"* 2005. Orlando FL: 2005

[18] Y. Bouzida and S. Gombault, Eigenconnections to intrusion detection," *Proceedings of the 19th IFIP international information security conference"*, Kluwer Academic (2004) August.

[19] S.S.Joshi ,V.V. Phoha." *Investigating hidden Markov model capabilities in anomaly detection"*. ACM Regional Conference. In Proceedings of the 43rd annual southeast regional conference, vol. 1, 2005; p. 98–103.

[20] *Cost base Scoring, " http://wwwcse.ucsd.edu/users/elkan/clresults.html. "*

[21]. S. Ashfaq, M. Umar Farooq, A. Karim", *Efficient Rule Generation for Cost-Sensitive Misuse Detection* using *Genetic Algorithms*",1-4244-0605-6/06/,2006

[22]. W. Li, "*Using Genetic Algorithm for Network Intrusion Detection||, Proceedings of the United States Department of Energy Cyber Security Group"*,2004

[23]. R. H. Gong, M. Zulkernine, and Purang, "*A software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection"*||, SNPD/SAWN'05, IEEE, 2005.

[24] Y. Chen, A. Abraham, B. Yang., "*Hybrid Flexible Neural-Tree-Based Intrusion Detection Systems // International Journal of Intelligent Systems*", 2007

[25] Dong Seong Kim, Ha-Nam Nguyen, Jong Sou Park, —*Genetic Algorithm to Improve SVM Based Network Intrusion Detection System //*, AINA'05, IEEE, 2005