جـامـعـة الـشـرق الأوسـط
MIDDLE EAST UNIVERSITY

An Enhanced Steady State Genetic Algorithm Model for Misuse Network Intrusion Detection System

نموذج محسن للخوارزمية الجينية المستقرة لاكتشاف التطفل في الشبكات الحاسوبية

By:

Firas Mohammad Ahmad AlAbsi

Supervisor:

Prof. Reyadh Shaker Naoum

A master thesis submitted in Partial Fulfillment of the Requirements for the Master Degree in Computer Science
Computer Science Department
Faculty of Information Technology
Middle East University

(Jul 2012)

Middle East University

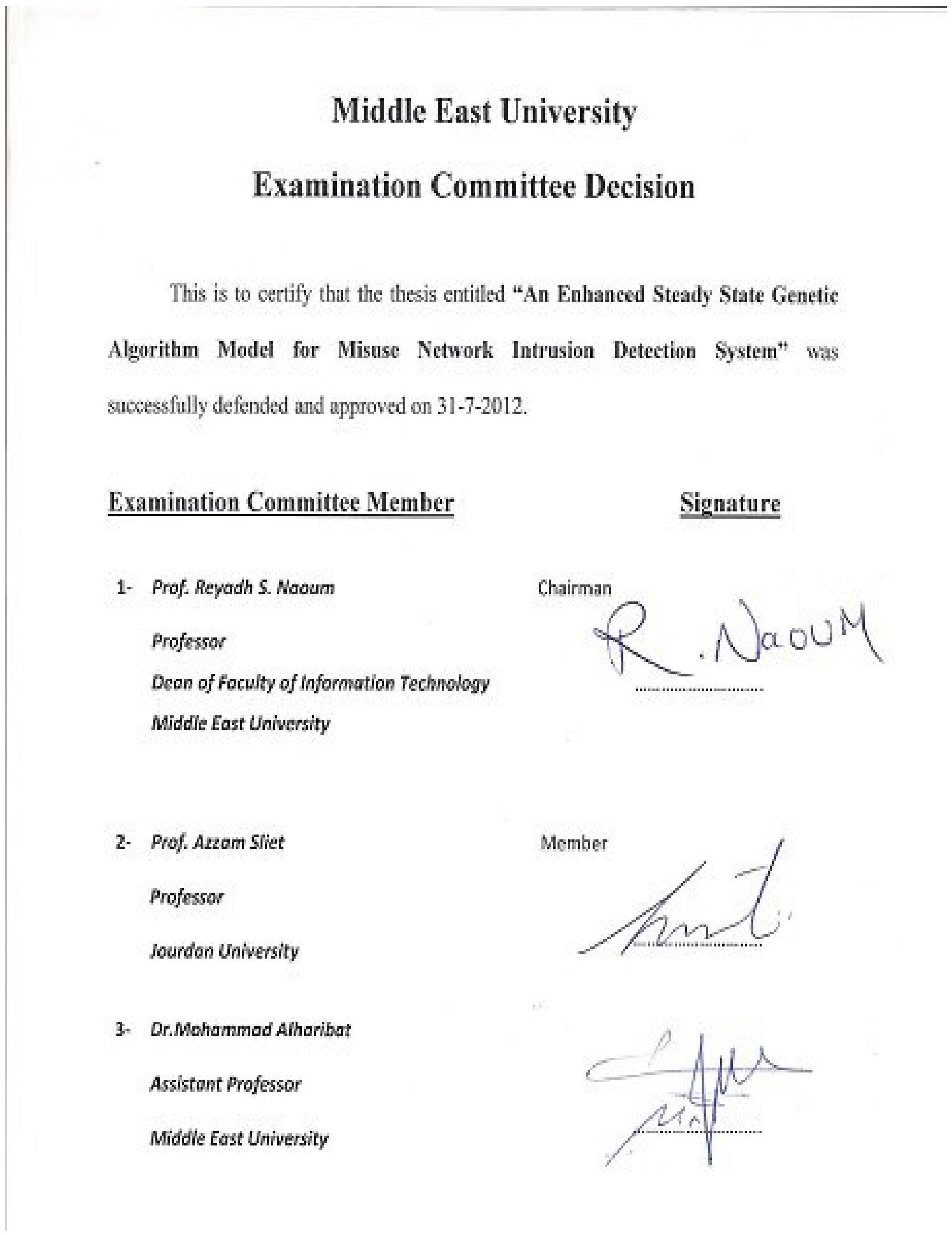
Authorization Statement

I am, Firas Mohammad Ahmad Alabsi, authorize Middle East University to supply hardcopies and electronic copies of my thesis to libraries, establishments, or bodies and institutions concerned with research and scientific studies upon request, according to the university regulations.

Name : Firas Mohammad Ahmad Alabsi

Date : 31-7-2012

Signature : [illegible]



Middle East University

Examination Committee Decision

This is to certify that the thesis entitled **"An Enhanced Steady State Genetic Algorithm Model for Misuse Network Intrusion Detection System"** was successfully defended and approved on 31-7-2012.

Examination Committee Member	**Signature**
1- *Prof. Reyadh S. Naoum* *Professor* *Dean of Faculty of Information Technology* *Middle East University*	Chairman R. NaouM
2- *Prof. Azzam Sliet* *Professor* *Jourdan University*	Member [illegible]
3- *Dr.Mohammad Alharibat* *Assistant Professor* *Middle East University*	[illegible]

Acknowledgments

I would like to extend my sincerest thanks and appreciation to my father and my mother, who supported me in my scientific life, and who I will not forget their virtue all my life. And I would like to extend my sincerest thanks and appreciation to my wife (Hala) who encouraged me and helped me to overcome the difficulties. I also commend my daughter who missed my time in order to finish this work. All my love to my brothers: Hosam, Moayad, Moaath and my sister Doaa.

I Admit that this work would not being accomplished without the effort of Prof Reyadh Shaker Naoum, who clarified the domain of this work to me. He has the right to be kind enough to accept my sincere respect and appreciation.

Finally, special thanks to the management of Middle East University, which established a customized environment for learning to its students.

To whom helped me, and was with me in heart and mind: I love you.

شكر وتقدير

أتقدم بجزيل الشكر والتقدير إلى والدي ووالدتي، اللذان كانوا عمادا لي في مسيرتي العلمية، واللذان لا أنسى فضلهما علي ما حييت، كما أتقدم بجزيل الشكر لزوجتي هالة التي شجعتني وكافحت معي وساعدتني في تذليل الصعوبات، كما أثني على طفلتي ريماس التي حرمتها الكثير من وقتي حتى أنجز هذا العمل، وكل المحبة لإخوتي حسام ومؤيد ومعاذ وأختي دعاء.

وأقر وأعترف أن هذا العمل لم يكن لينجز بدون جهد الأستاذ الدكتور رياض شاكر نعوم، الذي أوضح لي معالم هذا الطريق، فله الحق في أن يتقبل مني فائق الاحترام والتقدير.

وفي النهاية، شكر خاص لإدارة جامعة الشرق الأوسط التي توفر بيئة تعليمية متميزة لطلابها.

إلى كل من ساعدني.. وكان معي قلبا وقالبا أقول: إني أحبكم.

Dedication

I dedicate this work to my father, my mother, my wife and partner of my life (Hala), my daughter (Remas), my brothers and sister; for their love, understanding and support. They were the light in my path. Without them nothing of this would have been possible. Thank you for everything. I love you!

Table of Contents

List of Tables

List of Figures

List of Equations

List of Abbreviations

DoS	Denial Of Service
DR	Detection Rate
FPR	False Positive Rate
IDS	Intrusion Detection System
ID	Intrusion Detection
GA	Genetic Algorithm
KDD	Knowledge Data Discovery
NIDS	Network Intrusion Detection System
NN	Neural Network
R2L	Remote to Local
SGA	Simple Genetic Algorithm
SSGA	Steady State Genetic Algorithm
U2R	User to Root

List of Appendices

Abstract

The networks usage has been increased in the last decades. The intruders began to do violations and abuses over the networks. This had led the researchers to do additional researches to support Intrusion Detection Systems. The main aim of this thesis is to build Intrusion Detection System supported by enhanced Steady State Genetic Algorithm in order to increase Detection Rate and to decrease False Positive Rate.

This proposed research proved Reward Penality based Fitness Function to be used in the evaluation process. It also compared selection and crossover to choose the best choice to implement it in a system; it was found that Stochastic Universal Sampling Selection can be used with Uniform Crossover to produce the best results. This research was applied Stochastic Universal Sampling Selection and Uniform Crossover as parameters in Steady State Genetic Algorithm to be used in Network Intrusion Detection System.

In this thesis an enhancement has been done to the algorithm by using Reward-Penality based Fitness Function and choosing the best choice for selection and crossover; this has affected the Misuse based Network Intrusion Detection System by increase DR to be equal 95% and decrease FPR to be equal 0.297%.

الملخص

لقد تزايد استخدام الشبكات في القرون الماضية. وبدأ المتطفلون بإحداث الانتهاكات والتجاوزات في الشبكات. الأمر الذي أدى إلى إقبال الباحثين لإجراء دراسات لتقوية أنظمة التطفل. والهدف من هذه الرسالة هو بناء نظام لاكتشاف التطفل مدعّم بالخوارزمية الجينية المستقرة المحسّنة من أجل زيادة نسبة الالتقاط الصحيح وتقليل نسبة الالتقاط الخاطىء.

هذا البحث المقترح أثبت اقتران الصلاحية المعتمد على الثواب والعقاب لاستخدامه في عملية التقييم. وأيضا قام بمقارنة الاختيار والتزاوج لاعتماد أفضل خيار لتطبيقه في النظام، حيث أنه وجد أن انتقاء العينات الشاملة التصادفية بالإمكان استخدامه مع التزاوج المنتظم لإعطاء أفضل النتائج وقد تم في هذا البحث استخدام انتقاء العينات الشاملة التصادفية والتزاوج المنتظم كعوامل للخوارزمية الجينية المستقرة لاستخدامها في نظام كشف التطفل.

في هذه الرسالة تم تحسين الخوارزمية الجينية باستخدام اقتران الصلاحية المعتمد على الثواب والعقاب واختيار الخيار الأفضل للانتقاء والتزاوج، هذا العمل أثر على نظام اكتشاف التطفل في الشبكات الحاسوبية بزيادة معدل الالتقاط ليصبح 95% وتقليل معدل الالتقاط الخاطىء ليصبح 0.297%.

Chapter One

Introduction

1.1 Preface

Networks security has the researcher's attention because of the increase of networks usage. Network Intrusion Detection System (NIDS) is an important issue of network security. NIDS designed to be strong enough to ensure secure environment. It must detect different attack types. And to keep the (NIDS) as secure as need, the system must detect intrusions with high Detection Rate (DR) and low False Positive Rate (FPR). Steady State Genetic Algorithm (SSGA) is one of the most important schemes used to solve this problem.

1.2 Problem Identification

In the last two decades, computer specialists noticed the tremendous growth of networks usage. Networks usage gives the chance to intruders to attack these networks in somehow. The attacks led to highlight the network security. (NIDS) is one of the main components on network security. If the (NIDS) is strong, this will increase the trustworthy of the system and decrease the probability of system misuse and malicious attacks. And to keep the (NIDS) as secure as the need, this thesis must escalate the challenge against the intruders by detecting intrusions with increase Detection Rate (DR) and decrease False Positive Rate (FPR). One of the most important and new schemes which was used to solve this problem is Steady State Genetic Algorithm (SSGA). This thesis tried to use enhanced Steady State Genetic Algorithm (SSGA) to examine the (DR) and (FPR) of different network intrusions.

The research questions are:

- How to escalate the challenge of the intruders?
- How to give the strength and trustworthy to Misuse Network Intrusion Detection Systems?
- How to detect intrusions with high Detection Rate and low False Positive Rate?
- Is that optimal to use SSGA in dealing with Misuse Network Intrusion Detection Systems?
- In case of comparing the results (Misuse Based result) to other results (Anomaly Based Result), are that the same results in detection rate?
- In case of comparing the results during changing Genetic Algorithm operator's types, what is the most perfect type for each operator to detect intrusions through misuse network intrusion detection system?

1.3 Contributions

The main goal of the research is to apply SSGA to detect intrusions in the network environment under misuse analysis.

The researcher did the following:

1- Enhance the Steady State Genetic Algorithm for Misuse NIDS.

2- Use SSGA to increase DR and decrease FPR to produce new results.

3- Compare the produced results with the previous results.

1.4 Significance

The importance of the research is because it solves a new issue. It will support the security. It will expand the results in the field of dealing with intrusions. Many of the researchers did their researches on NIDS using anomaly analysis, but there are a few researches that used GA in NIDS using misuse analysis, and this is the main reason which gives the importance to the research.

This research may increase the strength and trustworthy of intrusion detection system with high detection rate and low false positive rate.

1.5 Limitations

- The research will solve the problem of Network-based IDS, not host-based.
- The research will be applied on misuse-IDS only, not on anomaly.
- The systems can play many actions to deal with intrusions; these actions will be as detection, prevention or both. The research will deal only with detection systems.

1.6 Thesis Outline

Chapter Two: This chapter views some knowledge about Intrusion Detection System (elements, classes of detection technologies and network attacks), Genetic Algorithm (elements and operator types) and finally literature review and related work.

Chapter Three: Proposed model shows the stages of Intrusion Detection System, and the elements of Steady State Genetic Algorithm. The methodology that has been followed to find the results was discussed.

Chapter Four: This is the most important chapter because it shows in details the results achieved such as: how the most significant features have been selected, why Reward Penality based fitness function has been used, the results of tracing of different

Selection methods, the comparison of selection and crossover and the results of DR and FPR for the Misuse Network Intrusion Detection System.

Chapter Five: In this chapter, the conclusion of thesis has been presented, and also future work related to this thesis domain.

Chapter Two
Theoretical Background and Literature Review

2.1 Overview:

Intrusion Detection System (IDS) has been used to detect the unwanted threats. IDSs are varied from one to another according to the design principals of the system. Steady State Genetic Algorithm has many elements; each element can affect the performance of the algorithm. This chapter will present the definition of IDS and some knowledge about different types of IDSs. Then it will explain with examples the elements of Steady State Genetic Algorithm. Finally it will show some related work to the domain of this thesis.

2.2 Theoretical Background

This part will consider Intrusion Detection System and Steady State Genetic Algorithm.

2.2.1 Intrusion Detection System

An Intrusion Detection System was defined by Information Assurance Technology Analysis Center (IATAC), (2009) as: "Intrusion detection is the act of detecting unwanted traffic on a network or a device. An IDS can be a piece of installed software or a physical appliance that monitors network traffic in order to detect unwanted activity"

It is also defined by Scarfone and Mell, (2007) as "the process of monitoring the events occurring in a computer system or network, and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. IDS is a software that automates the intrusion detection process".

According to the definitions, the IDS can observe the event and store the information related to these events, or send this information to another system, such as Security Information and Event Management System.

IDS also can help security administrators by giving an alert to notify administrators about the events, or giving them a summarized report about those events. (Scarfone & Mell ,2007)

Kozushko, (2003) interpreted that intrusion detection complement network firewalls which acts as a barrier between internal network and the outside world because of the following reasons:

1- Not all the internet accesses will be done through firewall.

2- Some of threats are originated inside the firewall

3- Firewalls are subject to attacks.

2.2.2 Intrusion Detection System Taxonomy Elements

1- Knowledge based System vs. Behavior based System:

Al-Sharafat, (2009) explained that Knowledge based Intrusion Detection technique accumulates knowledge explicitly from specific attack and possible vulnerabilities to exploit at different attacking attempts. The knowledge accumulated in advance, this means that the system will have very low false alarm rates, which is one of the most strength points for this approach.

Another strength point for knowledge based approach is that the system will analyze the problem in order to understand it and take an appropriate action.

Nevertheless, this method has a set of drawbacks (Al-Sharafat, 2009). Firstly, to have a good and an effective knowledge based IDS, the information must be up to date.

This information must be gathered after a detailed analytical process over each attack, so there is difficulty in gathering required information. Secondly, this method may have to face a generalization issue. Misuse IDS, sometimes called signature, is considered as Knowledge Based IDS.

Al-Sharafat, (2009) also explained that behavior based intrusion detection system must have the normal and the intruder behaviors. To detect the intrusion, the system should notice the deviation between those two behaviors by comparing the model of normal behavior with the current activity. The alarm will be generated if there is a difference, and the behavior will be considered as intrusion.

The drawback of this method (as referred in the same reference) is about its accuracy. This approach may have a high false-alarm rate because the learning phase may not cover all of the entire behavior scope.

Anomaly IDS is considered as behavior base IDS.

2- Host based System vs. Network based System

In order to recognize and catch the attack, IDS can use Network based IDS or Host based IDS. Both of them look for specific pattern which called signature to indicate malicious activities or policy violations. If the recognition and indication are done using IDS over network traffic, such as traffic volume, protocol usage .. etc, then it was called Network based IDS. Whereas if they done to look for attack signatures in log files, such as process identifier, system calls.. etc, then it was called Host based IDS. (Das, etal., 2010)

Kozushko, (2003) mentioned that these two technologies are similar in the root but they are different in their operational use. Network Intrusion Detection is used to analyze network packets and examine events as information packets, so it focuses on the abuse of vulnerabilities, but the detection engine in this technology cannot detect encrypted traffic.

In other side, Host based Intrusion Detection relates to processing data that originated in computer. This technology focuses on the abuse of privilege to examine events related to files and applications.

Selvakani and Rajesh, (2007) showed the following figure which explained the difference between Host and Network based IDS and the location of IDS and firewall:

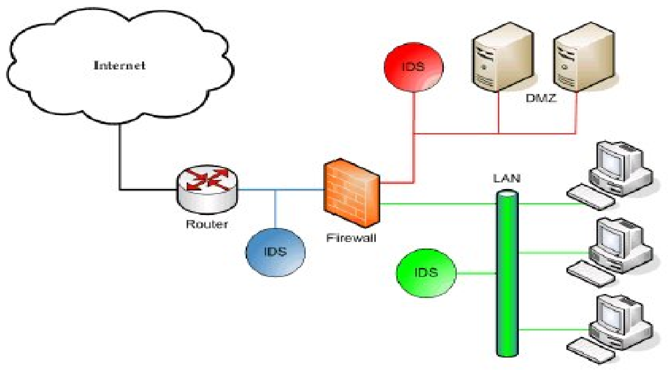


Figure (2.1) Network IDS vs. Host IDS

Primary classes of Detection methodologies:

1- Misuse Detection

2- Anomaly Detection

1- Misuse detection:

In this methodology there is a signature or pattern which will be compared with the observed events to detect the unknown threat, (Scarfone & Mell, 2007). For example, if there is an email with subject "free picture" and an attachment its name is "freepics.exe", this can be effectively detected, because its characteristics known as malicious activity.

But misuse cannot be effective if the attachment file has been renamed as "freepics2.exe", because the signature "freepics.exe" does not match the name "freepics2.exe".

Misuse detection will compare the current noticed activity with a list of signatures.

2- Anomaly Detection:

In this methodology the IDS has many profiles which describe the normal behaviors; for example, the user profile will describe the normal behavior of the user, when the user events occurred. The IDS will compare the user events with the definition of what activity is normal or with the normal activity which stored in the user profile, so that, the IDS can check if the event is normal or abnormal. (Scarfone & Mell ,2007).

Ibrahim, (2010) supported the figure below to build a good understanding about these two methodologies.

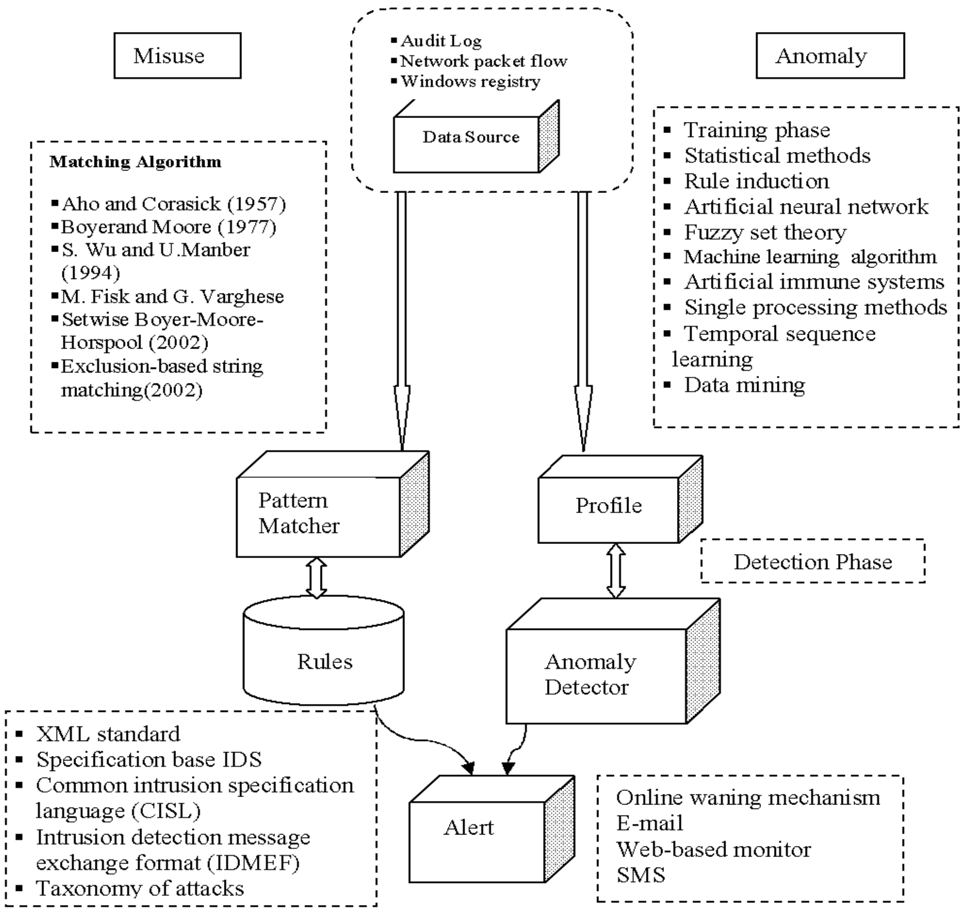


Figure (2.2) Misuse IDS vs. Anomaly IDS

2.2.3 Intrusion Detection Evaluation:

Kang, Fuller and Honavar (2005) defined False Positive Rate as "a fraction of normal data mis-identified as intrusion", and defined Detection Rate as "a fraction of the intrusion identified". Al-Sharafat, (2009) interpreted those two definitions as the following:

DR equation:

$$DR = \frac{\sum_{service=1}^{n} No.OfDetectedAttacks}{\sum_{service=1}^{n} No.OfTotalAttacks} \tag{2.1}$$

FPR equation:

$$FPR = \frac{\sum_{service=1}^{n} No.OfFalseAlarms}{\sum_{service=1}^{n} No.OfTotalNormalAlarms} \tag{2.2}$$

Agravat, (2011) evaluated the performance of the system by finding the value of Precision, Recall and overall accuracy as the following:

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

$$OverallAccuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

TP = True Positive

TN = True Negative

FN = False Negative

FP = False Positive

2.2.4 Network Attacks:

Attack types fall into the following categories:

a- Denial of Service.

b- User to Root

c- Remote to Local

d- Probing

Sung and Mukkamala, (2003) defined these Categories as:

a- Denial of Service (DoS):" is a class of attacks in which an attacker makes some computing or memory resources too busy or too fall to handle legitimate requests, or denies legitimate user access to a machine".

b- User to Root (U2R):" is a class of attacks in which all attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system".

c- Remote to Local (R2L):" is a class of attacks in which an attacker sends packets to a machine over a network, but who does not have an account of that machine, exploits some vulnerability to gain local access as a user of that machine".

d- Probing: "is a class of attacks in which an attacker scans a network of computers to gather information or find known vulnerabilities".

Sung and Mukkamala, (2003) displayed a figure about the data distribution of each attack type in the 10% of KDD Cup 99. The following table displays each category with its types and the distribution of that type which included in 10% of KDD Cup 99.

Category	Type	Distribution	Category	Type	Distribution
R2L	ftp_write	0.00014	**Normal**	normal	19.85903
	guess_passwd	0.00100	**DoS**	back	0.04199
	imap	0.00022		land	0.00041
	multihop	0.00013		neptune	21.88491
	phf	0.00007		pod	0.00503
	spy	0.00003		smurf	57.32215
	warezclient	0.01944		teardrop	0.01866
	warezmaster	0.00038	**Probe**	ipsweep	0.23796
U2R	buffer_overflow	0.00056		nmap	0.04415
	loadmodule	0.00016		portsweep	0.19853
	perl	0.00005		satan	0.30298
	rootkit	0.00018			

Table (2.1): The distribution of attack types within 10% of KDD Cup 99

Network Intrusion Detection System must detect these types to be effective system and to get a trustworthy.

2.3 Genetic Algorithm (GA)

As described by Kumar, Husian, Upreti and Gupta, (2010), GA which introduced by John Holland in the early 1970's, is a probabilistic search algorithm based on the natural genetic action mechanism, and it is an adaptive heuristic search method based on population genetics, this method is used in computing to find optimal solutions for complex problems and to find exact or approximate solutions for optimization and searching problems. GA's are also described as a particular class of evolutionary algorithms that use techniques inspired by evolution such as selection, crossover and mutation.

A Chromosome has many genes. Each gene has a locus and alleles; it may be phenotype or genotype.

The explanation of the above statement is: the chromosome is a solution; the chromosome has a lot of genes, which are parts of solution. Each gene has Locus (position of gene) and Alleles (values of gene). Phenotype is a decoded solution whereas Genotype is an encoded solution.

Why Genetic Algorithm was used in search and optimization?

Deb, (1998) discussed the idea of using GA in search and optimization. He explained that there are two groups of traditional search and optimization methods. Those are direct which uses objective function and constraints to guide the search strategy. But the problem with this method is: it is slow and it needs many functions evaluation for convergence.

The second group of traditional search and optimization is gradient-based method which uses the first and second-order derivations of the objective function, and/or constraints to guide this search process.

It quickly converges to an optimal solutions, but still not efficient in discontinuous problems. For this, GAs are used to pass a mentioned problem, moreover using GAs supports convergence to an optimal solution. It is efficient to solve different problems, and it is efficient to handle problems having discrete variables.

The effectiveness of using GA in problem solving depends on many factors. Chinneck, (2006), the first one is the quality of the initial population, which is generated randomly with low quality but it is better if GA is provided with high quality population.

The second factor depends on the selection of Fitness Function. The third factor is individual's representation, and the last is the GA parameters.

Adewumi, (2010) explained that among the evolutionary algorithms, GA's are the most successful because of its unique characteristics, these characteristics include:

1- Parallelism.
2- Derivative-free nature.
3- Ability to explore large solution space.
4- Ability to handle complex fitness landscape and deal with multi-objective problems.
5- Ability to handle noisy function and escape from local optima and the best of all.
6- Ability to handle large but poorly understood search space.

These are the reasons which make the success of GA as evolutionary algorithms, and make the success of GA in problem solving.

Chinneck, (2006) put the whole process of GA together as the following steps:

Step 0: Specify the algorithm by choosing the population size *n* and mutation rate; choosing operators and determining the stopping condition.

Step 1: Generate an initial population randomly, and calculate the Fitness Function for each string.

Step 2: Apply the reproduction operator on the current population to generate a mating pool of size *n*.

Step 3: Apply the crossover operator on the strings in the mating pool to generate a tentative new population of size *n*.

Step 4: Apply mutation operator on the new population to create the final new population. Calculate the fitness values of the solution strings in the new population.

Step 5: Exit with the best solution if the stopping condition is met, otherwise go to step 2.

To clarify the process, Selvakani and Rajesh, (2007) showed the following Genetic Algorithm structure:

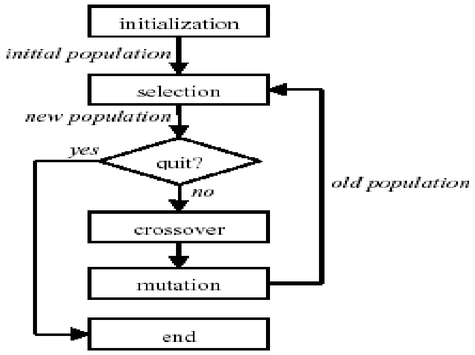


Figure (2.3): Genetic Algorithm Structure

The figure above shows us the steps of Simple Genetic Algorithm, but this thesis will focus on Steady State Genetic Algorithm. Richter, (2010) explained the process steps of SSGA in his dissertation as the following:

1 Select the s(n) individuals of the initial population independently and randomly.

2 With probability $p_c(n)$ go to step 3' and with the remaining probability of $1-p_c(n)$ go to step 3''. Steps 3' and 3'' are mutually exclusive.

3 3') Choose two parents x and y from the current population. Let z* be the result of uniform crossover applied on x and y and let z be the result of mutation applied on z*.

3'') Choose one parent x from the current population. Let z be the result of mutation applied on x.

4 If the fitness of z is smaller than the fitness of the worst individual of the current population, go to step 2. Otherwise, add z to the population. Let W be the multi-set of individuals in the enlarged population which all have the worst fitness and let W' be the set of those individuals in W which have the largest number of copies in W. Eliminate randomly one element in W from the current population. Go to step 2.

2.4 Steady State Genetic Algorithm (SSGA) elements:

- **Population**

Cleary, (2011) described the population as a result of single iteration of Genetic Algorithm. Iteration can create a new population. Population contains a set of chromosomes; each chromosome is one complete possible solution to the problem to be solved using Genetic Algorithm. It will concatenate each problem parameters as binary encoding genes into a single binary string.

Kumar, Husian, Upreti and Gupta, (2010) explained that to generate an initial population, many individual solutions are generated randomly; the nature of the problem will determine the population size to cover the range of possible solutions.

Deb, (1998) mentioned that there is a specific performance is measured versus the population size:

- In very small population size, you need more generations to find optimal solutions.
- In moderate population size, GA's operators start to get adequate population members; GA here begins to improve with an increase in the population size.

- In Adequate population size, it is adequate to allow necessary scheme processing and GA performs successfully.
- In very large population size, GA is only allowed to run for a small number of generations, the performance begins to reduce.

- **Evaluation**

For each chromosome there is a Fitness Function used to evaluate the fitness of each chromosome. Fitness's value reflects the quality of each chromosome.

- **Encoding**

The gene is a problem parameter; it can be encoded as a binary, integer, or real number. Cleary, (2011) preferred to use the gene as binary string; he described encoding using integer parameter as trivial encoding. If the parameter defined as enumerated type, then the encoding must deal with the situations of the parameter value within its enumeration. If the parameter defined as float or real number, the developed linear equation will map the range of floating point values with a range of integer range. So a parameter is an integer binary string in the genetic representation.

- **Selection**

It is the process of selecting the chromosomes to apply Genetic Algorithm.

Types of selection are:

- **Roulette Wheel Selection (RWS)**: The chance of a chromosome being selected is proportional to its fitness value. This can be work as in the following steps:

  Step 1: Find the fitness value (fv) for each chromosome in the population using Fitness Function.

  Step 2: Calculate sum fitness (Sf) for all chromosomes in the population:

$$\mathbf{Sf} = \sum_{i=1}^{n} \mathbf{fv}i \qquad (2.6)$$

Step 3: Calculate average fitness (Af) in the population:

$$\mathbf{Af} = \frac{\mathbf{Sf}}{\mathbf{n}} \qquad (2.7)$$

Step 4: Find expected fitness (Ef) for each chromosome in the population:

$$\mathbf{Ef}i = \frac{\mathbf{fv}i}{\mathbf{Af}} \qquad (2.8)$$

Step 5: Calculate sum expected fitness (SumEf) for all the chromosomes in the population:

$$\mathbf{SumEf} = \sum_{i=1}^{n} \mathbf{Ef}i \qquad (2.9)$$

Step 6: Generate random number (G) in the range [0,SumEf]

$$\mathbf{G} = \mathbf{Rnd}()mod\mathbf{SumEf} \qquad (2.10)$$

Step 7: Select the chromosome that added its fitness value to the previous chromosomes fitness value's to make (SumEf > = G).

Step 8: Redo n times from step 6, where n = population size.

- **Elitism Selection**:

  The idea here is to arrange the chromosomes in the decreasing order according to theirs fitness values. Then apply the selection with each two chromosomes in the arranged set. In this way, Genetic Algorithm will be applied between strong chromosomes or between weak chromosomes. This mean there is no chance to apply Genetic Algorithm between weak and strong chromosomes.

- **Rank Selection**: The rank values can be distributed through the set of chromosomes according to theirs fitness values, after that the new fitness values can be calculated using another Fitness Function. Finally the Roulette Wheel can be used to choose the selected chromosomes. This can be work as in the following steps:

Step 1: Arrange the chromosomes in decreasing order according to their's fitness values.

Step 2: Assign a rank value to each chromosome according to its arrangement in the set.

Step 3: Calculate the new fitness value for each chromosome using the following equation (Schmidt & Stidsen, 1997):

$$F = max - (max - min) * \frac{rank - 1}{Npop \quad 1} \qquad (2.11)$$

Where 1<max<=2 & min = 2-max

- **Stochastic Universal Sampling (SUS)**: instead of spinning the Roulette Wheel n times as described in Roulette Wheel Selection, in this technique one can spin the Roulette Wheel just once but after determining n points in the wheel. Then choose chromosomes that situated in front of determined points.
- **Binary Tournament Selection**: for n times do the following:
  - Choose two chromosomes randomly.
  - Select the chromosome with the highest fitness value.

- **GA operator: Crossover**

This process is used to interchange genes between chromosomes to create offsprings. Types of crossover are:

- Single Point

Step 1: randomly selects the Crossover point within a chromosome.

Step 2: interchange the two parent chromosomes at this point to produce two new offspring's.

Ex: Parent 1: 10010|000

Parent 2: 00101|101

After Crossover:

Offspring 1: 10010|101

Offspring 2: 00101|000

- Two Points

Step 1: randomly selects two points.

Step 2: interchange the two parent chromosomes between these points.

Ex: Parent 1: 100|100|00

Parent 2: 001|011|01

After Crossover:

Offspring 1: 100|011|00

Offspring 2: 001|100|01

- Uniform

According to some probability, Crossover will decide which parent will contribute each of the gene values in the offspring chromosome.

Ex: Parent 1: 10010000

Parent 2: 00101101

If mixing ratio is equal to 0.5 this means 50% of genes in the offspring will come from parent 1 and the other will come from parent 2.

Offspring 1: $1_1 0_2 1_2 1_1 1_2 0_1 0_1 1_2$

Offspring 2: $0_2 0_1 0_1 0_2 0_1 1_2 0_2 0_1$

- **GA operator: Mutation**

This process will change the value of one gene of the chromosome. Types of Mutation are:

- Flip bit (Used for binary represented genes)

  Step 1: Choose one gene randomly.

  Step 2: Flip the value of the chosen gene.

- Boundary (Used for integer and float represented genes)

  Step 1: Choose one gene randomly

  Step 2: Replace the value of the gene with the upper or the lower value.

  Ex: the following chromosome:

  2.5,1.0,3.7,4.2,<u>5.1</u>,7.6,4.4,2.9

  When the bit mutated to the upper boundary:

  2.5,1.0,3.7,4.2,<u>6.0</u>,7.6,4.4,2.9

- Uniform (Used for integer and float representation)

  Step 1: Choose a gene.

  Step 2: Replace the value of a chosen gene with a uniform random value selected between the user specified upper and lower bounds for that gene.

  Ex: Chromosome:

  2.5,1.0,3.7,4.2,<u>5.1</u>,7.6,4.4,2.9

  The value must be in the range (1.0,8.0)

  5.1 $\rightarrow$ randomly changed to 6.7

  The new chromosome will be:

  2.5,1.0,3.7,4.2,<u>6.7</u>,7.6,4.4,2.9

- **Replacement**

This process will compare between several chromosomes to choose the best. Types of Replacement are:

- Binary Tournament:

  It will take two chromosomes and according to their fitness values it will choose the best of them.

- Triple Tournament:

  It will replace the worst two chromosomes between three chromosomes by the chromosome with the highest fitness value.

- **Stopping Criteria**

Starting with an initial population, the evolution process is repeated until the satisfaction of the end condition.

Kumar, Husian, Upreti and Gupta (2010) mentioned common terminating conditions such as:

- The found solution satisfies minimum criteria.
- A fixed number of generations reached.
- Allocating budget (ex: time, money) reached.
- Successive iterations no longer produce better results.

As mentioned before, the process will continue until satisfaction of the end condition, but in some cases according to Chinneck, (2006), it is better to stop the process in case of very little change between generations; this means that the worst solution string fitness in the population has not changed for several generations.

In this case, look to the both populations together (the last and current-generated populations).Select the best solution between these populations and consider it as a final new population.

Using machine learning, the hidden patterns can be explored among growing data, so that the computers can program themselves. Machine learning can be used to support the rules in the rules pool.

- **GA application**

There are many Applications used to apply GA. Kumar, Husian, Upreti and Gupta, (2010) described some applications of GA in problem solving. The following are two of them:

1- Sensor based robot path planning:

Using GA to generate a short and safe path to goal with obstacle avoidance, the paths are a sequence of control vectors of orientation. The path represented as a set of orientation vectors with equal distance. The final path is the composition of polygonal lines.

2- Gaming

There is a large structure of possible traits for the game (aggressiveness, probability of running away.. etc). Those possible traits can be created and then Genetic Algorithm can be used to find the best combination of these structures to beat the player.

2.5 Literature Review and Related Work

Diaz-Gomez and Hougen, (2007) presented an iterative process for doing Misuse Detection and compared the results with Genetic Misuse Detection. They found that if the number of individuals in the GA's population is changed (keeping other parameter the same), then the GA's False Negative percentage becomes better.

Selvakani and Rajesh, (2007) performed GA-based approach for Network Misuse Detection. They utilized a technique for creating rules for R2L and DoS attacks. They used KDD Cup 99 data set to find the probability of Detection and they found the overall performance = 59% Detection Rate and False Alarm Rate = 0.1 %. They used GA-based approach for Network Misuse Detection in order to frame the rules needed.

Ghali, (2009) proposed a new hybrid algorithm called Rough Set Neural Network Algorithm (RSNNA). To reduce the number of computer resources required to detect an attack. The algorithm used Rough Set theory to select features, and used NN to identify the kind of attack. They used KDD-99 in testing. The result showed that the model was able to select features resulting in 83% data reduction and 85%-90% time reduction and 90% reduction in error in detecting a new attack.

She tested her model on Anomaly NID, and did not test it on Misuse as this research did.

Stewart, (2009) modified GA and NN in order to propose a new GA and NN combination to be capable of tuning not only the weighting of a NN, but also its size and connectivity. The improved GA was reduced user interface, and improved acceptance probability. He used modified Genetic Algorithm and got 75.25 DR and 3.412 FPR, he also improved GA and got 79.672 DR and 2.69 FPR.

Al-Sharafat and Naoum, (2009) used SSGA to detect intrusions in Anomaly based NIDS. They modified a new Fitness Function and used it in a Genetic Algorithm, and they used a Genetic Algorithm as engine and got the following results: (DoS:

97.6%) ,(Probe: 37.7%) ,(U2R: 28.8%) and (R2L: 83.9%). But they didn't use genetic algorithm in misuse detection.

Khan, (2011) showed that GA can be used for formulating decision rules in Intrusion Detection System. His research was done just over DoS and Probe, but this thesis done over four types of attack. In addition, he used 10000 records for training and 2000 iteration, but this thesis used 50000 records for training and less than 50 iterations.

Agravat and Rao, (2011) described two objectives of fuzzy Genetic-based learning algorithms and discussed its usage to detect intrusion in a computer network.

The objectives are to minimize the number of fuzzy rules and to maximize the classification rate, they used the fuzzy Genetic Algorithm for Misuse Detection to be evaluated and tested over KDD Cup 99 dataset. They used 10% labeled data for training and testing of the GA, they used 20 attributes of 41 from KDD Cup 99, and they used the following parameters:

The number of elite solutions = 20 %

The Crossover probability = 0.9

The Mutation probability = 0.1

The number of generation = 50

The results after testing are with Precision = 0.9979, Recall = 1, Accuracy = 0.985

Uppalaiah, Anand, Narsimha, Swaraj & Bharat, (2012) presented Genetic Algorithm to identify various attack type of connection. They trained the Genetic Algorithm on the KDD Cup 99 dataset to be used with Intrusion Detection System. The average Detection Rate was 83.65, but this thesis got better results.

Naoum, Abed & Al-sultani, (2012) classified intrusions using an Enhanced Resilience Back Propagation Neural Network. They got 94.7% average Detection Rate, and 15.7 % False Positive Rate. But this thesis used Genetic Algorithm instead of Resilience Back Propagation Neural Network, and got better results.

Hoque, Abdul Mukit & Bikas, (2012) presented an Intrusion Detection System by applying Genetic Algorithm to detect various types of network intrusions. They got the following Detection Rate results (Probe: 71.1%), (DoS: 99.4%), (U2R: 18.9%) and (R2L: 5.4%), but in this thesis we got new results.

Chapter Three

Methods and Procedures

3.1 Overview

This chapter will explain how to apply the system and how to get the results. It will display the proposed model and explain the model details.

3.2 Methodology:

This thesis examines the enhanced Steady State Genetic Algorithm (SSGA) for Network Intrusion Detection System (NIDS), and evaluates the system strength and trustworthy by calculating the system Detection Rate (DR) and False Positive Rate (FPR).

The relevant data has been collected from KDD Cup 99. The system uses two datasets; training dataset which is 5% of KDD Cup 99 and testing dataset which is 1% of KDD Cup 99.

Training data has been received and distinguished as condition part that holds the features values, and action part that holds the label of the attack.

A comparison has been done to determine the best features, then the rules have been filtered and stored in rules database.

Reward Penality based Fitness Function has been developed and used as an element of (SSGA) to evaluate the rule.

Using (SSGA) a comparison has been done between several Selection methods and several Crossover types, to be combined together and used in SSGA to perform better.

The resulted rules after using (SSGA) will be stored in the rules pool, to be used at the next step to examine the testing dataset.

Finally, to evaluate the system; two values must be calculated, Detection Rate (DR) and False Positive Rate (FPR). The strong system must have high Detection Rate (DR) and low False Positive Rate (FPR).

Proposed Solution Model

The following figure displays Proposed Solution Model.

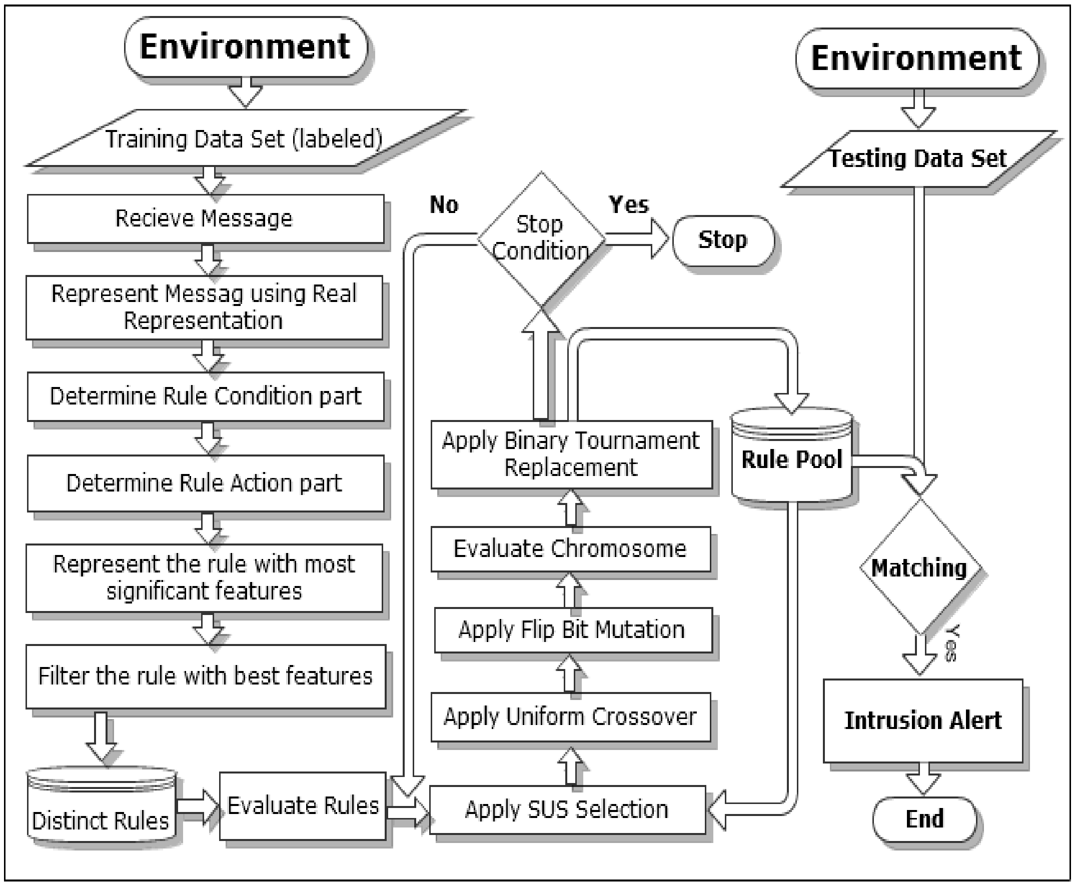

Environment
Training Data Set (labeled)
Recieve Message
Represent Messag using Real Representation
Determine Rule Condition part
Determine Rule Action part
Represent the rule with most significant features
Filter the rule with best features
Distinct Rules
Evaluate Rules
Apply SUS Selection
Apply Uniform Crossover
Apply Flip Bit Mutation
Evaluate Chromosome
Apply Binary Tournament Replacement
Stop Condition
No
Yes
Stop
Rule Pool
Environment
Testing Data Set
Matching
Yes
Intrusion Alert
End

Figure (3.1)

Enhanced Steady State Genetic Algorithm Model for Misuse Network Intrusion Detection System

3.3 The algorithm of Steady State Genetic Algorithm

Start a new Generation:

Step (1): Determine a population size.

Step (2): Represent data using real representation.

For each population in the rule pool, do:

Step (3): Select the chromosome using Stochastic Universal Sampling Selection.

Step (4): Apply Uniform Crossover.

Step (5): Apply Flip Bit Mutation.

Step (6): Evaluate the chromosome using Reward Penality Fitness Function.

Step (7): Apply Binary Tournament Replacement.

Step (8): Save the created rules in the Rules Pool.

Step (9): Go to the next population.

Step (10): Check the stop criteria, if not satisfied then go to start a new Generation.

3.4 Proposed Model Structure

The structure of the proposed model is described as the following:

3.4.1 Environment:

It is the KDD Cup 99 dataset, it has 4,940,210 records, and each record has 41 features in addition to the feature number 42 which determines the type of the attack. Each feature determined a value; some features have a binary value, and other features have a real value. But it is possible for such features to hold values represented as string.

The combination of feature values for 41 features will determine an instance of attack as it is in the feature 42.

From the environment, the researcher can get a subset as a training dataset and another subset as a testing dataset.

3.4.1.1 Training Data set:

Using the training data set, the system began to receive data that helps in rules creation in order to support decision making to decide if the received record is normal or attack. The data of training data set came from KDD Cup 99. This thesis used 250000 records as training data set.

3.4.1.2 Testing Data set:

Using the testing data set, the system began to receive data that helps in evaluating the system task; the data set helped the system in examining the matching between unlabeled records and the rules stored in Rule Pool, the environment of testing data set came from KDD Cup 99. This thesis used 50000 records as testing data set.

3.4.2 Detector:

The main task of detector is to classify the data by receiving the message and determining the condition and action parts of the rule. The detector should also filter the message from redundancy and deal with the most significant features. The parts of the detector will be discussed below.

The first part of the detector is the message receiver which has received the message that came from the training dataset.

The second part of the detector is to represent the message using real representation. Someone may ask that the representation is a part of Genetic Algorithm, but why does the researcher include the representation in the detector?

Logically, the answer is that the representation element is correct to be anywhere before the selection element, but after receiving message. In addition to that, it is important to represent the message here to create the rule as a chromosome in order to use the most significant features as will be explained below.

The values of the most significant features selected in this research are varied between binary numbers and real numbers, so the real representation is preferred, because it includes both types; binary and real type.

The third part of the detector has to determine the condition part of the rule. The condition part of the rule has a collection of values related to collection of features. Those values of that features made the condition to cause such attack.

The fourth part has to make a relation between the condition values and type of attack to keep in consideration that if those features have those values then the type of attack will be as it is in the feature number 42. So the Misuse Network Intrusion Detection System should save such facts in the system during training period to use it during testing period, in order to detect the intrusions.

The fifth part of the detector is to represent a rule with the most significant features. Comparing 41 features of testing data set records with 41 features of training data set records is a time consuming process. So there are many researches to answer the question: What are the most significant features that sufficient to recognize the attack?

This research collected some of those researches and represented the condition action rule according to the researches results. After representing the rules it is easy to determine the best features to prepare the rules to enter the sixth part of detector, which is responsible for filtering the rule with the best features.

3.4.3 Distinct Rules Database:

This database has the same data stored in training data set, but the data had the following operations done over it:

1- Classifying to the 5 classes (Normal, DoS, Probe, R2L and U2R).

2- Creating the rules as Condition-Action form.

3- Filtering the rules with the best significant features.

4- Removing the redundant rules from the rules database

3.4.4 Rules Evaluation:

As an element of Genetic Algorithm, the chromosome evaluated previously in order to be prepared to the selection process. The evaluation process had been done using Reward-Penality based Fitness Function.

3.4.5 Steady State Genetic Algorithm Unit:

This part is responsible for creating new rules.

Evaluation:

Using Reward Penality Fitness Function as proposed by (Alabsi and Naoum, 2012), each chromosome will be evaluated, in order to be selected in the selection stage.

Selection:

The research will use Stochastic Universal Sampling Selection, because it gets best results when it is used with Uniform Crossover as it will be explained in chapter 4.

Crossover:

At this stage, Uniform Crossover will be used by selecting random points within a chromosome and interchange the two parent chromosomes at these points to produce two new offsprings. As explained in chapter 4; Uniform Crossover got the best results when used after Stochastic Universal Sampling Selection.

Mutation:

If the feature value is discreate value, then Mutation will use flip bit by choosing a random gene then flip the value of a chosen gene; if the gene is equal to 0 then it will be equal to 1 else it will be equal to 0. Otherwise, if the feature value is continuous, the new value will be equal to a random number of specific ranges.

Apply Evaluation:

Evaluations of each chromosome using Reward Penality Fitness Function as described in chapter 4. But this stage helps in evaluation of generated chromosomes. And also helps in applying Replacement.

Apply Replacement:

Apply the Replacement process using Binary Tournament.

Check the Stop Criteria:

Checking the stop criteria can be done by searching about an answer for the question: Are there any additional rules to be produced? If the answer is yes, then the Genetic Algorithm applied additional generation, otherwise, the Genetic Algorithm will be stopped.

3.4.6 Rules Pool:

It contains many rules gathered from training phase and SSGA Process in order to use it in the testing phase.

3.4.7 Testing Classifier (Matching):

In this phase the proposed model will try to match the received packet with the existent rules in order to distinguish the data and discover the intrusions to be alerted.

3.5 Population: ***The Data set (KDD Cup 99):***

Mukkamala, Sung and Abraham, (2004) mentioned that: "In 1998 DARBA Intrusion Detective program acquired raw TCP/IP dump data for network by simulating a typical U.S. Air Force LAN. The LAN blasted with multiple attacks, for each TCP/IP connection, 41 various quantitative and qualitative features were extracted".

KDD'99 features can be classified into three groups, (Tavallaee, Bagheri & Ghorbani, 2009):

- Basic features: are all the attributes that can be extracted from a TCP/IP connection.
- Traffic features: include features that are computed with respect to a window interval.
- Content features: include features used to look for suspicious behavior in the data portion e.g., number of failed login attempt.

KDD Cup 99 described as the most widely used data set in the field of IDSs evaluation, so this research will use it as an environment for training and testing, the proposed model will use a sample of 5% of the whole environment.

System Evaluation:

The evaluation of proposed model will be determined by the Detection Rate (DR) and False Positive Rate (FPR), in this sense; the results can be compared with others.

Feature Classes:

Attack types fall into the following categories (Mukkamala, & Sung, 2003):

a- Denial of Service.

b- User to Root

c- Remote to Local

d- Probing

To identify each type of attack, the feature issue must be taken in consideration.

Ghali, (2009) listed the 41 network features and their labels, those labels used for easier referencing as the following:

Label	Network Data Features	Label	Network Data Features	Label	Network Data Features
A	Duration	O	Su_attempted	AC	Same_srv_rate
B	Protocol-Type	P	Num_root	AD	Diff_srv_rate
C	Service	Q	Num_file_creations	AE	Srv_diff_host_rate
D	Flag	R	Num_shells	AF	Dst_host_count
E	Sec_Byte	S	Num_access_files	AG	Dst_host_srv_count
F	Dst_Byte	T	Num_Cutbounds_cmds	AH	Dst_host_same_srv_rate
G	Land	U	Is_host_login	AI	Dsv_host_diff_srv_rate
H	Wrong_fragment	V	Is_guest_login	AJ	Dst_host_same_src_port_rate
I	Urgent	W	Cont	AK	Dst_host_srv_diff_host_rate
J	Hot	Z	Sev_count	AL	Dst_host_server_rate
K	Num_failed_login	Y	Serror_rate	AM	Dst_host_srv_serror_rate
L	Logged_in	X	Sev_serror_rate	AN	Dst_host_rerror_rate
M	Num_comprised	AA	Rerror_rate	AO	Dst_host_srv_rerror_rate
N	Root_shell	AB	Srv_rerror_rate		

Table (3.1): Network Data Feature Label

Zainal, Maarof, Shamsuddin, and Abraham, (2008), proposed an ensemble of one-class classifier, the classifier deployed three techniques which are: Linear Genetic Programming (LGP), Adaptive Neural Fuzzy Inference System (ANFIS) and Random Forest (RF).

They addressed the issue of accuracy and false alarm rate to select the relevant significant features. They addressed the issue by reducing the input features in order to disclose the hidden significant features; they used the following features for each attack:

Attack	**Features**
Normal	**f12, f31, f32, f33, f35, f36, f37, f41**
Probe	**f2, f3, f23, f34, f36, f40**
DoS	**f5, f10, f24, f29, f33, f34, f38, f40**
U2R	**f3, f4, f6, f14, f17, f22**
R2L	**f3, f4, f10, f23, f33, f36**

Table (3.2): Selected Features with One-class Classifier

The ensemble model gets results more accurate, more true positive and less false positive than LGP, ANFIS or Random Forest. In their paper they demonstrated that the ensemble of different learning paradigms can improve the detection accuracy.

Mukkamala and Sung, (2003) described the features as important, secondary and unimportant features, based on some tested rules, they classified the input nodes to 5 classes, each class contains three categories {important}, <Secondary>, (unimportant).

Class 1:

{1,3,5,6,8-10,14,15,17,20-23,25-29,33,35,36,38,39,41}

<2,4,7,11,12,16,18,19,24,30,31,34,37,40> (13,32)

Class 2:

{3,5,6,23,24,32,33} <1,4,7-9,12-19,21,22,25-28,34-41> (2,10,11,20,29,30,31,36,37)

Class 3:

{1,3,5,6,8,19,23-28,32,33,35,36,38-41} <2,7,9-11,14,17,20,22,29,30,34,37>

(4,12,13,15,16,18,19,21,3)

Class 4:

{5,6,15,16,18,32,33} <7,8,11,13,17,19-24,26,30,36-39>

(9,10,12,14,27,29,31,34,35,40,41)

Class 5:

{3,5,6,24,32,33} <2,4,7-23,26-31,34-41> (1,20,25,38)

These classified features will help the work in selecting feature step in the detecting phase.

Mukkamala, Sung and Abraham (2004), tried to eliminate the useless features to enhance the accuracy of detection while speeding up the process of computation. One can use empirical methods to test all possibilities by taking two features at a time, then three features at a time and so on until they got the significant features, but here, they tried to remove one feature each time and tried empirical methods, so they got the following results:

Attack	**Features**
Normal	**F5, F6, F10, F13, F40**
Probe	**F3, F12, F27, F31, F35**
DoS	**F7, F8, F12, F13, F23**
U2R	**F14, F17, F25, F36, F38**
R2L	**F6, F11, F12, F19, F22**

Table (3.3): Selected Features by eliminating useless features.

However, this research found that the research of Mukkamala, Sung and Abraham (2004) has given acceptable results so it is adopted to be used in the stage of representing rules with the most significant features.

Chapter Four

Experimental Results

4.1 Overview

Network Intrusion Detection System has been built. The system has been supported by Steady State Genetic Algorithm. There are many results which have appeared through the system execution. In this chapter these experimental results has been shown.

4.2 KDD Cup 99

The whole data of KDD Cup 99 is 4940210 records. On the KDD official website the whole data is available, but this research used 5% of the whole data as training dataset. The researcher uses 250000 records as training dataset, and other 50000 records as testing dataset. The following table shows the distribution of the attacks through the training dataset:

Attack	No. Of Rows	Percentage
Normal	71225	28.49 %
DoS	174302	69,72 %
R2L	1125	0.45 %
U2R	29	0.0116 %
Probe	3319	1.3276 %
Total	250000	100 %

Table (4.1): Distribution of Attacks within Training Dataset

The training dataset used to support the system about the knowledge related to the attacks, whereas the testing dataset used to be tested and to evaluate the system itself by evaluating the Detection Rate and False Positive Rate.

4.3 Selecting the most Significant Features

The KDD Cup 99 dataset contains a huge number of records, each of which has 41 features plus one attribute has a name of the attack to be used as a labeled record. To judge that the record of testing dataset belongs to specific category of attack, the tested record must be the same in features values as at least one of the training dataset record.

The process of comparing a record of testing a dataset with the whole data in the training dataset using 41 features will have resources and time consuming. To solve this problem, there are many researches related to selecting the most significant features, the result of 4 researches in this domain was used and compared to find the best research that may serve the research in this thesis.

Mukkamala, Sung and Abraham, (2004) selected the most significant features by eliminating useless features. They determined the selected features as explained in the Table (3.3). Using their results, the data has been filtered and got the number of records as shown in Table (4.2).

Attack	No. of records before filtering	No. of records after filtering
Normal	71225	46952
DoS	174302	542
Probe	3319	448
R2L	1125	68
U2R	29	16

Table (4.2) Distribution of Attacks before and after filtering [Mukkamala, Sung and Abraham (2004)]

Zainal, Maarof, Shamsuddin, and Abraham, (2008) selected the most significant features by using one-class classifier; the classifier deployed three techniques which are: Linear Genetic Programming (LGP), Adaptive Neural Fuzzy Inference System (ANFIS) and Random Forest (RF). They determined the selected features as determined in the Table (3.2). The data has been filtered as their results to get the number of records as shown in the Table (4.3).

Attack	No. of records before filtering	No. of records after filtering
Normal	71225	30748
DoS	174302	2521
Probe	3319	769
R2L	1125	479
U2R	29	28

Table (4.3) Distribution of Attacks before and after filtering [Zainal, Maarof, Shamsuddin, and Abraham, (2008)]

Mukkamala and Sung, (2003) selected the most significant features after describing the features as important, secondary and unimportant. They determined the selected features as explained in the following table:

Attack	Important Features
DoS	1,5,6,23,24,25,26,32,36,38,39
Probe	1,2,3,4,5,6,23,24,29,32,33
U2R	1,2,3,5,6,12,23,24,32,33
R2L	1,3,5,6,32,33
Normal	1,2,3,4,5,6,10,12,17,23,24,27,28,29,31,32,33,34,36,39

Table (4.4): Important, secondary and unimportant features [Mukkamala and sung (2003)].

The data has been filtered as their results to get the number of records as shown in the Table (4.5).

Attack	No. of records before filtering	No. of records after filtering
Normal	71225	67405
DoS	174302	5892
Probe	3319	1130
R2L	1125	864
U2R	29	29

Table (4.5) Distribution of Attacks before and after filtering [Mukkamala and sung (2003)]

Chou, Yen, and Luo, (2008) selected the most significant features by using an algorithm to remove irrelevant features and redundant features. They determined the selected features as shown in the following table

Attack	Important Features
DoS	1,2,3,4,5,6,12,23,24,31,32,37
Probe	1,2,3,4,12,16,25,27,28,29,30,40
U2R	1,2,3,10,16
R2L	1,2,3,4,5,10,22

Table (4.6): Features after removing both irrelevant and redundant features [Chou, Yen, and Luo(2008)]

The data has been filtered as their results to get the number of records as shown below (4.7).

Attack	No. of records before filtering	No. of records after filtering
Normal	71225	------------
DoS	174302	7642
Probe	3319	1092
R2L	1125	319
U2R	29	22

Table (4.7) Distribution of Attacks before and after filtering [Chou, Yen, and Luo(2008)]

To determine exactly the most significant features that may be helpful in this research, two sub datasets were chosen from the testing dataset. Each one contains 5000 records. Those sub datasets were tested and got the following results.

Attack	No. of records	Class 1	Class 2	Class 3	Class 4
DoS	1800	100%	0.1%	0.1%	0.3%
Probe	11	91%	91%	0%	91%
U2R	6	83%	33%	0%	66%
R2L	1	100%	0%	0%	0%
Normal	3183	18%	40%	13%	------

Table (4.8) Detection Rate for each attack according to four classes – First evaluation

Attack	No. of records	Class 1	Class 2	Class 3	Class 4
DoS	3400	100%	0.06%	0.2%	0.6%
Probe	76	97%	97%	96%	97%
U2R	0	------	------	------	------
R2L	0	------	------	------	------
Normal	1525	13.5%	36%	1%	------

Table (4.9) Detection Rate for each attack according to four classes – Second evaluation

Due to the mentioned tables it's clear that the results of class 1 are the best results and may be helpful to obtain good results in the final stage of this research. Hence, the most significant features according to class 1 will be selected.

4.4 Fitness Function

Intrusion Detection System was used to protect the system against malicious activities. Steady State Genetic Algorithms were applied to support Intrusion Detection

Systems. Steady State Genetic Algorithm can't be done without the selection process which depends mainly on fitness value that obtained using Fitness Function.

But, chromosomes vary in their strength and weakness. Hence, Fitness Function must take two points in its consideration:

- First: the reward must be as more as the chromosome's strength.
- Second: the Penality must be as more as the chromosome's weakness.

Hence, this research suggested Reward-Penality based Fitness Function.

The data of 5% of KDD Cup 99 was classified into 5 main categories; Normal, DoS, Probe, U2R and R2L. Each category record was compared to the whole data. After classification stage, there are 5 tables, each table for just one category, each table has got a name as the category type and it has included 8 columns as the following: id Column, 5 columns to have 5 features, and other two columns to hold A and AB values.

To understand the reason of creating column A, and column AB, suppose there are 5 features for DoS category, each feature's value should be in a specific range or equal to a specific value in order to evaluate the record as DoS, but in such cases, the five features got the same values as a record in DoS but still not DoS because of a specific value of one or more of the hidden features.

Suppose that the features' values are a condition part and the category's name is an action, then for each record compared with the whole 5% of CDDCup99, if the condition and action of the selected record equal to the condition and action of the Compared record, then this will increase the value of column AB of the selected record by 1. Else, if the condition of the selected record is equal to the condition of the compared record but the actions of both records don't meet each other, then the value of column A of the selected record will increase by one.

The new fitness function will depend mainly on the values of A and AB, the formula of the function is as the following:

$$Fitness = 2 + \frac{AB - A}{AB + A} + \frac{AB}{X} - \frac{A}{Y} \qquad (4.1)$$

Where:

X = the maximum value of AB in the population.

Y = the maximum value of A in the population.

Now, let us discuss the content of the function:

(AB/(AB+A)) gives the rate of the AB value in proportion to the sum of AB and A values, the resulted value will reflect the strength of the record.

(A/(AB+A)) gives the rate of the A value in proportion to the sum of AB and A values, the resulted value will reflect the weakness of the record.

To obtain the importance and strength of the record, one can subtract the weakness value from the strength value by calculating ((AB-A)/(AB+A)) as in the function above.

Now, suppose that there are two records with the following values:

Record	A	AB	Fitness = ((AB-A)/(AB+A))
Rec1	0	1	1
Rec2	0	5	1

Table (4.10) : Similar Fitness Values

But, in such cases the resulted value will not be accurate because it will deal with record1 and record2 as the same strength, whereas it is clear that record2 is stronger than record1 because of the value of AB, so the function should be supported with other positive and negative values to apply policy of Reward and Penality to the records as following:

AB/X: gives the rate which reflects the strength of the record depending on the strongest record in the population, the resulted value will be equal to Zero in the worst case (If AB value = 0) and will be equal to One in the best case if the AB value of that record is the highest AB value in the population, so it logically should be added to the function to reward the record.

A/Y: gives the rate which reflects the weakness of the record depending on the weakest record in the population, the resulted value will be equal to Zero in the best case (If A value =0) and will be equal to One in the worst case if the A value of that record is the highest value in the population, so the value of A/Y must be subtracted from the function to give the Penality of the record.

Now, assume that the record with the best case, so AB value of that record is the highest AB value in the AB column, and A value is equal to Zero, this means that Fitness = 2, on the other hand, assume that the record with the worst case, so A value of that record is the highest A value in the A column, and AB value is equal to Zero, this means that fitness = -2 , but the fitness value provided by the Fitness Function must assign a non-negative cost to each candidate (Bottaci, 2001), so the constant value of 2 will be added to the function to make the fitness value equal to 0 in the worst case, and fitness value equal to 4 in the best case, in this manner, the fitness value will be positive and in the interval [0,4] at any case.

The system has been built to calculate A value, AB value and the Fitness value for each record in the attacks tables.

4.4.1 Fitness Results

The following table is from the real data set, the table contents of column A and column AB are filled according to the comparison process described above with a simple population of 4 records for each category, whereas the contents of the column Fitness is calculated using suggested Fitness Function.

	A	AB	Fitness
Normal	0	1	3.143
	0	3	3.429
	50	7	1.246
	44	4	0.858
DoS	A	AB	Fitness
	3	419	3.985
	5	280	3.632
	5687	18	0.049
	23	5	1.365
Probe	A	AB	Fitness
	130691	2	0.002
	242	12	1.107
	0	856	4.000
	0	6	3.007
R2L	A	AB	Fitness
	180930	11	0.016
	2114	714	2.493
	0	4	3.006
	0	16	3.022
U2R	A	AB	Fitness
	134917	6	1.000
	1	4	3.267
	818	3	1.501
	10	1	1.348

Table (4.11): Real Data

4.4.2 Discussion of Fitness Results

Now, Observe that the normal record with AB = 3 is fitter than the normal record with A=1 in the case of A=0 in both records.

Observe that DoS record with AB = 5 is fitter than DoS record with AB = 18 because the first record has less A value than the second.

Observe the best case Probe record with fitness value = 4 that means constant number (2) + 1 (because A value = 0) + 1 (Because AB is the greatest AB value in the population).

Observe the R2L record with AB = 4 is fitter than the R2L record with AB = 714 because of the high value of A for the record with AB = 714.

Observe that U2R record with A = 818 and AB = 3 is fitter than U2R record with A=10 and AB = 1, because the maximum value of A is very high, and the maximum value of AB is very low, in these cases the Reward and Penality issue affect the fitness value obviously.

4.4.3 Comparing Strategy

In order to prove the validity of the new Fitness Function, another fitness function should be tested to get the results and to compare them with the new Fitness Function results.

If the fitness value of the rule X is greater than the fitness value of the rule Y according to the first Fitness Function, then the fitness value of the rule X also is greater than the fitness value of the rule Y according to the second Fitness Function. For any record in the population there are two results R1 and R2 as the following two equations:

$$R1 = \frac{FitnessValue1}{MaxFitnessValue1inPopulation} \tag{4.2}$$

$$R2 = \frac{FitnessValue2}{MaxFitnessValue2inPopulation} \tag{4.3}$$

Where: Fitness Value 1 is the result of the Reward-Penality based Fitness Function, and Fitness Value 2 is the result of the second Fitness Function of the same record. To say that the new Fitness Function is getting a good result, the values of R1 and R2 must be close to each other.

Some of the researches (Selvakani and Rajesh (2007), Berlanga, Del Jesus, Gatco and Herrera (2006)) used Support Confidence Framework as a Fitness Function, they used the following equation:

$$FitnessFunction = t1 * Support + t2 * Confidence \qquad (4.4)$$

Where:

Support: indicates the recurrence of AB within all the rules in the population.

Confidence: indicates the recurrence of AB within all the rules that have the same condition.

t1and t2 were used as thresholds to balance between support value and confidence value, assume that (t1 = 0.0257) and (t2 = 0.9843).

To get the accurate results, for each record in the population, fitness value 1 has been calculated using Fitness Function 1 and fitness value 2 has been calculated using Fitness Function 2, the second step is to find the values of R1 and R2 using the equations 8 and 9, the third step is to find the result of the following equation:

$$R3 = \frac{\sum_{i=1}^{N} R1 - R2}{N} \qquad (4.5)$$

Where,

N: the number of records in the population.

To judge that both Fitness Functions getting the same results in assigning the appropriate fitness value to each record in the population, the result of R3 must approach to zero.

4.4.4 Comparing Results

The system has been built for a population of 68 records of R2L attack. For each record in the population the system calculated the values of A, AB, Fitness Value 1, Fitness Value 2, R1 and R2.

Fitness Value 1 and R1 are related to the Reward-Penality based Fitness Function, whereas Fitness Value 2 and R2 are related to the Support-Confidence Framework Fitness Function.

The following table contains some of the records and their values:

A	AB	Fit. Val.1	Fit. Val 2	R1	R2
180930	11	0.016	0.004	0.005	0.004
0	1	3.001	0.985	0.953	0.961
0	51	3.071	1.004	0.975	0.979
0	2	3.003	0.985	0.953	0.961
0	4	3.006	0.986	0.954	0.962
0	3	3.004	0.985	0.954	0.962
0	4	3.006	0.986	0.954	0.962
0	6	3.008	0.987	0.955	0.963
0	16	3.022	0.990	0.959	0.966
0	37	3.052	0.998	0.969	0.974
0	91	3.127	1.019	0.993	0.994
0	107	3.15	1.025	1.000	1.000

Table (4.12): Real Data from the Comparison System

The results showed that R1 and R2 are close to each other. Finally, the result of R3 was calculated using equation (4.5) and the result was approached to Zero, (R3 = -0.0001).

4.5 Tracing the Selection process:

This section will present the tracing of many types of Steady State Genetic Algorithm Selection process. The tracing applied on the U2R table which has 16 dependent records. The tracing results suppose that there are two populations with size 8.The following table contains id and the fitness value for each record.

Chromosome ID	Fitness Value
1	1.311
2	1.163
3	1
4	1.348
5	3.167
6	1.667
7	3.167
8	3.167
9	3.167
10	3.267
11	3.333
12	3.333
13	3.167
14	3.167
15	3.167
16	2.333

Table (4.13): ID and fitness value for U2R chromosomes

The data is divided into two populations. For each population there are three values; summation of fitness values, average of fitness values and summation of the expected fitness values. Those values can be achieved using the following equations:

Sum Fitness (Sf), Average Fitness (Af) and Sum Expected Fitness Value (SumEf), as represented in the equations (2.6, 2.7, 2.9).

4.5.1 Trace with Roulette Wheel Selection

Here, there is a randomly generated number, then Sum Expected Fitness value (SumEf) calculation will be applied continuously until reaching to the record that makes the (SumEf) value greater than the generated number, then that record considered as the selected individual using RWS.

The Table (4.14) below, shows the tracing of generated random number and the selected individual corresponding to the generated number.

Counter	Random Num.	Selected Individual
1	6	7
2	5	7
3	1	2
4	4	6
5	6	7
6	5	7
7	1	2
8	4	6
9	0.999	9
10	5.999	14
11	4	12
12	0.999	9
13	5.999	14
14	6.999	15
15	4.999	13
16	7.999	15

Table (4.14): Generated random numbers with selected individual

According to the selected individual, Table (4.15) shows the final prepared population with RWS:

ID	Fitness Value
7	3.167
7	3.167
2	1.163
6	1.667
7	3.167
7	3.167
2	1.163
6	1.667
9	3.167
14	3.167
12	3.333
9	3.167
14	3.167
15	3.167
13	3.167
15	3.167

Table (4.15): Prepared population with RWS

4.5.2 Trace with Elitism Selection

This type of selection will arrange the population records in decreasing order according to their Fitness Values. Table (4.16) shows the result of Elitism Selection.

ID	Fitness Value
5	3.167
7	3.167
8	3.167
6	1.667
4	1.348
1	1.311
2	1.163
3	1
11	3.333
12	3.333
10	3.267
9	3.167
13	3.167
14	3.167
15	3.167
16	2.333

Table (4.16): The result of Elitism Selection

4.5.3 Trace with Ranking Selection

The idea of this type of Selection is to arrange each population record according to their fitness values, then give a rank to each record. The new fitness values calculated for the records using the equation (2.11):

Table (4.17) shows the result of ranking selection:

ID	Fitness Value	Rank	Max	Min	New Fitness Value
5	3.167	8	1.71	0.29	0.29
7	3.167	7	1.53	0.47	0.62
8	3.167	6	1.58	0.42	0.75
6	1.667	5	1.29	0.71	0.95
4	1.348	4	1.3	0.7	1.04
1	1.311	3	1.77	0.23	1.33
2	1.163	2	1.01	0.99	1.007
3	1	1	1.76	0.24	1.76
11	3.333	8	1.87	0.13	0.13
12	3.333	7	1.06	0.94	0.95
10	3.267	6	1.95	0.05	0.59
9	3.167	5	1.36	0.64	0.94
13	3.167	4	1.52	0.48	1.07
14	3.167	3	1.77	0.23	1.33
15	3.167	2	1.05	0.95	1.03
16	2.333	1	1.59	0.41	1.59

Table (4.17): The values of rank and new fitness values for each record using rank selection

After calculating the fitness values, RWS selection will be applied to get the selected individual. Table (4.18) shows the result of RWS applied on the new fitness values.

Counter	Random Number	Selected individual
1	15	2
2	4	6
3	4	6
4	8.03	4
5	3.59	6
6	13.07	2
7	4.03	6
8	13.59	2
9	19.95	14
10	2.95	11
11	9.90	10
12	11.90	9
13	26	15
14	0.95	11
15	3.86	12
16	3.86	12

Table (4.18): The selected individual after applying RWS over the new fitness values

4.5.4 Trace with Stochastic Universal Sampling (SUS):

The idea of this type of selection is to select individuals at specific points. The points determined previously. The record fitness value affects the Selection of the individual. Table (4.19) shows the result of selected individual and their fitness values:

Counter	Selected individual ID	Selected individual Fitness Value
1	1	1.311
2	2	1.163
3	4	1.348
4	5	3.167
5	6	1.667
6	7	3.167
7	7	3.167
8	8	3.167
9	9	3.167
10	9	3.167
11	10	3.267
12	11	3.333
13	12	3.333
14	13	3.167
15	14	3.167
16	15	3.167

Table (4.19): Selected ID's and their fitness values using SUS

4.5.5 Trace with Tournament Selection

The idea of this type of Selection is to choose two numbers randomly, then select two individuals using RWS. Finally, choose the record with the highest fitness values records. For each record in the population, Table (4.20) shows the result of randomly chosen numbers, the selected individuals corresponding to the random numbers and the selected record.

Counter	Random1	Random2	Selected ID
1	7	6	7
2	2	5	5
3	7	6	7
4	2	5	5
5	2	7	7
6	5	2	5
7	7	8	8
8	6	1	6
9	16	14	14
10	15	13	13
11	14	12	12
12	15	14	14
13	15	9	9
14	11	12	12
15	12	15	12
16	14	11	11

Table (4.20): Selected individuals using Tournament Selection

Table (4.21) shows the arrangement of selected individuals using many types of selection.

#	RWS	Elitism	Ranking	SUS	Tournament
1	7	5	3	1	7
2	7	7	6	2	5
3	2	8	6	4	7
4	6	6	4	5	5
5	7	4	6	6	7
6	7	1	2	7	5
7	2	2	6	7	8
8	6	3	3	8	6
9	9	11	14	9	14
10	14	12	11	9	13
11	12	10	10	10	12
12	9	9	9	11	14
13	14	13	16	12	9
14	15	14	11	13	12
15	13	15	12	14	12
16	15	16	12	14	11

Table (4.21): The arrangement of ID records due to the Selection type in the first generation

From the previous results, you can notice that the arrangement of the records in the populations depends mainly on the selection type. Notice that there aren't two Selection types have the same chromosomes arrangement; this note will be helpful in the next section.

4.6 Comparing between Selection and Crossover types.

As mentioned in section 2.4; the Steady State Genetic Algorithm has many stages. Each stage has many types. But when applying the algorithm for each stage, just one type can be taken to get the result.

This part of thesis will search for the best types to be used together with getting the best results. It will determine the Selection type and Crossover type that gives the best result when they combined together within Steady State Genetic Algorithm.

4.6.1 Comparing Strategy.

The detection system with Steady State Genetic Algorithm has been built. The parameters used in the system presented in the following table:

Population size	8
Representation	Real
Evaluation	Reward Penality Fitness Function
Selection	RWS, Ranking, Stochastic, Elitism, Tournament
Crossover	Single Point, Two Points, Uniform
Mutation	Flip Bit
Replacement	Binary Tournament Replacement
Stopping Criteria	When Genetic Algorithm Cannot discover additional Rules

Table (4.22): The parameters used in the system

The Steady State Genetic algorithm was applied with Roulette Wheel Selection and Single Point Crossover in the first trial, and then applied with Roulette Wheel Selection and Two Points Crossover in the second trial. And so on until applying Tournament Selection with Uniform Crossover in the 15th trial.

The results are observed for each trial in the 10th generation and 15th generation to ensure the judgment.

4.6.2 Comparison Results and Discussion.

After applying the GA with many generations, the following results have been gotten:

Choice	Selection	Crossover	After 10 Generations		After 15 Generations	
			# of records	DR	# of records	DR
1	RWS	One Point	94	***0.46***	229	***0.53***
2	RWS	Two Points	117	***0.53***	292	***0.53***
3	RWS	Uniform	112	***0.4***	344	***0.4***
4	Elitism	One Point	98	***0.4***	230	***0.4***
5	Elitism	Two Points	125	***0.4***	349	***0.53***
6	Elitism	Uniform	137	***0.53***	414	***0.53***
7	Ranking	One Point	87	***0.4***	198	***0.46***
8	Ranking	Two Points	91	***0.46***	242	***0.53***
9	Ranking	Uniform	110	***0.46***	346	***0.53***
10	SUS	One Point	98	***0.4***	230	***0.40***
11	SUS	Two Points	125	***0.4***	348	***0.46***
12	SUS	Uniform	137	***0.53***	414	***0.53***
13	Tournament	One Point	79	***0.46***	186	***0.46***
14	Tournament	Two Points	110	***0.4***	283	***0.46***
15	Tournament	Uniform	117	***0.4***	347	***0.46***

Table (4.23): The choices of different selection and crossover types

So, there are 15 different choices. Some are bad and some are good. The idea of using Steady State Genetic Algorithm is to discover hidden rules. So SSGA will discover all the hidden rules but the process will vary from choice to choice in the term of number of generations to discover the rules and the time consuming in discovering rules, Because the time consuming will be as high as the number of generations increase.

From the results, one can notice the following:

1- For each Selection process, the Two Point Crossover produced better results than One Point Crossover.
2- For each Selection process, the Uniform Crossover produced better results than Two Points Crossover.
3- Roulette Wheel Selection with One Point Crossover (Choice 1) produced better results than Tournament Selection with One Point Crossover (Choice 13). But Roulette Wheel Selection with Uniform Crossover (Choice 3) produced worse results than Tournament Selection with Uniform Crossover (Choice 15).
4- Elitism Selection (Choices 4,5 & 6) produced the same results as Stochastic Universal Sampling Selection (Choices 10,11 & 12)
5- Elitism Selection with Uniform Crossover (Choice 6) and Stochastic Universal Sampling Selection with Uniform Crossover (Choice 12) both produced the best results through different fifteen choices.

4.7 The Results of Detection Rate (DR) and False Positive Rate (FPR).

Intrusion Detection System has been built, and Steady State Genetic Algorithm has been used to support the system. DR was calculated using equation (2.1) and FPR was calculated using equation (2.2). Both values where calculated for each type of attack. The goal is to get DR as high as possible, and to get FPR as low as possible. If DR approaches to 100%, it means that system has a good DR. And if FPR approaches to 0%, it means that system has a good FPR. After system execution, the execution gave the following results.

Attack	R2L	U2R	DoS	Probe	Average
DR	100%	86%	94%	100%	95%
FPR	0%	0.03%	0.79%	0.37%	0.297%

Table (4.24): Results of system DR and FPR

4.8 Comparing thesis results with other results.

This part will compare thesis results with other results; the criteria are Average of the DR and Average of the FPR.

Alsharafat, (2009) found that the average DR is equal to 98.9%. Hence, the value of DR is greater than the results of this research. Stewart, (2009) found that the average of DR is equal to 79.67% which is less than this research results.

Al-sharafat (2009) found that the average FPR equal 0.094% which is better than the results of this research, but (Stewart, 2009) found that the average FPR is 2.69%, which is worse than the results of this research.

The comparison shows that this research achieved better results than (stewart,2009), but worse than (alsharafat, 2009). The comparison is clear in Table (4.25):

	Average of DR	Average of FPR
Our results	95%	0.29%
Alsharafat	98.9%	0.094%
Stewart	79.67%	2.69%

Table (4.25): Comparison with other results

These results ensure that the proposed model can be used in Intrusion Detection System to increase DR and to decrease FPR.

4.9 How did this thesis achieve its objectives?

This thesis has achieved its objectives as the following:

It enhanced a Steady State Genetic Algorithm by two methods. The first method is Reward Penality based Fitness Function, which is completely new. And the second method is comparing between Selection methods and Crossover operators to use the best choice in the SSGA.

This thesis has got a high DR, and a low FPR. The average of Detection Rate achieved was 95%, and the average of False Positive Rate achieved was 0.297%.

This thesis makes two types of comparison, internal and external. The internal comparison is between Selection methods and Crossover operators, which help the model in choosing the best choice which performs better. But the external comparison is done between this thesis results and other thesis results.

Chapter Five

Conclusion and Future Work

5.1 Conclusion

This research presents a solution for a problem of detecting attack. The main goal of this research is to enhance the SSGA for misuse NIDS in order to increase DR and decrease FPR. The proposed solution has many parts which play a role in achieving thesis results.

KDD Cup 99 has a huge amount of records, each record has 42 features. There are many researches that determined the most significant features. IDS tried to use the results of each research to determine the suitable research result.

Reward Penality based fitness function was proposed, to examine the function, it was compared with another function and produced results that is similar to the results produced by another function.

Different Selection methods and Crossover types was combined together and tested. The results show that Uniform Crossover is the best between Crossover types and it is better to be combined with SUS selection or Elitist Selection methods.

The results of comparing between Selection and Crossover type are similar when using some types, the tracing helps in ensuring that different selection types have different arrangement of the records.

The DR of the system was 95% whereas the FPR was 0.297%. The results of DR and FPR were compared with other results. The comparison results show that this research is accepted.

5.2 Future Work

The proposed model for Intrusion Detection System has been built and supported with Steady State Genetic Algorithm. But there are a lot of issues that must be taken in consideration in the future to enhance this thesis results.

1- Additional researches needed to find the DR and FPR for Normal behavior.

2- Additional attempts needed to compare the results of SSGA with many types of Replacement. i.e. Binary Tournament Replacement and Triple Tournament Replacement.

3- The population size must be determined for the IDS case, to use the most suitable size for the population.

4- There is a need to build an Intrusion Detection System which depends on the hybrid of Anomaly and Misuse analysis.

5- The research in this thesis should be applied on the Intrusion Detection Prevention System.

6- The research should be applied on another type of attacks.

7- Additional work must be done to find the effect of the SSGA process on the time and convergence.

References

Adewumi, A.O. (2010). “Some improved genetic-algorithms based heuristics for global optimization with innivative applications”.(master thesis). University of the witwatersrand. Johannesburg. South Africa. Available at: http://wiredspace.wits.ac.za/bitstream/handle/10539/8621/PhD%20Thesis%20Abstract%20only.pdf?sequence=6

Agravat,M.Rao,U.(2011). "Computer intrusion detection by two-objective fuzzy genetic algorithm". **First international conference on computer science , engineering and application (CCSEA)**. July (15-17), Hyatt regency channai, india. Available at: http://airccj.org/CSCP/vol1/csit1226.pdf

Alabsi,F. Naoum,R.(2012) "Fitness Function for Genetic Algorithm used in Intrusion Detection System". International Journal of Applied Science and Technology. Vol(2). No(4). PP (129-134), available at: http://www.ijastnet.com/journals/Vol_2_No_4_April_2012/17.pdf

Al-Sharafat,W. Naoum,R. (2009). Development of genetic-based machine learning for Network Intrusion Detection. World Academy of Science, Engineering and Technology,(55-2009), pages 20-24. Available at: http://www.waset.org/journals/waset/v55/v55-5.pdf

Al-Sharafat, Wafa' Slaibi (2009). *Development of genetic-based machine learning algorithm for network intrusion detection (gbml-nid)*. (doctorate dissertation) , The Arab Academy for banking and financial sciences, Amman, Jordan.

Berlanga.F.J., Del Jesus.M.J., Gatco.M.J., Herrera.F.,A Genetic-Programming-Based Approach for the Learning of Compact Fuzzy Rule-Based Classification Systems, the eighth International Conference on Artificial Intelligence and Soft Computing (ICAISC), Zakopane, Poland, on 25-29 June 2006, PP 182-191. From: http://sci2s.ugr.es/docencia/doctoMineriaDatos/Ber06-ICAIS.pdf

Bottaci,L.,2001, A Genetic Algorithm Fitness Function for mutation testing presented at SEMINAL 2001, International workshop on software engineering using Metaheuristic Innovative algorithm, a workshop at 23-rd Int. Conference on Software Engineering, Toronto, May 12-19. From: http://www2.hull.ac.uk/science/pdf/workshop.pdf

Chinneck,J.W. (2006). “Practical optimization: a gentle introduction. Canada. Carleton university”. Available at: http://www.sce.carleton.ca/faculty/chinneck/po/TitlePageAndTOC.pdf

Chou,T.S. Yen,K.K. Luo,J. (2008). "Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms". World Academy of Science, Engineering and Technology. No (47). Page (529).

Cleary, B.(2011). "Problems with crossover bias for binary string representations in genetic algorithms". Master thesis. Californai state university. Long beach. Californa. United states.

Das, V., Pathak, V. Sharma, S. Ravi, S. Srikanth, M. Kumar, G. (2010). "Network intrusion detection system based on machine learning algorithms". ***International Journal of Computer Science and Information Technology (IJCSIT)***. Vol (2). No (6). Page 139. Available at: http://airccse.org/journal/jcsit/1210ijcsit13.pdf

Deb, K.,(1998), Genetic Algorithm in search and optimization: the technique and applications, Proceedings of International Workshop on Soft Computing and Intelligent Systems, (ISI, Calcutta, India). PP. 58-87.

Diaz-Gomez,P. Hougen,D. (2006). A Genetic Algorithm Approach for Doing Misuse Detection in Audit Trial Files. 15th international conference on computing. On Nov-2006, Pages (329-338). Available at: http://www.cameron.edu/~pdiaz-go/diazp-Misuse.pdf

Diaz-Gomez,P. Hougen,D. (2007). Misuse Detection: An Iterative Process vs. A Genetic Algorithm Approach. On proceeding International Conference on Enterprise Information Systems(2). Available at: http://students.ou.edu/D/Pedro.A.Diaz-Gomez-1/Iter_GAsMisUseF.pdf

Garcia-Teodoro, P. Diaz-Verdejo, J. Macia-Fernandoz, G. Vazquez, E. (2009). Anomaly-based network intrusion detection: techniques, systems and challenge. ***ScienceDirect***. 28(2009). Pages 18-28. Available at: http://ceres.ugr.es/~gmacia/papers/COMSEC09_AnidsPublishedVersion.pdf

Ghali,N. (2009). Feature selection for effective anomaly based intrusion detection. International Journal for Computer Science and Network Security IJCSNS, VOL(9) ,No(3), pages 285-289. Available at: http://paper.ijcsns.org/07_book/200903/20090339.pdf

Hoque, M. Abdul Mukit, M. Bikas, M.(2012)."An Implementation of Intrusion Detection System using Genetic Algorithm". International Journal of Network Security & its Applications. Vol(4).No(2).PP(109-120).

Information Assurance Technology Analysis Center (IATAC). (2009). *Intrusion Detection System.* (Sixth Edition). USA.

Ibrahim, Laheeb Mohammad,(2010). Anomaly network intrusion detection system based on distributed time-delay neural network (DTDNN). ***Journal of Engineering Science and Technology.*** Vol (5). No (4). Pages 457-471. Available at: http://jestec.taylors.edu.my/Vol%205%20Issue%204%20December%2010/Vol_5_4_457_471_L.%20M.%20Ibrahim.pdf

Kang,D.Fuller,D.Honavar,V.(2005)."Learning classifier for misuse and anomaly detection using a bag of system calls representation", proceeding of the 6th IEEE, workshop on information assurance and security, NY, USA. Available at: http://www.cs.iastate.edu/~honavar/Papers/iaw05.pdf

Khan, S.(2011). "Rule based Network Intrusion Detection using Genetic Algorithm". International Journal of Computer Application. Vol (18). No(8).PP (26-29).

Kozushko, H. (2003). "Intrusion detection: host-based and network-based intrusion detection systems",(on-line)

Available:http://infohost.nmt.edu/~sfs/Students/HarleyKozushko/Papers/IntrusionDetectionPaper.pdf Viewed at: 13-Dec-2011.

Kumar, M., Husian, M., Upreti, N., Gupta, D. (2010). Genetic algorithm: review and application. ***International Journal of Information Technology and Knowledge Management***. Vol (2). No (2). Page 451. Available at: http://www.csjournals.com/IJITKM/PDF%203-1/55.pdf

Mukkamala, S., Sung, A., Abrham, A., (2004), Modeling Intrusion Detection System using Linear Genetic Programming Approach, Proceeding IEA/AIE 17th International Conference on Innovations in Applied Artificial Intelligence, PP 633-642, ISBN: 3-540-22007-0, From: http://www.rmltech.com/doclink/LGP%20Based%20IDS.pdf

Naoum, R., Abid, N., Al-Sultani,Z.(2012). "An Enhanced Resilient Back propagation Artificial Neural Network for Intrusion Detection System". International Journal of Computer Science and Network Security. Vol(12). No(3). PP (11-16).

Ritcher, J. (2010). On Mutation and Crossover in the Theory of Evolutionary Algorithms. (Doctorate Dissertation). Montana State University. Montana. United States. Available at: http://www.cs.montana.edu/files/techreports/0910/Richter.pdf

Scarfone, K., Mell, P. (2007). Guide to intrusion detection and prevention systems (IDPS). ***National Institute of Standards and Technology***. Special publication 800-94, Page 2-1. Available at: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf

Schmidt, M., Stidsen, T. (1997).Hybird systems:genetic algorithms,neural networks and fuzzy logic, DAIMI IR

Selvakani,S., Rajesh,R.S.(2007). Genetic algorithm for framing rules for intrusion detection. International Journal for Computer Science and Network Security IJCSNS, VOL(7), No(11), 285-290. Available at: http://paper.ijcsns.org/07_book/200711/20071144.pdf

Stewart, L. (2009). "A Modified Genetic Algorithm and Switch-Based Neural Network Model Applied To Misuse Based Intrusion Detection".(master thesis). Queens University. Ontario. Canada. Available at: http://qspace.library.queensu.ca/bitstream/1974/1720/1/Stewart_Ian_D_200903_MSc.pdf

Sung, A., Mukkamala, S. (2003), "Feature Selection for Intrusion Detection using Neural Networks and Support Vector Machines", *To appear in Journal of the Transportation Research Board (of the National Academies*). Available at: http://www.ltrc.lsu.edu/TRB_82/TRB2003-002459.pdf

Tavallaee,M., Bagheri,E., Lu,W., Ghorbani,A. (2009). A detailed analysis of the KDD Cup 99 data set. Proceedings of the 2009 IEEE symposium on computational intelligence in security and defense applications (CISDA 2009). Available at: http://www.tavallaee.com/publications/CISDA.pdf

Uppalaiah, B., Anand, K., Narsimha,B., Swaraj, S., Bharat, T.(2012). "Genetic Algorithm Approach to Intrusion Detection System". International Journal for Computer Science and Technology. Vol(3). Issue (1). PP(156-160).

Zainal, A.,Maarof, M., Shamsuddin, S., Abraham, A., (Sept – 2008)"Ensemble of One-class Classifier for Network Intrusion Detection System", Fourth International Conference on Information Assurance and Security (ISIAS '08). PP (180-185). Available at: http://www.softcomputing.net/ias08_1.pdf

Appendix: Code Listing

- Code for choosing the most significant features class

```
Dim danormal As New SqlDataAdapter("Select * from normal", CS4)
        Dim daprobe As New SqlDataAdapter("Select * from probe", CS4)
        Dim dados As New SqlDataAdapter("Select * from dos", CS4)
        Dim dau2r As New SqlDataAdapter("Select * from u2r", CS4)
        Dim dar2l As New SqlDataAdapter("Select * from r2l", CS4)

        Dim ds As New DataSet
        danormal.Fill(ds, "normal")
        daprobe.Fill(ds, "probe")
        dados.Fill(ds, "dos")
        dau2r.Fill(ds, "u2r")
        dar2l.Fill(ds, "r2l")
        datest.Fill(ds, "KDDtest$")

        Dim n5, n6, n10, n13, n40 As Double

        Dim p3 As String
        Dim p12, p27, p31, p35 As Double
        Dim d7, d8, d12, d13, d23 As Double
        Dim u14, u17, u25, u36, u38 As Double
        Dim r6, r11, r12, r19, r22 As Double

        Dim NoRowsTableTest As Integer
        Dim NoRowsTableNormal As Integer
        Dim NoRowsTableProbe As Integer
        Dim NoRowsTableDos As Integer
        Dim NoRowsTableU2R As Integer
        Dim NoRowsTableR2L As Integer

        NoRowsTableTest = ds.Tables("KDDtest$").Rows.Count
        NoRowsTableNormal = ds.Tables("Normal").Rows.Count
        NoRowsTableProbe = ds.Tables("Probe").Rows.Count
        NoRowsTableDos = ds.Tables("Dos").Rows.Count
        NoRowsTableU2R = ds.Tables("U2R").Rows.Count
        NoRowsTableR2L = ds.Tables("R2L").Rows.Count

        For TestCounter = 0 To NoRowsTableTest - 1

            n5 = ds.Tables("KDDtest$").Rows(TestCounter).Item(5)
            n6 = ds.Tables("KDDtest$").Rows(TestCounter).Item(6)
            n10 = ds.Tables("KDDtest$").Rows(TestCounter).Item(10)
            n13 = ds.Tables("KDDtest$").Rows(TestCounter).Item(13)
            n40 = ds.Tables("KDDtest$").Rows(TestCounter).Item(40)

            Dim normalcounter As Integer
            For normalcounter = 0 To NoRowsTableNormal - 1
                If ds.Tables("Normal").Rows(normalcounter).Item(1) = n5 Then
                    If ds.Tables("Normal").Rows(normalcounter).Item(2) = n6 Then
                    If ds.Tables("Normal").Rows(normalcounter).Item(3) = n10 Then
                    If ds.Tables("Normal").Rows(normalcounter).Item(4) = n13 Then
                    If ds.Tables("Normal").Rows(normalcounter).Item(5) = n40 Then
                    If ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "normal." Then
                              TxtNormal4Count.Text = Val(TxtNormal4Count.Text) + 1
                              Else
                              TxtNotNormal4Count.Text = Val(TxtNotNormal4Count.Text) + 1
                                    End If
                                    Exit For
                                End If
                            End If
                        End If
                    End If
                End If
            Next

            p3 = ds.Tables("KDDtest$").Rows(TestCounter).Item(3)
            p12 = ds.Tables("KDDtest$").Rows(TestCounter).Item(12)
            p27 = ds.Tables("KDDtest$").Rows(TestCounter).Item(27)
```

```
-               p31 = ds.Tables("KDDtest$").Rows(TestCounter).Item(31)
-               p35 = ds.Tables("KDDtest$").Rows(TestCounter).Item(35)
-
-               Dim ProbeCounter As Integer
-               For ProbeCounter = 0 To NoRowsTableProbe - 1
-                   If ds.Tables("Probe").Rows(ProbeCounter).Item(1) = p3 Then
-                       If ds.Tables("Probe").Rows(ProbeCounter).Item(2) = p12 Then
-                           If ds.Tables("Probe").Rows(ProbeCounter).Item(3) = p27 Then
-                               If ds.Tables("Probe").Rows(ProbeCounter).Item(4) = p31 Then
-                               If ds.Tables("Probe").Rows(ProbeCounter).Item(5) = p35 Then
-                                       If ds.Tables("KDDtest$").Rows(TestCounter).Item(42)
    = "ipsweep." Or ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "nmap." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "portsweep." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "satan." Then
-                                   TxtProbe4Count.Text = Val(TxtProbe4Count.Text) + 1
-                                   Else
-                                   TxtNotProbe4Count.Text = Val(TxtNotProbe4Count.Text) + 1
-                                       End If
-                                       Exit For
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               d7 = ds.Tables("KDDtest$").Rows(TestCounter).Item(7)
-               d8 = ds.Tables("KDDtest$").Rows(TestCounter).Item(8)
-               d12 = ds.Tables("KDDtest$").Rows(TestCounter).Item(12)
-               d13 = ds.Tables("KDDtest$").Rows(TestCounter).Item(13)
-               d23 = ds.Tables("KDDtest$").Rows(TestCounter).Item(23)
-
-               Dim DosCounter As Integer
-               For DosCounter = 0 To NoRowsTableDos - 1
-                   If ds.Tables("Dos").Rows(DosCounter).Item(1) = d7 Then
-                       If ds.Tables("Dos").Rows(DosCounter).Item(2) = d8 Then
-                           If ds.Tables("Dos").Rows(DosCounter).Item(3) = d12 Then
-                               If ds.Tables("Dos").Rows(DosCounter).Item(4) = d13 Then
-                                   If ds.Tables("Dos").Rows(DosCounter).Item(5) = d23 Then
-                                       If ds.Tables("KDDtest$").Rows(TestCounter).Item(42)
    = "back." Or ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "land." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "neptune." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "pod." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "smurf." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "teardrop." Then
-                                    TxtDos4Count.Text = Val(TxtDos4Count.Text) + 1
-                                    Else
-                                    TxtNotDos4Count.Text = Val(TxtNotDos4Count.Text) + 1
-                                       End If
-                                       Exit For
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               u14 = ds.Tables("KDDtest$").Rows(TestCounter).Item(14)
-               u17 = ds.Tables("KDDtest$").Rows(TestCounter).Item(17)
-               u25 = ds.Tables("KDDtest$").Rows(TestCounter).Item(25)
-               u36 = ds.Tables("KDDtest$").Rows(TestCounter).Item(36)
-               u38 = ds.Tables("KDDtest$").Rows(TestCounter).Item(38)
-
-               Dim U2RCounter As Integer
-               For U2RCounter = 0 To NoRowsTableU2R - 1
-                   If ds.Tables("U2R").Rows(U2RCounter).Item(1) = u14 Then
-                       If ds.Tables("U2R").Rows(U2RCounter).Item(2) = u17 Then
-                           If ds.Tables("U2R").Rows(U2RCounter).Item(3) = u25 Then
-                               If ds.Tables("U2R").Rows(U2RCounter).Item(4) = u38 Then
-                                   If ds.Tables("U2R").Rows(U2RCounter).Item(5) = u36 Then
-                                       If ds.Tables("KDDtest$").Rows(TestCounter).Item(42)
    = "buffer_overflow." Or ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "loadmodule."
    Or ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "perl." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "rootkit." Then
-                                    TxtU2R4Count.Text = Val(TxtU2R4Count.Text) + 1
-                                    Else
-                                    TxtNotU2R4Count.Text = Val(TxtNotU2R4Count.Text) + 1
-                                      End If
-                                      Exit For
-                                  End If
-                              End If
-                          End If
-                      End If
-                  End If
-              Next
-
-              r6 = ds.Tables("KDDtest$").Rows(TestCounter).Item(6)
-              r11 = ds.Tables("KDDtest$").Rows(TestCounter).Item(11)
-              r12 = ds.Tables("KDDtest$").Rows(TestCounter).Item(12)
-              r19 = ds.Tables("KDDtest$").Rows(TestCounter).Item(19)
-              r22 = ds.Tables("KDDtest$").Rows(TestCounter).Item(22)
-
-              Dim R2LCounter As Integer
-              For R2LCounter = 0 To NoRowsTableR2L - 1
-                  If ds.Tables("R2L").Rows(R2LCounter).Item(1) = r6 Then
-                      If ds.Tables("R2L").Rows(R2LCounter).Item(2) = r11 Then
-                          If ds.Tables("R2L").Rows(R2LCounter).Item(3) = r12 Then
-                              If ds.Tables("R2L").Rows(R2LCounter).Item(4) = r19 Then
-                                  If ds.Tables("R2L").Rows(R2LCounter).Item(5) = r22 Then
-                                      If ds.Tables("KDDtest$").Rows(TestCounter).Item(42)
    = "ftp_write." Or ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "guess_passwd." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "imap." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "multihop." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "phf." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "spy." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "warezclient." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "warezmaster." Then
-                                    TxtR2L4Count.Text = Val(TxtR2L4Count.Text) + 1
-                                    Else
-                                    TxtNotR2L4Count.Text = Val(TxtNotR2L4Count.Text) + 1
-                                      End If
-                                      Exit For
-                                  End If
-                              End If
-                          End If
-                      End If
-                  End If
-              Next
-          Next
```

- Code for calculating A and AB.

```
-   Private CSTrain As New SqlConnection("Data Source=MO3ATH-PC;Initial
    Catalog=master;Integrated Security=True")
-   Private datrain As New SqlDataAdapter("Select * from kddcup$", CSTrain)
-   Private dos As New SqlConnection("Data Source=MO3ATH-PC;Initial Catalog=S-DoS;Integrated
    Security=True")
-   Private u2r As New SqlConnection("Data Source=MO3ATH-PC;Initial Catalog=S-U2R;Integrated
    Security=True")
-   Private r2l As New SqlConnection("Data Source=MO3ATH-PC;Initial Catalog=S-R2L;Integrated
    Security=True")
-   Private probe As New SqlConnection("Data Source=MO3ATH-PC;Initial Catalog=S-
    Probe;Integrated Security=True")
-
-
-
-   Private Sub Button21_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button21.Click
-           Dim dau2r As New SqlDataAdapter("Select * from buffer_overflow", u2r)
-           Dim ds As New DataSet
-           dau2r.Fill(ds, "buffer_overflow")
-           datrain.Fill(ds, "kddcup$")
-
-           Dim u14, u17, u25, u36, u38 As Double
```

```
-
-           Dim uA, uAB, uid As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableU2R As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableU2R = ds.Tables("buffer_overflow").Rows.Count
-           For Attack = 0 To NoRowsTableU2R - 1
-
-               u14 = ds.Tables("buffer_overflow").Rows(Attack).Item(1)
-               u17 = ds.Tables("buffer_overflow").Rows(Attack).Item(2)
-               u25 = ds.Tables("buffer_overflow").Rows(Attack).Item(3)
-               u38 = ds.Tables("buffer_overflow").Rows(Attack).Item(4)
-               u36 = ds.Tables("buffer_overflow").Rows(Attack).Item(5)
-               uA = ds.Tables("buffer_overflow").Rows(Attack).Item(6)
-               uAB = ds.Tables("buffer_overflow").Rows(Attack).Item(7)
-               uid = ds.Tables("buffer_overflow").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If u14 = ds.Tables("kddcup$").Rows(KddCounter).Item(13) Then
-                   If u17 = ds.Tables("kddcup$").Rows(KddCounter).Item(16) Then
-                   If u25 = ds.Tables("kddcup$").Rows(KddCounter).Item(24) Then
-                   If u38 = ds.Tables("kddcup$").Rows(KddCounter).Item(37) Then
-                   If u36 = ds.Tables("kddcup$").Rows(KddCounter).Item(35) Then
-                If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "buffer_overflow." Then
-                                       uAB = uAB + 1
-                                   Else
-                                       uA = uA + 1
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-                   End If
-               Next
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = u2r
-               Ocmd.Parameters.AddWithValue("@id", uid)
-               Ocmd.Parameters.AddWithValue("@A", uA)
-               Ocmd.Parameters.AddWithValue("@AB", uAB)
-               Ocmd.CommandText = "updateu2r1"
-               Try
-                   u2r.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               u2r.Close()
-           Next
-           TxtU2RDone.Text = "Done"
-       End Sub
-
-       Private Sub Button20_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button20.Click
-           Dim dau2r As New SqlDataAdapter("Select * from rootkit", u2r)
-           Dim ds As New DataSet
-           dau2r.Fill(ds, "rootkit")
-           datrain.Fill(ds, "kddcup$")
-
-           Dim u14, u17, u25, u36, u38 As Double
-
-           Dim uA, uAB, uid As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableU2R As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableU2R = ds.Tables("rootkit").Rows.Count
-           For Attack = 0 To NoRowsTableU2R - 1
```

```
-
-               u14 = ds.Tables("rootkit").Rows(Attack).Item(1)
-               u17 = ds.Tables("rootkit").Rows(Attack).Item(2)
-               u25 = ds.Tables("rootkit").Rows(Attack).Item(3)
-               u38 = ds.Tables("rootkit").Rows(Attack).Item(4)
-               u36 = ds.Tables("rootkit").Rows(Attack).Item(5)
-               uA = ds.Tables("rootkit").Rows(Attack).Item(6)
-               uAB = ds.Tables("rootkit").Rows(Attack).Item(7)
-               uid = ds.Tables("rootkit").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If u14 = ds.Tables("kddcup$").Rows(KddCounter).Item(13) Then
-                   If u17 = ds.Tables("kddcup$").Rows(KddCounter).Item(16) Then
-                   If u25 = ds.Tables("kddcup$").Rows(KddCounter).Item(24) Then
-                   If u38 = ds.Tables("kddcup$").Rows(KddCounter).Item(37) Then
-                   If u36 = ds.Tables("kddcup$").Rows(KddCounter).Item(35) Then
-                   If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "rootkit." Then
-                                           uAB = uAB + 1
-                                       Else
-                                           uA = uA + 1
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = u2r
-               Ocmd.Parameters.AddWithValue("@id", uid)
-               Ocmd.Parameters.AddWithValue("@A", uA)
-               Ocmd.Parameters.AddWithValue("@AB", uAB)
-               Ocmd.CommandText = "updateu2r2"
-               Try
-                   u2r.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               u2r.Close()
-           Next
-           TxtU2RDone.Text = "Done"
-       End Sub
-
-       Private Sub Button22_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button22.Click
-           Dim dar2l As New SqlDataAdapter("Select * from phf", r2l)
-
-           Dim ds As New DataSet
-
-           dar2l.Fill(ds, "phf")
-           datrain.Fill(ds, "kddcup$")
-
-           Dim r6, r11, r12, r19, r22 As Double
-           Dim rA, rAB, rid As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableR2L As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableR2L = ds.Tables("phf").Rows.Count
-           For Attack = 0 To NoRowsTableR2L - 1
-
-               r6 = ds.Tables("phf").Rows(Attack).Item(1)
-               r11 = ds.Tables("phf").Rows(Attack).Item(2)
-               r12 = ds.Tables("phf").Rows(Attack).Item(3)
-               r19 = ds.Tables("phf").Rows(Attack).Item(4)
-               r22 = ds.Tables("phf").Rows(Attack).Item(5)
-               rA = ds.Tables("phf").Rows(Attack).Item(6)
-               rAB = ds.Tables("phf").Rows(Attack).Item(7)
-               rid = ds.Tables("phf").Rows(Attack).Item(8)
```

```
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If r6 = ds.Tables("kddcup$").Rows(KddCounter).Item(5) Then
-                       If r11 = ds.Tables("kddcup$").Rows(KddCounter).Item(10) Then
-                           If r12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                               If r19 = ds.Tables("kddcup$").Rows(KddCounter).Item(18) Then
-                               If r22 = ds.Tables("kddcup$").Rows(KddCounter).Item(21) Then
-                            If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "phf." Then
-                                       rAB = rAB + 1
-                                   Else
-                                       rA = rA + 1
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               End If
-               Next
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = r2l
-               Ocmd.Parameters.AddWithValue("@id", rid)
-               Ocmd.Parameters.AddWithValue("@A", rA)
-               Ocmd.Parameters.AddWithValue("@AB", rAB)
-               Ocmd.CommandText = "updater2l"
-               Try
-                   r2l.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               r2l.Close()
-           Next
-           TxtR2LDone.Text = "Done"
-       End Sub
-
-       Private Sub Button5_Click_1(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button5.Click
-           Dim daprobe As New SqlDataAdapter("Select * from ipsweep", probe)
-
-           Dim ds As New DataSet
-           daprobe.Fill(ds, "ipsweep")
-           datrain.Fill(ds, "kddcup$")
-
-
-           Dim p3 As String
-           Dim p12, p27, p31, p35 As Double
-           Dim pA, pAB, pid As Integer
-
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableProbe As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableProbe = ds.Tables("ipsweep").Rows.Count
-
-           For Attack = 0 To NoRowsTableProbe - 1
-
-               p3 = ds.Tables("ipsweep").Rows(Attack).Item(1)
-               p12 = ds.Tables("ipsweep").Rows(Attack).Item(2)
-               p27 = ds.Tables("ipsweep").Rows(Attack).Item(3)
-               p31 = ds.Tables("ipsweep").Rows(Attack).Item(4)
-               p35 = ds.Tables("ipsweep").Rows(Attack).Item(5)
-               pA = ds.Tables("ipsweep").Rows(Attack).Item(6)
-               pAB = ds.Tables("ipsweep").Rows(Attack).Item(7)
-               pid = ds.Tables("ipsweep").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If p3 = ds.Tables("kddcup$").Rows(KddCounter).Item(2) Then
-                       If p12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                           If p27 = ds.Tables("kddcup$").Rows(KddCounter).Item(26) Then
-                               If p31 = ds.Tables("kddcup$").Rows(KddCounter).Item(30) Then
-                                   If p35 = ds.Tables("kddcup$").Rows(KddCounter).Item(34) Then
-                            If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "ipsweep." Then
-                                           pAB = pAB + 1
-                                       Else
-                                           pA = pA + 1
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = probe
-               Ocmd.Parameters.AddWithValue("@id", pid)
-               Ocmd.Parameters.AddWithValue("@A", pA)
-               Ocmd.Parameters.AddWithValue("@AB", pAB)
-               Ocmd.CommandText = "updateprobe1"
-               Try
-                   probe.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               probe.Close()
-           Next
-           TxtProbeDone.Text = "Done"
-       End Sub
-
-       Private Sub Button2_Click_1(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button2.Click
-           Dim daprobe As New SqlDataAdapter("Select * from portsweep", probe)
-
-           Dim ds As New DataSet
-           daprobe.Fill(ds, "portsweep")
-           datrain.Fill(ds, "kddcup$")
-
-
-           Dim p3 As String
-           Dim p12, p27, p31, p35 As Double
-           Dim pA, pAB, pid As Integer
-
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableProbe As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableProbe = ds.Tables("portsweep").Rows.Count
-
-           For Attack = 0 To NoRowsTableProbe - 1
-
-               p3 = ds.Tables("portsweep").Rows(Attack).Item(1)
-               p12 = ds.Tables("portsweep").Rows(Attack).Item(2)
-               p27 = ds.Tables("portsweep").Rows(Attack).Item(3)
-               p31 = ds.Tables("portsweep").Rows(Attack).Item(4)
-               p35 = ds.Tables("portsweep").Rows(Attack).Item(5)
-               pA = ds.Tables("portsweep").Rows(Attack).Item(6)
-               pAB = ds.Tables("portsweep").Rows(Attack).Item(7)
-               pid = ds.Tables("portsweep").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If p3 = ds.Tables("kddcup$").Rows(KddCounter).Item(2) Then
-                       If p12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                           If p27 = ds.Tables("kddcup$").Rows(KddCounter).Item(26) Then
-                               If p31 = ds.Tables("kddcup$").Rows(KddCounter).Item(30) Then
-                                   If p35 = ds.Tables("kddcup$").Rows(KddCounter).Item(34) Then
-                      If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "portsweep." Then
-                                           pAB = pAB + 1
-                                       Else
-                                           pA = pA + 1
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = probe
-               Ocmd.Parameters.AddWithValue("@id", pid)
-               Ocmd.Parameters.AddWithValue("@A", pA)
-               Ocmd.Parameters.AddWithValue("@AB", pAB)
-               Ocmd.CommandText = "updateprobe2"
-               Try
-                   probe.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               probe.Close()
-           Next
-           TxtProbeDone.Text = "Done"
-       End Sub
-
-       Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
-           Dim daprobe As New SqlDataAdapter("Select * from satan", probe)
-
-           Dim ds As New DataSet
-           daprobe.Fill(ds, "satan")
-           datrain.Fill(ds, "kddcup$")
-
-
-           Dim p3 As String
-           Dim p12, p27, p31, p35 As Double
-           Dim pA, pAB, pid As Integer
-
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableProbe As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableProbe = ds.Tables("satan").Rows.Count
-
-           For Attack = 0 To NoRowsTableProbe - 1
-
-               p3 = ds.Tables("satan").Rows(Attack).Item(1)
-               p12 = ds.Tables("satan").Rows(Attack).Item(2)
-               p27 = ds.Tables("satan").Rows(Attack).Item(3)
-               p31 = ds.Tables("satan").Rows(Attack).Item(4)
-               p35 = ds.Tables("satan").Rows(Attack).Item(5)
-               pA = ds.Tables("satan").Rows(Attack).Item(6)
-               pAB = ds.Tables("satan").Rows(Attack).Item(7)
-               pid = ds.Tables("satan").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If p3 = ds.Tables("kddcup$").Rows(KddCounter).Item(2) Then
-                       If p12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                           If p27 = ds.Tables("kddcup$").Rows(KddCounter).Item(26) Then
-                               If p31 = ds.Tables("kddcup$").Rows(KddCounter).Item(30) Then
-                               If p35 = ds.Tables("kddcup$").Rows(KddCounter).Item(34) Then
-                          If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "satan." Then
-                                           pAB = pAB + 1
-                                       Else
-                                           pA = pA + 1
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = probe
-               Ocmd.Parameters.AddWithValue("@id", pid)
-               Ocmd.Parameters.AddWithValue("@A", pA)
-               Ocmd.Parameters.AddWithValue("@AB", pAB)
-               Ocmd.CommandText = "updateprobe3"
-               Try
-                   probe.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               probe.Close()
-           Next
-           TxtProbeDone.Text = "Done"
-       End Sub
-
-       Private Sub Button19_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button19.Click
-
-           Dim dados As New SqlDataAdapter("Select * from back", dos)
-           Dim ds As New DataSet
-           dados.Fill(ds, "back")
-           datrain.Fill(ds, "kddcup$")
-
-           Dim d7, d8, d12, d13, d23 As Double
-           Dim dA, dAB, did As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableDos As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableDos = ds.Tables("back").Rows.Count
-
-           For Attack = 0 To NoRowsTableDos - 1
-
-               d7 = ds.Tables("back").Rows(Attack).Item(1)
-               d8 = ds.Tables("back").Rows(Attack).Item(2)
-               d12 = ds.Tables("back").Rows(Attack).Item(3)
-               d13 = ds.Tables("back").Rows(Attack).Item(4)
-               d23 = ds.Tables("back").Rows(Attack).Item(5)
-               dA = ds.Tables("back").Rows(Attack).Item(6)
-               dAB = ds.Tables("back").Rows(Attack).Item(7)
-               did = ds.Tables("back").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If d7 = ds.Tables("kddcup$").Rows(KddCounter).Item(6) Then
-                       If d8 = ds.Tables("kddcup$").Rows(KddCounter).Item(7) Then
-                           If d12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                               If d13 = ds.Tables("kddcup$").Rows(KddCounter).Item(12) Then
-                               If d23 = ds.Tables("kddcup$").Rows(KddCounter).Item(22) Then
-                           If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "back." Then
-                                       dAB = dAB + 1
-                                   Else
-                                       dA = dA + 1
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = dos
-               Ocmd.Parameters.AddWithValue("@id", did)
-               Ocmd.Parameters.AddWithValue("@A", dA)
-               Ocmd.Parameters.AddWithValue("@AB", dAB)
-               Ocmd.CommandText = "updatedos1"
-               Try
-                   dos.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               dos.Close()
-           Next
-           TxtDosDone.Text = "Done"
-       End Sub
-
-       Private Sub Button18_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button18.Click
-           Dim dados As New SqlDataAdapter("Select * from land", dos)
-           Dim ds As New DataSet
-           dados.Fill(ds, "land")
-           datrain.Fill(ds, "kddcup$")
-
-
-           Dim d7, d8, d12, d13, d23 As Double
-           Dim dA, dAB, did As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableDos As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableDos = ds.Tables("land").Rows.Count
-
-           For Attack = 0 To NoRowsTableDos - 1
-
-               d7 = ds.Tables("land").Rows(Attack).Item(1)
-               d8 = ds.Tables("land").Rows(Attack).Item(2)
-               d12 = ds.Tables("land").Rows(Attack).Item(3)
-               d13 = ds.Tables("land").Rows(Attack).Item(4)
-               d23 = ds.Tables("land").Rows(Attack).Item(5)
-               dA = ds.Tables("land").Rows(Attack).Item(6)
-               dAB = ds.Tables("land").Rows(Attack).Item(7)
-               did = ds.Tables("land").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If d7 = ds.Tables("kddcup$").Rows(KddCounter).Item(6) Then
-                       If d8 = ds.Tables("kddcup$").Rows(KddCounter).Item(7) Then
-                           If d12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                               If d13 = ds.Tables("kddcup$").Rows(KddCounter).Item(12) Then
-                               If d23 = ds.Tables("kddcup$").Rows(KddCounter).Item(22) Then
-                           If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "land." Then
-                                           dAB = dAB + 1
-                                       Else
-                                           dA = dA + 1
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = dos
-               Ocmd.Parameters.AddWithValue("@id", did)
-               Ocmd.Parameters.AddWithValue("@A", dA)
-               Ocmd.Parameters.AddWithValue("@AB", dAB)
-               Ocmd.CommandText = "updatedos2"
-               Try
-                   dos.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               dos.Close()
-           Next
-           TxtDosDone.Text = "Done"
-       End Sub
-
-       Private Sub Button17_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button17.Click
-           Dim dados As New SqlDataAdapter("Select * from neptune", dos)
-           Dim ds As New DataSet
-           dados.Fill(ds, "neptune")
-           datrain.Fill(ds, "kddcup$")
-
-
-           Dim d7, d8, d12, d13, d23 As Double
-           Dim dA, dAB, did As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableDos As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableDos = ds.Tables("neptune").Rows.Count
-
-           For Attack = 0 To NoRowsTableDos - 1
-
-               d7 = ds.Tables("neptune").Rows(Attack).Item(1)
-               d8 = ds.Tables("neptune").Rows(Attack).Item(2)
-               d12 = ds.Tables("neptune").Rows(Attack).Item(3)
-               d13 = ds.Tables("neptune").Rows(Attack).Item(4)
-               d23 = ds.Tables("neptune").Rows(Attack).Item(5)
-               dA = ds.Tables("neptune").Rows(Attack).Item(6)
-               dAB = ds.Tables("neptune").Rows(Attack).Item(7)
-               did = ds.Tables("neptune").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If d7 = ds.Tables("kddcup$").Rows(KddCounter).Item(6) Then
-                       If d8 = ds.Tables("kddcup$").Rows(KddCounter).Item(7) Then
-                           If d12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                               If d13 = ds.Tables("kddcup$").Rows(KddCounter).Item(12) Then
-                               If d23 = ds.Tables("kddcup$").Rows(KddCounter).Item(22) Then
-                        If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "neptune." Then
-                                           dAB = dAB + 1
-                                       Else
-                                           dA = dA + 1
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = dos
-               Ocmd.Parameters.AddWithValue("@id", did)
-               Ocmd.Parameters.AddWithValue("@A", dA)
-               Ocmd.Parameters.AddWithValue("@AB", dAB)
-               Ocmd.CommandText = "updatedos3"
-               Try
-                   dos.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               dos.Close()
-           Next
-           TxtDosDone.Text = "Done"
-       End Sub
-
-       Private Sub Button16_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button16.Click
-           Dim dados As New SqlDataAdapter("Select * from pod", dos)
-           Dim ds As New DataSet
-           dados.Fill(ds, "pod")
-           datrain.Fill(ds, "kddcup$")
-
-
-           Dim d7, d8, d12, d13, d23 As Double
-           Dim dA, dAB, did As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableDos As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableDos = ds.Tables("pod").Rows.Count
-
-           For Attack = 0 To NoRowsTableDos - 1
-
-               d7 = ds.Tables("pod").Rows(Attack).Item(1)
-               d8 = ds.Tables("pod").Rows(Attack).Item(2)
-               d12 = ds.Tables("pod").Rows(Attack).Item(3)
-               d13 = ds.Tables("pod").Rows(Attack).Item(4)
-               d23 = ds.Tables("pod").Rows(Attack).Item(5)
-               dA = ds.Tables("pod").Rows(Attack).Item(6)
-               dAB = ds.Tables("pod").Rows(Attack).Item(7)
-               did = ds.Tables("pod").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If d7 = ds.Tables("kddcup$").Rows(KddCounter).Item(6) Then
-                       If d8 = ds.Tables("kddcup$").Rows(KddCounter).Item(7) Then
-                           If d12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                               If d13 = ds.Tables("kddcup$").Rows(KddCounter).Item(12) Then
-                               If d23 = ds.Tables("kddcup$").Rows(KddCounter).Item(22) Then
-                            If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "pod." Then
-                                       dAB = dAB + 1
-                                   Else
-                                       dA = dA + 1
-                                   End If
-                               End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = dos
-               Ocmd.Parameters.AddWithValue("@id", did)
-               Ocmd.Parameters.AddWithValue("@A", dA)
-               Ocmd.Parameters.AddWithValue("@AB", dAB)
-               Ocmd.CommandText = "updatedos4"
-               Try
-                   dos.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               dos.Close()
-           Next
-           TxtDosDone.Text = "Done"
-       End Sub
-
-       Private Sub Button15_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button15.Click
-           Dim dados As New SqlDataAdapter("Select * from smurf", dos)
-           Dim ds As New DataSet
-           dados.Fill(ds, "smurf")
-           datrain.Fill(ds, "kddcup$")
-
-
-           Dim d7, d8, d12, d13, d23 As Double
-           Dim dA, dAB, did As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableDos As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableDos = ds.Tables("smurf").Rows.Count
-
-           For Attack = 0 To NoRowsTableDos - 1
-
-               d7 = ds.Tables("smurf").Rows(Attack).Item(1)
-               d8 = ds.Tables("smurf").Rows(Attack).Item(2)
-               d12 = ds.Tables("smurf").Rows(Attack).Item(3)
-               d13 = ds.Tables("smurf").Rows(Attack).Item(4)
-               d23 = ds.Tables("smurf").Rows(Attack).Item(5)
-               dA = ds.Tables("smurf").Rows(Attack).Item(6)
-               dAB = ds.Tables("smurf").Rows(Attack).Item(7)
-               did = ds.Tables("smurf").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If d7 = ds.Tables("kddcup$").Rows(KddCounter).Item(6) Then
-                       If d8 = ds.Tables("kddcup$").Rows(KddCounter).Item(7) Then
-                           If d12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                               If d13 = ds.Tables("kddcup$").Rows(KddCounter).Item(12) Then
-                               If d23 = ds.Tables("kddcup$").Rows(KddCounter).Item(22) Then
-                          If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "smurf." Then
-                                           dAB = dAB + 1
-                                       Else
-                                           dA = dA + 1
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = dos
-               Ocmd.Parameters.AddWithValue("@id", did)
-               Ocmd.Parameters.AddWithValue("@A", dA)
-               Ocmd.Parameters.AddWithValue("@AB", dAB)
-               Ocmd.CommandText = "updatedos5"
-               Try
-                   dos.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               dos.Close()
-           Next
-           TxtDosDone.Text = "Done"
-       End Sub
-
-       Private Sub Button6_Click_1(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button6.Click
-           Dim dados As New SqlDataAdapter("Select * from teardrop", dos)
-           Dim ds As New DataSet
-           dados.Fill(ds, "teardrop")
-           datrain.Fill(ds, "kddcup$")
-
-
-           Dim d7, d8, d12, d13, d23 As Double
-           Dim dA, dAB, did As Integer
-
-           Dim NoRowsTablekddcup As Integer
-           Dim NoRowsTableDos As Integer
-           Dim Attack As Integer
-
-           NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-           NoRowsTableDos = ds.Tables("teardrop").Rows.Count
-
-           For Attack = 0 To NoRowsTableDos - 1
-
```

```
-               d7 = ds.Tables("teardrop").Rows(Attack).Item(1)
-               d8 = ds.Tables("teardrop").Rows(Attack).Item(2)
-               d12 = ds.Tables("teardrop").Rows(Attack).Item(3)
-               d13 = ds.Tables("teardrop").Rows(Attack).Item(4)
-               d23 = ds.Tables("teardrop").Rows(Attack).Item(5)
-               dA = ds.Tables("teardrop").Rows(Attack).Item(6)
-               dAB = ds.Tables("teardrop").Rows(Attack).Item(7)
-               did = ds.Tables("teardrop").Rows(Attack).Item(8)
-
-               For KddCounter = 0 To NoRowsTablekddcup - 1
-                   If d7 = ds.Tables("kddcup$").Rows(KddCounter).Item(6) Then
-                       If d8 = ds.Tables("kddcup$").Rows(KddCounter).Item(7) Then
-                           If d12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                               If d13 = ds.Tables("kddcup$").Rows(KddCounter).Item(12) Then
-                               If d23 = ds.Tables("kddcup$").Rows(KddCounter).Item(22) Then
-                       If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "teardrop." Then
-                                           dAB = dAB + 1
-                                       Else
-                                           dA = dA + 1
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   End If
-               Next
-
-               Dim Ocmd As New Data.SqlClient.SqlCommand
-               Ocmd.CommandType = CommandType.StoredProcedure
-               Ocmd.Connection = dos
-               Ocmd.Parameters.AddWithValue("@id", did)
-               Ocmd.Parameters.AddWithValue("@A", dA)
-               Ocmd.Parameters.AddWithValue("@AB", dAB)
-               Ocmd.CommandText = "updatedos6"
-               Try
-                   dos.Open()
-                   Ocmd.ExecuteNonQuery()
-               Catch ex As Exception
-               End Try
-               dos.Close()
-           Next
-           TxtDosDone.Text = "Done"
-       End Sub
```

- Code for calculating Fitness Function

```
Private Sub Button21_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button21.Click
        Dim dau2r As New SqlDataAdapter("Select * from buffer_overflow", u2r)
        Dim ds As New DataSet
        dau2r.Fill(ds, "buffer_overflow")
        Dim uA, uAB, uid As Integer
        Dim uFitnessValue As Double
        Dim NoRowsTableU2R As Integer
        NoRowsTableU2R = ds.Tables("buffer_overflow").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTableU2R - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTableU2R < FinalPopValue Then
                FinalPopValue = NoRowsTableU2R - 1
            End If
            For j = FirstPopValue To FinalPopValue

                If ds.Tables("buffer_overflow").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("buffer_overflow").Rows(j).Item(6)
                End If
                If ds.Tables("buffer_overflow").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("buffer_overflow").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                uA = ds.Tables("buffer_overflow").Rows(y).Item(6)
                uAB = ds.Tables("buffer_overflow").Rows(y).Item(7)
                uid = ds.Tables("buffer_overflow").Rows(y).Item(8)
                uFitnessValue = 2 + ((uAB - uA) / (uA + uAB)) + uAB / maxAB - uA / maxA
                uFitnessValue = Double.Parse(uFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = u2r
                Ocmd.Parameters.AddWithValue("@id", uid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", uFitnessValue)
                Ocmd.CommandText = "fitnessu2r1"
                Try
                    u2r.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                u2r.Close()
            Next
        Next
        TxtU2RDone.Text = "Done"
    End Sub

    Private Sub Button20_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button20.Click
        Dim dau2r As New SqlDataAdapter("Select * from rootkit", u2r)
        Dim ds As New DataSet
        dau2r.Fill(ds, "rootkit")
        Dim uA, uAB, uid As Integer
        Dim uFitnessValue As Double
        Dim NoRowsTableU2R As Integer
        NoRowsTableU2R = ds.Tables("rootkit").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTableU2R - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTableU2R < FinalPopValue Then
                FinalPopValue = NoRowsTableU2R - 1
            End If
            For j = FirstPopValue To FinalPopValue

                If ds.Tables("rootkit").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("rootkit").Rows(j).Item(6)
                End If
                If ds.Tables("rootkit").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("rootkit").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                uA = ds.Tables("rootkit").Rows(y).Item(6)
                uAB = ds.Tables("rootkit").Rows(y).Item(7)
                uid = ds.Tables("rootkit").Rows(y).Item(8)
                uFitnessValue = 2 + ((uAB - uA) / (uA + uAB)) + uAB / maxAB - uA / maxA
                uFitnessValue = Double.Parse(uFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = u2r
                Ocmd.Parameters.AddWithValue("@id", uid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", uFitnessValue)
                Ocmd.CommandText = "fitnessu2r2"
                Try
                    u2r.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                u2r.Close()
            Next
        Next
        TxtU2RDone.Text = "Done"
    End Sub

    Private Sub Button22_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button22.Click
        Dim dar2l As New SqlDataAdapter("Select * from phf", r2l)
        Dim ds As New DataSet
        dar2l.Fill(ds, "phf")
        Dim rA, rAB, rid As Integer
        Dim rFitnessValue As Double
        Dim NoRowsTableR2L As Integer
        NoRowsTableR2L = ds.Tables("phf").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTableR2L - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTableR2L < FinalPopValue Then
                FinalPopValue = NoRowsTableR2L - 1
            End If
            For j = FirstPopValue To FinalPopValue

                If ds.Tables("phf").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("phf").Rows(j).Item(6)
                End If
                If ds.Tables("phf").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("phf").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                rA = ds.Tables("phf").Rows(y).Item(6)
                rAB = ds.Tables("phf").Rows(y).Item(7)
                rid = ds.Tables("phf").Rows(y).Item(8)
                rFitnessValue = 2 + ((rAB - rA) / (rA + rAB)) + rAB / maxAB - rA / maxA
                rFitnessValue = Double.Parse(rFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = r2l
                Ocmd.Parameters.AddWithValue("@id", rid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", rFitnessValue)
                Ocmd.CommandText = "fitnessr2l"
                Try
                    r2l.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                r2l.Close()
            Next
        Next
        TxtR2LDone.Text = "Done"
    End Sub
    Private Sub Button5_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button5.Click
        Dim daprobe As New SqlDataAdapter("Select * from ipsweep", probe)
        Dim ds As New DataSet
        daprobe.Fill(ds, "ipsweep")
        Dim pA, pAB, pid As Integer
        Dim pFitnessValue As Double
        Dim NoRowsTableprobe As Integer
        NoRowsTableprobe = ds.Tables("ipsweep").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTableprobe - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTableprobe < FinalPopValue Then
                FinalPopValue = NoRowsTableprobe - 1
            End If
            For j = FirstPopValue To FinalPopValue
                If ds.Tables("ipsweep").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("ipsweep").Rows(j).Item(6)
                End If
                If ds.Tables("ipsweep").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("ipsweep").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                pA = ds.Tables("ipsweep").Rows(y).Item(6)
                pAB = ds.Tables("ipsweep").Rows(y).Item(7)
                pid = ds.Tables("ipsweep").Rows(y).Item(8)
                pFitnessValue = 2 + ((pAB - pA) / (pA + pAB)) + pAB / maxAB - pA / maxA
                pFitnessValue = Double.Parse(pFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = probe
                Ocmd.Parameters.AddWithValue("@id", pid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", pFitnessValue)
                Ocmd.CommandText = "fitnessprobe1"
                Try
                    probe.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                probe.Close()
            Next
        Next
        TxtProbeDone.Text = "Done"
    End Sub
    Private Sub Button2_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        Dim daprobe As New SqlDataAdapter("Select * from portsweep", probe)
        Dim ds As New DataSet
        daprobe.Fill(ds, "portsweep")
        Dim pA, pAB, pid As Integer
        Dim pFitnessValue As Double
        Dim NoRowsTableprobe As Integer
        NoRowsTableprobe = ds.Tables("portsweep").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTableprobe - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTableprobe < FinalPopValue Then
                FinalPopValue = NoRowsTableprobe - 1
            End If
            For j = FirstPopValue To FinalPopValue
                If ds.Tables("portsweep").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("portsweep").Rows(j).Item(6)
                End If
                If ds.Tables("portsweep").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("portsweep").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                pA = ds.Tables("portsweep").Rows(y).Item(6)
                pAB = ds.Tables("portsweep").Rows(y).Item(7)
                pid = ds.Tables("portsweep").Rows(y).Item(8)
                pFitnessValue = 2 + ((pAB - pA) / (pA + pAB)) + pAB / maxAB - pA / maxA
                pFitnessValue = Double.Parse(pFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = probe
                Ocmd.Parameters.AddWithValue("@id", pid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", pFitnessValue)
                Ocmd.CommandText = "fitnessprobe2"
                Try
                    probe.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                probe.Close()
            Next
        Next
        TxtProbeDone.Text = "Done"
    End Sub
    Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim daprobe As New SqlDataAdapter("Select * from satan", probe)
        Dim ds As New DataSet
        daprobe.Fill(ds, "satan")
        Dim pA, pAB, pid As Integer
        Dim pFitnessValue As Double
        Dim NoRowsTableprobe As Integer
        NoRowsTableprobe = ds.Tables("satan").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTableprobe - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTableprobe < FinalPopValue Then
                FinalPopValue = NoRowsTableprobe - 1
            End If
            For j = FirstPopValue To FinalPopValue
                If ds.Tables("satan").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("satan").Rows(j).Item(6)
                End If
                If ds.Tables("satan").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("satan").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                pA = ds.Tables("satan").Rows(y).Item(6)
                pAB = ds.Tables("satan").Rows(y).Item(7)
                pid = ds.Tables("satan").Rows(y).Item(8)
                pFitnessValue = 2 + ((pAB - pA) / (pA + pAB)) + pAB / maxAB - pA / maxA
                pFitnessValue = Double.Parse(pFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = probe
                Ocmd.Parameters.AddWithValue("@id", pid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", pFitnessValue)
                Ocmd.CommandText = "fitnessprobe3"
                Try
                    probe.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                probe.Close()
            Next
        Next
        TxtProbeDone.Text = "Done"
    End Sub
    Private Sub Button19_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button19.Click
        Dim dados As New SqlDataAdapter("Select * from back", dos)
        Dim ds As New DataSet
        dados.Fill(ds, "back")
        Dim dosA, dosAB, dosid As Integer
        Dim dosFitnessValue As Double
        Dim NoRowsTabledos As Integer
        NoRowsTabledos = ds.Tables("back").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTabledos - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTabledos < FinalPopValue Then
                FinalPopValue = NoRowsTabledos - 1
            End If
```

```
                For j = FirstPopValue To FinalPopValue
                    If ds.Tables("back").Rows(j).Item(6) > maxA Then
                        maxA = ds.Tables("back").Rows(j).Item(6)
                    End If
                    If ds.Tables("back").Rows(j).Item(7) > maxAB Then
                        maxAB = ds.Tables("back").Rows(j).Item(7)
                    End If
                Next
                For y = FirstPopValue To FinalPopValue
                    dosA = ds.Tables("back").Rows(y).Item(6)
                    dosAB = ds.Tables("back").Rows(y).Item(7)
                    dosid = ds.Tables("back").Rows(y).Item(8)
               dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
                    dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
                    Dim Ocmd As New Data.SqlClient.SqlCommand
                    Ocmd.CommandType = CommandType.StoredProcedure
                    Ocmd.Connection = dos
                    Ocmd.Parameters.AddWithValue("@id", dosid)
                    Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
                    Ocmd.CommandText = "fitnessdos1"
                    Try
                        dos.Open()
                        Ocmd.ExecuteNonQuery()
                    Catch ex As Exception
                        MsgBox(ex.Message)
                    End Try
                    dos.Close()
                Next
            Next
            TxtDosDone.Text = "Done"
        End Sub
        Private Sub Button18_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button18.Click
            Dim dados As New SqlDataAdapter("Select * from land", dos)
            Dim ds As New DataSet
            dados.Fill(ds, "land")
            Dim dosA, dosAB, dosid As Integer
            Dim dosFitnessValue As Double
            Dim NoRowsTabledos As Integer
            NoRowsTabledos = ds.Tables("land").Rows.Count
            Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
            For i = 0 To NoRowsTabledos - 1 Step 400
                maxA = 0
                maxAB = 0
                FirstPopValue = i
                FinalPopValue = i + 399
                If NoRowsTabledos < FinalPopValue Then
                    FinalPopValue = NoRowsTabledos - 1
                End If

                For j = FirstPopValue To FinalPopValue
                    If ds.Tables("land").Rows(j).Item(6) > maxA Then
                        maxA = ds.Tables("land").Rows(j).Item(6)
                    End If
                    If ds.Tables("land").Rows(j).Item(7) > maxAB Then
                        maxAB = ds.Tables("land").Rows(j).Item(7)
                    End If
                Next
                For y = FirstPopValue To FinalPopValue
                    dosA = ds.Tables("land").Rows(y).Item(6)
                    dosAB = ds.Tables("land").Rows(y).Item(7)
                    dosid = ds.Tables("land").Rows(y).Item(8)
               dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
                    dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
                    Dim Ocmd As New Data.SqlClient.SqlCommand
                    Ocmd.CommandType = CommandType.StoredProcedure
                    Ocmd.Connection = dos
                    Ocmd.Parameters.AddWithValue("@id", dosid)
                    Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
                    Ocmd.CommandText = "fitnessdos2"
                    Try
                        dos.Open()
                        Ocmd.ExecuteNonQuery()
                    Catch ex As Exception
                        MsgBox(ex.Message)
                    End Try
                    dos.Close()
                Next
            Next
            TxtDosDone.Text = "Done"
        End Sub
        Private Sub Button17_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button17.Click
            Dim dados As New SqlDataAdapter("Select * from neptune", dos)
            Dim ds As New DataSet
            dados.Fill(ds, "neptune")
            Dim dosA, dosAB, dosid As Integer
            Dim dosFitnessValue As Double
            Dim NoRowsTabledos As Integer
            NoRowsTabledos = ds.Tables("neptune").Rows.Count
            Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
            For i = 0 To NoRowsTabledos - 1 Step 400
                maxA = 0
                maxAB = 0
                FirstPopValue = i
                FinalPopValue = i + 399
                If NoRowsTabledos < FinalPopValue Then
                    FinalPopValue = NoRowsTabledos - 1
                End If
                For j = FirstPopValue To FinalPopValue
                    If ds.Tables("neptune").Rows(j).Item(6) > maxA Then
                        maxA = ds.Tables("neptune").Rows(j).Item(6)
                    End If
                    If ds.Tables("neptune").Rows(j).Item(7) > maxAB Then
                        maxAB = ds.Tables("neptune").Rows(j).Item(7)
                    End If
                Next
                For y = FirstPopValue To FinalPopValue
                    dosA = ds.Tables("neptune").Rows(y).Item(6)
                    dosAB = ds.Tables("neptune").Rows(y).Item(7)
                    dosid = ds.Tables("neptune").Rows(y).Item(8)
               dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
                    dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
                    Dim Ocmd As New Data.SqlClient.SqlCommand
                    Ocmd.CommandType = CommandType.StoredProcedure
                    Ocmd.Connection = dos
                    Ocmd.Parameters.AddWithValue("@id", dosid)
                    Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
                    Ocmd.CommandText = "fitnessdos3"
                    Try
                        dos.Open()
                        Ocmd.ExecuteNonQuery()
                    Catch ex As Exception
                        MsgBox(ex.Message)
                    End Try
                    dos.Close()
                Next
            Next
            TxtDosDone.Text = "Done"
        End Sub
        Private Sub Button16_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button16.Click
            Dim dados As New SqlDataAdapter("Select * from pod", dos)
            Dim ds As New DataSet
            dados.Fill(ds, "pod")
            Dim dosA, dosAB, dosid As Integer
            Dim dosFitnessValue As Double
            Dim NoRowsTabledos As Integer
            NoRowsTabledos = ds.Tables("pod").Rows.Count
            Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
            For i = 0 To NoRowsTabledos - 1 Step 400
                maxA = 0
                maxAB = 0
                FirstPopValue = i
                FinalPopValue = i + 399
                If NoRowsTabledos < FinalPopValue Then
```

```
                FinalPopValue = NoRowsTabledos - 1
            End If
            For j = FirstPopValue To FinalPopValue
                If ds.Tables("pod").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("pod").Rows(j).Item(6)
                End If
                If ds.Tables("pod").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("pod").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                dosA = ds.Tables("pod").Rows(y).Item(6)
                dosAB = ds.Tables("pod").Rows(y).Item(7)
                dosid = ds.Tables("pod").Rows(y).Item(8)
           dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
                dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = dos
                Ocmd.Parameters.AddWithValue("@id", dosid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
                Ocmd.CommandText = "fitnessdos4"
                Try
                    dos.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                dos.Close()
            Next
        Next
        TxtDosDone.Text = "Done"
    End Sub
    Private Sub Button15_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button15.Click
        Dim dados As New SqlDataAdapter("Select * from smurf", dos)
        Dim ds As New DataSet
        dados.Fill(ds, "smurf")
        Dim dosA, dosAB, dosid As Integer
        Dim dosFitnessValue As Double
        Dim NoRowsTabledos As Integer
        NoRowsTabledos = ds.Tables("smurf").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTabledos - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTabledos < FinalPopValue Then
                FinalPopValue = NoRowsTabledos - 1
            End If
            For j = FirstPopValue To FinalPopValue
                If ds.Tables("smurf").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("smurf").Rows(j).Item(6)
                End If
                If ds.Tables("smurf").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("smurf").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                dosA = ds.Tables("smurf").Rows(y).Item(6)
                dosAB = ds.Tables("smurf").Rows(y).Item(7)
                dosid = ds.Tables("smurf").Rows(y).Item(8)
           dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
                dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = dos
                Ocmd.Parameters.AddWithValue("@id", dosid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
                Ocmd.CommandText = "fitnessdos5"
                Try
                    dos.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                dos.Close()
            Next
        Next
        TxtDosDone.Text = "Done"
    End Sub
    Private Sub Button6_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button6.Click
        Dim dados As New SqlDataAdapter("Select * from teardrop", dos)
        Dim ds As New DataSet
        dados.Fill(ds, "teardrop")
        Dim dosA, dosAB, dosid As Integer
        Dim dosFitnessValue As Double
        Dim NoRowsTabledos As Integer
        NoRowsTabledos = ds.Tables("teardrop").Rows.Count
        Dim FirstPopValue, FinalPopValue, maxA, maxAB As Integer
        For i = 0 To NoRowsTabledos - 1 Step 400
            maxA = 0
            maxAB = 0
            FirstPopValue = i
            FinalPopValue = i + 399
            If NoRowsTabledos < FinalPopValue Then
                FinalPopValue = NoRowsTabledos - 1
            End If
            For j = FirstPopValue To FinalPopValue
                If ds.Tables("teardrop").Rows(j).Item(6) > maxA Then
                    maxA = ds.Tables("teardrop").Rows(j).Item(6)
                End If
                If ds.Tables("teardrop").Rows(j).Item(7) > maxAB Then
                    maxAB = ds.Tables("teardrop").Rows(j).Item(7)
                End If
            Next
            For y = FirstPopValue To FinalPopValue
                dosA = ds.Tables("teardrop").Rows(y).Item(6)
                dosAB = ds.Tables("teardrop").Rows(y).Item(7)
                dosid = ds.Tables("teardrop").Rows(y).Item(8)
           dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
                dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
                Dim Ocmd As New Data.SqlClient.SqlCommand
                Ocmd.CommandType = CommandType.StoredProcedure
                Ocmd.Connection = dos
                Ocmd.Parameters.AddWithValue("@id", dosid)
                Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
                Ocmd.CommandText = "fitnessdos6"
                Try
                    dos.Open()
                    Ocmd.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                dos.Close()
            Next
        Next
        TxtDosDone.Text = "Done"
    End Sub
```

- Code for using Steady State Genetic Algorithm

```
-   Public Class Form1
-       Private U2R As New SqlConnection("Data Source=MO3ATH-PC;Initial Catalog=S-
    U2R;Integrated Security=True")
-       Private CSTrain As New SqlConnection("Data Source=MO3ATH-PC;Initial
    Catalog=master;Integrated Security=True")
-       Private datrain As New SqlDataAdapter("Select * from kddcup$", CSTrain)
-
-       Dim ds As New DataSet
-       Dim NoRowsTableU2R_initial As Integer
-       Dim NoRowsTableU2R As Integer
-       Dim FirstPopValue, FinalPopValue As Integer
-       Dim GAParameter As String
-       Dim dau2r As New SqlDataAdapter("Select * from rootkit", U2R)
-       Dim steep1, steep2, steep3 As Integer
-       Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
-           dau2r.Fill(ds, "rootkit")
-       End Sub
-       Private Sub Button13_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button13.Click
-           If CmbSelection.SelectedItem = Nothing Then
-               MsgBox("Please Select Selection type")
-               Exit Sub
-           ElseIf CmbCrossover.SelectedItem = Nothing Then
-               MsgBox("Please Select Crossover type")
-               Exit Sub
-           ElseIf CmbReplacment.SelectedItem = Nothing Then
-               MsgBox("Please Select Replacement type")
-               Exit Sub
-           ElseIf TxtPopSize.Text = "" Then
-               MsgBox("Please determine the population size")
-               Exit Sub
-           End If
-           GAParameter = ""
-           If CmbSelection.SelectedIndex = 0 Then
-               GAParameter = GAParameter & "1"
-           ElseIf CmbSelection.SelectedIndex = 1 Then
-               GAParameter = GAParameter & "2"
-           ElseIf CmbSelection.SelectedIndex = 2 Then
-               GAParameter = GAParameter & "3"
-           ElseIf CmbSelection.SelectedIndex = 3 Then
-               GAParameter = GAParameter & "4"
-           ElseIf CmbSelection.SelectedIndex = 4 Then
-               GAParameter = GAParameter & "5"
-           End If
-           ' DataBase Definition
-           Dim NoRowsTableU2R As Integer
-           NoRowsTableU2R = ds.Tables("u2r").Rows.Count
-           ' Population definitions + parameter of population definition
-           Dim PopIndex, PopulationIndex As Integer
-           ' RWS -------------
-           Dim RndNum As Double
-           Dim SelectedIndividual As Integer
-           Dim SumFitness, AverageFitness, ExpectedFitness, SumExpectedFitness,
    SumExpectedFitness1 As Double
-           Dim j As Integer
-           Dim icount As Integer
-           ' Elitest -----------
-           Dim icount1, icount2, icount3 As Integer
-           Dim TempInt As Integer
-           Dim TempDouble As Double
-           ' Ranking -----------
-           Dim AllPopulation_Ranked As Integer(,) = New Integer(NoRowsTableU2R, 1) {}
-           Dim AllPopulation_FitnessRanked As Double(,) = New Double(NoRowsTableU2R, 1) {}
-           Dim min, max As Double
-           Dim y As Double
-           Dim x As Integer
-           ' Tournament -------------
-           Dim RndNum1, RndNum2, Difference, steep As Integer
-           Dim TableNameu2r As String
-           NoRowsTableU2R = ds.Tables("rootkit").Rows.Count
-           TableNameu2r = "rootkit"
-           dau2r.Fill(ds, TableNameu2r)
-           NoRowsTableU2R = ds.Tables(TableNameu2r).Rows.Count
-           Dim Generation As Integer
-           Generation = 1
-           Dim OldGeneration As Double(,) = New Double(NoRowsTableU2R, 11) {}
-           Dim LastGenerationNoRowsTabledos As Integer
-           LastGenerationNoRowsTabledos = 0
-
-           ' ##### Start of GENERATION   ##########################################
-           Do While LastGenerationNoRowsTabledos < NoRowsTableU2R 'And Generation <> 16
-               ds.Clear()
-               datrain.Fill(ds, "kddcup$")
-               dau2r.Fill(ds, TableNameu2r)
-               NoRowsTableU2R = ds.Tables(TableNameu2r).Rows.Count
-               Dim AllPopulation_Old As Double(,) = New Double(NoRowsTableU2R, 1) {}
-               Dim AllPopulation_New As Double(,) = New Double(NoRowsTableU2R, 1) {}
-               Dim CrossedPopulation As Double(,) = New Double(NoRowsTableU2R, 4) {}
-               Dim CurrentGeneration As Double(,) = New Double(NoRowsTableU2R, 11) {}
-               Dim SelectedGeneration As Double(,) = New Double(NoRowsTableU2R, 11) {}
-               For PopulationIndex = 0 To NoRowsTableU2R - 1 Step Val(TxtPopSize.Text)
-                   FirstPopValue = PopulationIndex
-                   FinalPopValue = PopulationIndex + Val(TxtPopSize.Text) - 1
-                   If NoRowsTableU2R < FinalPopValue Then
-                       FinalPopValue = NoRowsTableU2R - 1
-                   End If
-                   If Generation = 1 Then
-                       For PopIndex = 0 To NoRowsTableU2R - 1
-                           OldGeneration(PopIndex, 0) = 0
-                           OldGeneration(PopIndex, 1) = 0
-                           OldGeneration(PopIndex, 2) = 0
-                           OldGeneration(PopIndex, 3) = 0
-                           OldGeneration(PopIndex, 4) = 0
-                           OldGeneration(PopIndex, 5) = 0
-                           OldGeneration(PopIndex, 6) = 0
-                           OldGeneration(PopIndex, 7) = 0
-                           OldGeneration(PopIndex, 8) = 0
-                           OldGeneration(PopIndex, 9) = 0
-                           OldGeneration(PopIndex, 10) = 0
-                           OldGeneration(PopIndex, 11) = 0
-                       Next
-                   End If
-                   ' ##############################################
-                   ' Selection Process
-                   If CmbSelection.SelectedIndex = 0 Then
-                       ' RWS Selection
-                       SumFitness = 0
-                       For icount = FirstPopValue To FinalPopValue
-                       SumFitness = SumFitness + ds.Tables("rootkit").Rows(icount).Item(9)
-                       Next
-                       AverageFitness = SumFitness / (FinalPopValue - FirstPopValue + 1)
-                       SumExpectedFitness = 0
-                       For icount = FirstPopValue To FinalPopValue
-                       SumExpectedFitness = SumExpectedFitness +
    ds.Tables("rootkit").Rows(icount).Item(9) / AverageFitness
-                       Next
-                       For icount = FirstPopValue To FinalPopValue
-                           RndNum = Int((Rnd() * 100)) Mod SumExpectedFitness
-                           SumExpectedFitness1 = 0
-                           j = FirstPopValue
-                           While (j < FinalPopValue)
-                   ExpectedFitness = ds.Tables("rootkit").Rows(j).Item(9) / AverageFitness
-                               SumExpectedFitness1 = SumExpectedFitness1 + ExpectedFitness
-                               SelectedIndividual = ds.Tables("rootkit").Rows(j).Item(8)
-                               If SumExpectedFitness1 > RndNum Then
-                                   Exit While
-                               Else
-                                   j = j + 1
-                               End If
-                           End While
-   AllPopulation_New(icount, 0) = ds.Tables("rootkit").Rows(SelectedIndividual - 1).Item(8)
-   AllPopulation_New(icount, 1) = ds.Tables("rootkit").Rows(SelectedIndividual - 1).Item(9)
-                       Next
-                   ElseIf CmbSelection.SelectedIndex = 1 Then
-                       'Elitist Selection
-                       '********************************************************
-                       For icount1 = FirstPopValue To FinalPopValue
-   AllPopulation_New(icount1, 0) = ds.Tables("rootkit").Rows(icount1).Item(8)
-   AllPopulation_New(icount1, 1) = ds.Tables("rootkit").Rows(icount1).Item(9)
-                       Next
-                       For icount2 = FirstPopValue To FinalPopValue
-                           For icount3 = FirstPopValue To FinalPopValue - 1
-   If AllPopulation_New(icount3 + 1, 1) > AllPopulation_New(icount3, 1) Then
-                                   TempInt = AllPopulation_New(icount3 + 1, 0)
-                                   TempDouble = AllPopulation_New(icount3 + 1, 1)
-   AllPopulation_New(icount3 + 1, 0) = AllPopulation_New(icount3, 0)
-   AllPopulation_New(icount3 + 1, 1) = AllPopulation_New(icount3, 1)
-                               AllPopulation_New(icount3, 0) = TempInt
-                               AllPopulation_New(icount3, 1) = TempDouble
-                           End If
-                       Next
-                   Next
-                   '*********************************************************
-               ElseIf CmbSelection.SelectedIndex = 2 Then
-                   ' Ranking Selection
-                   For icount1 = FirstPopValue To FinalPopValue
-   AllPopulation_New(icount1, 0) = ds.Tables("rootkit").Rows(icount1).Item(8)
-   AllPopulation_New(icount1, 1) = ds.Tables("rootkit").Rows(icount1).Item(9)
-                   Next
-                   For icount2 = FirstPopValue To FinalPopValue
-                       For icount3 = FirstPopValue To FinalPopValue - 1
-   If AllPopulation_New(icount3 + 1, 1) > AllPopulation_New(icount3, 1) Then
-                               TempInt = AllPopulation_New(icount3 + 1, 0)
-                               TempDouble = AllPopulation_New(icount3 + 1, 1)
-   AllPopulation_New(icount3 + 1, 0) = AllPopulation_New(icount3, 0)
-   AllPopulation_New(icount3 + 1, 1) = AllPopulation_New(icount3, 1)
-                               AllPopulation_New(icount3, 0) = TempInt
-                               AllPopulation_New(icount3, 1) = TempDouble
-                           End If
-                       Next
-                   Next
-                   For PopIndex = FirstPopValue To FinalPopValue
-                     AllPopulation_Ranked(PopIndex, 0) = AllPopulation_New(PopIndex, 0)
-                       AllPopulation_Ranked(PopIndex, 1) = FinalPopValue - PopIndex + 1
-                   Next
-                   For PopIndex = FirstPopValue To FinalPopValue
-                       x = Rnd() * 100
-                       y = x / 100 + 1
-                       max = y
-                       If max = 1 Then
-                           max = 1.1
-                       End If
-                       min = 2 - max
-            AllPopulation_FitnessRanked(PopIndex, 0) = AllPopulation_Ranked(PopIndex, 0)
-             AllPopulation_FitnessRanked(PopIndex, 1) = max - (max - min) *
    ((AllPopulation_Ranked(PopIndex, 1) - 1) / (Val(TxtPopSize.Text) - 1))
-                   Next
-                   SumFitness = 0
-                   For icount = FirstPopValue To FinalPopValue
-                       SumFitness = SumFitness + AllPopulation_FitnessRanked(icount, 1)
-                   Next
-                   AverageFitness = SumFitness / (FinalPopValue - FirstPopValue + 1)
-                   SumExpectedFitness = 0
-                   For icount = FirstPopValue To FinalPopValue
-   SumExpectedFitness = SumExpectedFitness + ds.Tables("rootkit").Rows(icount).Item(9) /
    AverageFitness
-                   Next
-                   For icount = FirstPopValue To FinalPopValue
-                       RndNum = Int((Rnd() * 100)) Mod SumExpectedFitness
-                       SumExpectedFitness1 = 0
-                       j = FirstPopValue
-                       While (j < FinalPopValue)
-               ExpectedFitness = ds.Tables("rootkit").Rows(j).Item(9) / AverageFitness
-                           SumExpectedFitness1 = SumExpectedFitness1 + ExpectedFitness
-                           SelectedIndividual = AllPopulation_FitnessRanked(j, 0)
-                           If SumExpectedFitness1 > RndNum Then
-                               Exit While
-                           Else
-                               j = j + 1
-                           End If
-                       End While
-                       AllPopulation_New(icount, 0) = AllPopulation_FitnessRanked(j, 0)
-                       AllPopulation_New(icount, 1) = AllPopulation_FitnessRanked(j, 1)
-                   Next
-                   ' ***************************************************
-               ElseIf CmbSelection.SelectedIndex = 3 Then
-                   ' SUS
-                       SumFitness = 0
-                       For icount = FirstPopValue To FinalPopValue
-               SumFitness = SumFitness + ds.Tables("rootkit").Rows(icount).Item(9)
-                       Next
-                       AverageFitness = SumFitness / (FinalPopValue - FirstPopValue + 1)
-                       For icount = FirstPopValue To FinalPopValue
-                           RndNum = icount Mod Val(TxtPopSize.Text)
-                           SumExpectedFitness = 0
-                           j = FirstPopValue
-                           While (j < FinalPopValue)
-                   ExpectedFitness = ds.Tables("rootkit").Rows(j).Item(9) / AverageFitness
-                               SumExpectedFitness = SumExpectedFitness + ExpectedFitness
-                               SelectedIndividual = ds.Tables("rootkit").Rows(j).Item(8)
-                               If SumExpectedFitness > RndNum Then
-                                   Exit While
-                               Else
-                                   j = j + 1
-                               End If
-                           End While
-                     AllPopulation_New(icount, 0) = ds.Tables("rootkit").Rows(j).Item(8)
-                     AllPopulation_New(icount, 1) = ds.Tables("rootkit").Rows(j).Item(9)
-                       Next
-                       ' ********************************************
-                   ElseIf CmbSelection.SelectedIndex = 4 Then
-                       ' Tournament Selection
-                       Difference = FinalPopValue - FirstPopValue
-                       Difference = Difference + 1
-                       For icount = FirstPopValue To FinalPopValue
-                           Do
-                           RndNum1 = Int((Rnd() * 100)) Mod Difference + FirstPopValue + 1
-                           RndNum2 = Int((Rnd() * 100)) Mod Difference + FirstPopValue + 1
-                           Loop While (RndNum1 = RndNum2 And Difference - 1 > 0)
-                           If ds.Tables("rootkit").Rows(RndNum1 - 1).Item(9) >
    ds.Tables("rootkit").Rows(RndNum2 - 1).Item(9) Then
-             AllPopulation_New(icount, 0) = ds.Tables("rootkit").Rows(RndNum1 - 1).Item(8)
-             AllPopulation_New(icount, 1) = ds.Tables("rootkit").Rows(RndNum1 - 1).Item(9)
-                           Else
-             AllPopulation_New(icount, 0) = ds.Tables("rootkit").Rows(RndNum2 - 1).Item(8)
-             AllPopulation_New(icount, 1) = ds.Tables("rootkit").Rows(RndNum2 - 1).Item(9)
-                           End If
-                       Next
-                   End If
-               Next
-               ' ##############################################
-               ' Crossover Process
-               Dim Cross11, Cross21, Cross12, Cross22, Cross13, Cross23, Cross14, Cross24,
    Cross15, Cross25, Change As Double
-               If NoRowsTableU2R Mod 2 <> 0 Then
-                   NoRowsTableU2R = NoRowsTableU2R - 1
-               End If
-               If CmbCrossover.SelectedIndex = 0 Then
-                   GAParameter = GAParameter & "1"
-                   For steep = 0 To NoRowsTableU2R - 1 Step 2
-                       RndNum1 = Int((Rnd() * 100)) Mod Val(TxtPopSize.Text)
-                       Cross11 = ds.Tables("rootkit").Rows(steep).Item(1)
-                       Cross21 = ds.Tables("rootkit").Rows(steep + 1).Item(1)
-                       If RndNum1 > 1 Then
-                           Cross12 = ds.Tables("rootkit").Rows(steep).Item(2)
-                           Cross22 = ds.Tables("rootkit").Rows(steep + 1).Item(2)
-                       Else
-                           Cross12 = ds.Tables("rootkit").Rows(steep + 1).Item(2)
-                           Cross22 = ds.Tables("rootkit").Rows(steep).Item(2)
-                       End If
-                       If RndNum1 > 2 Then
-                           Cross13 = ds.Tables("rootkit").Rows(steep).Item(3)
-                           Cross23 = ds.Tables("rootkit").Rows(steep + 1).Item(3)
-                       Else
-                           Cross13 = ds.Tables("rootkit").Rows(steep + 1).Item(3)
-                           Cross23 = ds.Tables("rootkit").Rows(steep).Item(3)
-                       End If
-                       If RndNum1 > 3 Then
-                           Cross14 = ds.Tables("rootkit").Rows(steep).Item(4)
-                       Cross24 = ds.Tables("rootkit").Rows(steep + 1).Item(4)
-                   Else
-                       Cross14 = ds.Tables("rootkit").Rows(steep + 1).Item(4)
-                       Cross24 = ds.Tables("rootkit").Rows(steep).Item(4)
-                   End If
-                   If RndNum1 <= 4 Then
-                       Cross15 = ds.Tables("rootkit").Rows(steep + 1).Item(5)
-                       Cross25 = ds.Tables("rootkit").Rows(steep).Item(5)
-                   Else
-                       Cross15 = ds.Tables("rootkit").Rows(steep).Item(5)
-                       Cross25 = ds.Tables("rootkit").Rows(steep + 1).Item(5)
-                   End If
-                   CrossedPopulation(steep, 0) = Cross11
-                   CrossedPopulation(steep, 1) = Cross12
-                   CrossedPopulation(steep, 2) = Cross13
-                   CrossedPopulation(steep, 3) = Cross14
-                   CrossedPopulation(steep, 4) = Cross15
-                   CrossedPopulation(steep + 1, 0) = Cross21
-                   CrossedPopulation(steep + 1, 1) = Cross22
-                   CrossedPopulation(steep + 1, 2) = Cross23
-                   CrossedPopulation(steep + 1, 3) = Cross24
-                   CrossedPopulation(steep + 1, 4) = Cross25
-               Next
-           ElseIf CmbCrossover.SelectedIndex = 1 Then
-               GAParameter = GAParameter & "2"
-               For steep = 0 To NoRowsTableU2R - 1 Step 2
-                   RndNum1 = Int((Rnd() * 100)) Mod 2 + 1
-                   Cross11 = ds.Tables("rootkit").Rows(steep).Item(1)
-                   Cross21 = ds.Tables("rootkit").Rows(steep + 1).Item(1)
-                   If RndNum1 = 1 Then
-                       Cross12 = ds.Tables("rootkit").Rows(steep + 1).Item(2)
-                       Cross22 = ds.Tables("rootkit").Rows(steep).Item(2)
-
-                       Cross13 = ds.Tables("rootkit").Rows(steep + 1).Item(3)
-                       Cross23 = ds.Tables("rootkit").Rows(steep).Item(3)
-
-                       Cross14 = ds.Tables("rootkit").Rows(steep).Item(4)
-                       Cross24 = ds.Tables("rootkit").Rows(steep + 1).Item(4)
-
-                       Cross15 = ds.Tables("rootkit").Rows(steep).Item(5)
-                       Cross25 = ds.Tables("rootkit").Rows(steep + 1).Item(5)
-                   ElseIf RndNum1 = 2 Then
-                       Cross12 = ds.Tables("rootkit").Rows(steep).Item(2)
-                       Cross22 = ds.Tables("rootkit").Rows(steep + 1).Item(2)
-
-                       Cross13 = ds.Tables("rootkit").Rows(steep + 1).Item(3)
-                       Cross23 = ds.Tables("rootkit").Rows(steep).Item(3)
-
-                       Cross14 = ds.Tables("rootkit").Rows(steep + 1).Item(4)
-                       Cross24 = ds.Tables("rootkit").Rows(steep).Item(4)
-
-                       Cross15 = ds.Tables("rootkit").Rows(steep).Item(5)
-                       Cross25 = ds.Tables("rootkit").Rows(steep + 1).Item(5)
-                   End If
-                   CrossedPopulation(steep, 0) = Cross11
-                   CrossedPopulation(steep, 1) = Cross12
-                   CrossedPopulation(steep, 2) = Cross13
-                   CrossedPopulation(steep, 3) = Cross14
-                   CrossedPopulation(steep, 4) = Cross15
-                   CrossedPopulation(steep + 1, 0) = Cross21
-                   CrossedPopulation(steep + 1, 1) = Cross22
-                   CrossedPopulation(steep + 1, 2) = Cross23
-                   CrossedPopulation(steep + 1, 3) = Cross24
-                   CrossedPopulation(steep + 1, 4) = Cross25
-               Next
-           ElseIf CmbCrossover.SelectedIndex = 2 Then
-               GAParameter = GAParameter & "3"
-               For steep = 0 To NoRowsTableU2R - 1 Step 2
-                   RndNum1 = Int((Rnd() * 100)) Mod 5 + 1
-                   RndNum2 = Int((Rnd() * 100)) Mod 5 + 1
-                   If RndNum1 = 1 Then
-                       Cross11 = ds.Tables("rootkit").Rows(steep + 1).Item(1)
-                           Cross21 = ds.Tables("rootkit").Rows(steep).Item(1)
-                       Else
-                           Cross11 = ds.Tables("rootkit").Rows(steep).Item(1)
-                           Cross21 = ds.Tables("rootkit").Rows(steep + 1).Item(1)
-                       End If
-                       If RndNum1 = 2 Then
-                           Cross12 = ds.Tables("rootkit").Rows(steep + 1).Item(2)
-                           Cross22 = ds.Tables("rootkit").Rows(steep).Item(2)
-                       Else
-                           Cross12 = ds.Tables("rootkit").Rows(steep).Item(2)
-                           Cross22 = ds.Tables("rootkit").Rows(steep + 1).Item(2)
-                       End If
-                       If RndNum1 = 3 Then
-                           Cross13 = ds.Tables("rootkit").Rows(steep + 1).Item(3)
-                           Cross23 = ds.Tables("rootkit").Rows(steep).Item(3)
-                       Else
-                           Cross13 = ds.Tables("rootkit").Rows(steep).Item(3)
-                           Cross23 = ds.Tables("rootkit").Rows(steep + 1).Item(3)
-                       End If
-                       If RndNum1 = 4 Then
-                           Cross14 = ds.Tables("rootkit").Rows(steep + 1).Item(4)
-                           Cross24 = ds.Tables("rootkit").Rows(steep).Item(4)
-                       Else
-                           Cross14 = ds.Tables("rootkit").Rows(steep).Item(4)
-                           Cross24 = ds.Tables("rootkit").Rows(steep + 1).Item(4)
-                       End If
-                       If RndNum1 = 5 Then
-                           Cross15 = ds.Tables("rootkit").Rows(steep + 1).Item(5)
-                           Cross25 = ds.Tables("rootkit").Rows(steep).Item(5)
-                       Else
-                           Cross15 = ds.Tables("rootkit").Rows(steep).Item(5)
-                           Cross25 = ds.Tables("rootkit").Rows(steep + 1).Item(5)
-                       End If
-                       If RndNum2 = 1 Then
-                           Change = Cross11
-                           Cross11 = Cross21
-                           Cross21 = Change
-                       End If
-                       If RndNum2 = 2 Then
-                           Change = Cross12
-                           Cross12 = Cross22
-                           Cross22 = Change
-                       End If
-                       If RndNum2 = 3 Then
-                           Change = Cross13
-                           Cross13 = Cross23
-                           Cross23 = Change
-                       End If
-                       If RndNum2 = 4 Then
-                           Change = Cross14
-                           Cross14 = Cross24
-                           Cross24 = Change
-                       End If
-                       If RndNum2 = 5 Then
-                           Change = Cross15
-                           Cross15 = Cross25
-                           Cross25 = Change
-                       End If
-                       CrossedPopulation(steep, 0) = Cross11
-                       CrossedPopulation(steep, 1) = Cross12
-                       CrossedPopulation(steep, 2) = Cross13
-                       CrossedPopulation(steep, 3) = Cross14
-                       CrossedPopulation(steep, 4) = Cross15
-                       CrossedPopulation(steep + 1, 0) = Cross21
-                       CrossedPopulation(steep + 1, 1) = Cross22
-                       CrossedPopulation(steep + 1, 2) = Cross23
-                       CrossedPopulation(steep + 1, 3) = Cross24
-                       CrossedPopulation(steep + 1, 4) = Cross25
-                   Next
-               End If
-               ' ##############################################
-               ' Mutation Process
-               For steep = 0 To NoRowsTableU2R - 1 Step 5
-                   RndNum = Int((Rnd() * 100)) Mod 5 + 1
-                   If RndNum = 1 Then
-                       If CrossedPopulation(steep, 0) = 0 Then
-                           CrossedPopulation(steep, 0) = 1
-                       Else
-                           CrossedPopulation(steep, 0) = 0
-                       End If
-                   End If
-                   If RndNum = 2 Then
-                       CrossedPopulation(steep, 1) = Int((Rnd() * 100)) Mod 4 + 1
-                   End If
-                   If RndNum = 3 Then
-                       If CrossedPopulation(steep, 2) = 0 Then
-                           CrossedPopulation(steep, 2) = 1
-                       Else
-                           CrossedPopulation(steep, 2) = 0
-                       End If
-                   End If
-                   If RndNum = 4 Then
-                       If CrossedPopulation(steep, 3) = 0 Then
-                           CrossedPopulation(steep, 3) = 1
-                       Else
-                           CrossedPopulation(steep, 3) = 0
-                       End If
-                   End If
-                   If RndNum = 5 Then
-                       CrossedPopulation(steep, 4) = Int(Rnd() * 100) / 100
-                   End If
-               Next
-               ' ##############################################
-               ' Evaluation
-               Dim NoRowsTablekddcup As Integer
-               Dim kddcounter As Integer
-               Dim dAB, dA As Integer
-               Dim maxAB, maxA As Double
-               Dim d7, d8, d12, d13, d23 As Double
-               Dim Attack As Integer
-               NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-               maxA = 0
-               maxAB = 0
-               For j = 0 To NoRowsTableU2R - 1
-                   If ds.Tables(TableNameu2r).Rows(j).Item(6) > maxA Then
-                       maxA = ds.Tables(TableNameu2r).Rows(j).Item(6)
-                   End If
-                   If ds.Tables(TableNameu2r).Rows(j).Item(7) > maxAB Then
-                       maxAB = ds.Tables(TableNameu2r).Rows(j).Item(7)
-                   End If
-               Next
-               Dim FitnessValue As Double
-               For Attack = 0 To NoRowsTableU2R - 1
-                   dAB = 0
-                   dA = 0
-                   d7 = CrossedPopulation(Attack, 0)
-                   d8 = CrossedPopulation(Attack, 1)
-                   d12 = CrossedPopulation(Attack, 2)
-                   d13 = CrossedPopulation(Attack, 3)
-                   d23 = CrossedPopulation(Attack, 4)
-                   For kddcounter = 0 To NoRowsTablekddcup - 1
-                       If d7 = ds.Tables("kddcup$").Rows(kddcounter).Item(6) Then
-                           If d8 = ds.Tables("kddcup$").Rows(kddcounter).Item(7) Then
-                               If d12 = ds.Tables("kddcup$").Rows(kddcounter).Item(11) Then
-                               If d13 = ds.Tables("kddcup$").Rows(kddcounter).Item(12) Then
-                               If d23 = ds.Tables("kddcup$").Rows(kddcounter).Item(22) Then
-                        If ds.Tables("kddcup$").Rows(kddcounter).Item(41) = "rootkit." Then
-                                               dAB = dAB + 1
-                                           Else
-                                               dA = dA + 1
-                                           End If
-                                       End If
-                                   End If
-                               End If
-                           End If
-                       End If
-                   Next
-                   If dAB = 0 And dA = 0 Then
-                       FitnessValue = Val(Rnd() + 2.5)
-                   Else
-                   FitnessValue = 2 + ((dAB - dA) / (dA + dAB)) + dAB / maxAB - dA / maxA
-                   End If
-                   CurrentGeneration(Attack, 0) = 0
-                   CurrentGeneration(Attack, 1) = CrossedPopulation(Attack, 0)
-                   CurrentGeneration(Attack, 2) = CrossedPopulation(Attack, 1)
-                   CurrentGeneration(Attack, 3) = CrossedPopulation(Attack, 2)
-                   CurrentGeneration(Attack, 4) = CrossedPopulation(Attack, 3)
-                   CurrentGeneration(Attack, 5) = CrossedPopulation(Attack, 4)
-                   CurrentGeneration(Attack, 6) = dA
-                   CurrentGeneration(Attack, 7) = dAB
-                   CurrentGeneration(Attack, 8) = NoRowsTableU2R + Attack
-                   CurrentGeneration(Attack, 9) = FitnessValue
-                   CurrentGeneration(Attack, 10) = Generation
-                   CurrentGeneration(Attack, 11) = Val(GAParameter)
-               Next
-               ' ##############################################
-               ' Replacement Process
-               If CmbReplacment.SelectedIndex = 0 Then
-                   Dim ComparedIndex As Integer
-                   Dim RemindIndex As Integer
-                   If CurrentGeneration.Length < OldGeneration.Length Then
-                       ComparedIndex = CurrentGeneration.Length
-                   RemindIndex = Math.Abs(OldGeneration.Length - CurrentGeneration.Length)
-                   Else
-                       ComparedIndex = OldGeneration.Length
-                   RemindIndex = Math.Abs(CurrentGeneration.Length - OldGeneration.Length)
-                   End If
-                   For PopIndex = 0 To OldGeneration.GetUpperBound(0) - 1
-                       If Generation = 1 Then
-                           SelectedGeneration(PopIndex, 0) = CurrentGeneration(PopIndex, 0)
-                           SelectedGeneration(PopIndex, 1) = CurrentGeneration(PopIndex, 1)
-                           SelectedGeneration(PopIndex, 2) = CurrentGeneration(PopIndex, 2)
-                           SelectedGeneration(PopIndex, 3) = CurrentGeneration(PopIndex, 3)
-                           SelectedGeneration(PopIndex, 4) = CurrentGeneration(PopIndex, 4)
-                           SelectedGeneration(PopIndex, 5) = CurrentGeneration(PopIndex, 5)
-                           SelectedGeneration(PopIndex, 6) = CurrentGeneration(PopIndex, 6)
-                           SelectedGeneration(PopIndex, 7) = CurrentGeneration(PopIndex, 7)
-                           SelectedGeneration(PopIndex, 8) = CurrentGeneration(PopIndex, 8)
-                           SelectedGeneration(PopIndex, 9) = CurrentGeneration(PopIndex, 9)
-                         SelectedGeneration(PopIndex, 10) = CurrentGeneration(PopIndex, 10)
-                         SelectedGeneration(PopIndex, 11) = CurrentGeneration(PopIndex, 11)
-                       Else
-                        If CurrentGeneration(PopIndex, 9) > OldGeneration(PopIndex, 9) Then
-                          SelectedGeneration(PopIndex, 0) = CurrentGeneration(PopIndex, 0)
-                          SelectedGeneration(PopIndex, 1) = CurrentGeneration(PopIndex, 1)
-                          SelectedGeneration(PopIndex, 2) = CurrentGeneration(PopIndex, 2)
-                          SelectedGeneration(PopIndex, 3) = CurrentGeneration(PopIndex, 3)
-                          SelectedGeneration(PopIndex, 4) = CurrentGeneration(PopIndex, 4)
-                          SelectedGeneration(PopIndex, 5) = CurrentGeneration(PopIndex, 5)
-                          SelectedGeneration(PopIndex, 6) = CurrentGeneration(PopIndex, 6)
-                          SelectedGeneration(PopIndex, 7) = CurrentGeneration(PopIndex, 7)
-                          SelectedGeneration(PopIndex, 8) = CurrentGeneration(PopIndex, 8)
-                          SelectedGeneration(PopIndex, 9) = CurrentGeneration(PopIndex, 9)
-                         SelectedGeneration(PopIndex, 10) = CurrentGeneration(PopIndex, 10)
-                         SelectedGeneration(PopIndex, 11) = CurrentGeneration(PopIndex, 11)
-                           Else
-                               SelectedGeneration(PopIndex, 0) = OldGeneration(PopIndex, 0)
-                               SelectedGeneration(PopIndex, 1) = OldGeneration(PopIndex, 1)
-                               SelectedGeneration(PopIndex, 2) = OldGeneration(PopIndex, 2)
-                               SelectedGeneration(PopIndex, 3) = OldGeneration(PopIndex, 3)
-                               SelectedGeneration(PopIndex, 4) = OldGeneration(PopIndex, 4)
-                               SelectedGeneration(PopIndex, 5) = OldGeneration(PopIndex, 5)
-                               SelectedGeneration(PopIndex, 6) = OldGeneration(PopIndex, 6)
-                               SelectedGeneration(PopIndex, 7) = OldGeneration(PopIndex, 7)
-                               SelectedGeneration(PopIndex, 8) = OldGeneration(PopIndex, 8)
-                               SelectedGeneration(PopIndex, 9) = OldGeneration(PopIndex, 9)
-                             SelectedGeneration(PopIndex, 10) = OldGeneration(PopIndex, 10)
-                             SelectedGeneration(PopIndex, 11) = OldGeneration(PopIndex, 11)
-                            End If
-                        End If
-                    Next
-    For PopIndex = OldGeneration.GetUpperBound(0) To CurrentGeneration.GetUpperBound(0) - 1
-                        SelectedGeneration(PopIndex, 0) = CurrentGeneration(PopIndex, 0)
-                        SelectedGeneration(PopIndex, 1) = CurrentGeneration(PopIndex, 1)
-                        SelectedGeneration(PopIndex, 2) = CurrentGeneration(PopIndex, 2)
-                        SelectedGeneration(PopIndex, 3) = CurrentGeneration(PopIndex, 3)
-                        SelectedGeneration(PopIndex, 4) = CurrentGeneration(PopIndex, 4)
-                        SelectedGeneration(PopIndex, 5) = CurrentGeneration(PopIndex, 5)
-                        SelectedGeneration(PopIndex, 6) = CurrentGeneration(PopIndex, 6)
-                        SelectedGeneration(PopIndex, 7) = CurrentGeneration(PopIndex, 7)
-                        SelectedGeneration(PopIndex, 8) = CurrentGeneration(PopIndex, 8)
-                        SelectedGeneration(PopIndex, 9) = CurrentGeneration(PopIndex, 9)
-                        SelectedGeneration(PopIndex, 10) = CurrentGeneration(PopIndex, 10)
-                        SelectedGeneration(PopIndex, 11) = CurrentGeneration(PopIndex, 11)
-                    Next
-                End If
-                LastGenerationNoRowsTabledos = NoRowsTableU2R
-                ' Saving Data in the Table
-                Dim CheckCounter As Integer
-                Dim Redundant As Integer
-                For PopIndex = 0 To NoRowsTableU2R - 1
-                    d7 = SelectedGeneration(PopIndex, 1)
-                    d8 = SelectedGeneration(PopIndex, 2)
-                    d12 = SelectedGeneration(PopIndex, 3)
-                    d13 = SelectedGeneration(PopIndex, 4)
-                    d23 = SelectedGeneration(PopIndex, 5)
-                    Redundant = 0
-                    For CheckCounter = 0 To NoRowsTableU2R - 1
-                        If d7 = ds.Tables(TableNameu2r).Rows(CheckCounter).Item(1) And d8 =
   ds.Tables(TableNameu2r).Rows(CheckCounter).Item(2) And d12 =
   ds.Tables(TableNameu2r).Rows(CheckCounter).Item(3) And d13 =
   ds.Tables(TableNameu2r).Rows(CheckCounter).Item(4) And d23 =
   ds.Tables(TableNameu2r).Rows(CheckCounter).Item(5) Then
-                            Redundant = Redundant + 1
-                            Exit For
-                        End If
-                    Next
-                    Dim Ocmd As New SqlCommand
-                    If Redundant = 0 Then
-                        Ocmd.CommandType = CommandType.StoredProcedure
-                        Ocmd.Connection = U2R
-                  Ocmd.Parameters.AddWithValue("@tableid", SelectedGeneration(PopIndex, 0))
-                        Ocmd.Parameters.AddWithValue("@f7", SelectedGeneration(PopIndex, 1))
-                        Ocmd.Parameters.AddWithValue("@f8", SelectedGeneration(PopIndex, 2))
-                      Ocmd.Parameters.AddWithValue("@f12", SelectedGeneration(PopIndex, 3))
-                      Ocmd.Parameters.AddWithValue("@f13", SelectedGeneration(PopIndex, 4))
-                      Ocmd.Parameters.AddWithValue("@f23", SelectedGeneration(PopIndex, 5))
-                        Ocmd.Parameters.AddWithValue("@A", SelectedGeneration(PopIndex, 6))
-                        Ocmd.Parameters.AddWithValue("@AB", SelectedGeneration(PopIndex, 7))
-             Ocmd.Parameters.AddWithValue("@FitnessValue", SelectedGeneration(PopIndex, 9))
-             Ocmd.Parameters.AddWithValue("@generation", SelectedGeneration(PopIndex, 10))
-             Ocmd.Parameters.AddWithValue("@GAparameter", SelectedGeneration(PopIndex, 11))
-                        Ocmd.CommandText = "u2r00insert"
-                        Try
-                            U2R.Open()
-                            Ocmd.ExecuteNonQuery()
-                        Catch ex As Exception
-                            MsgBox(ex.Message)
-                        End Try
-                        U2R.Close()
-                    End If
-                Next
-                ' *************************************************
-                ' afeter replacement exchange AllPop1 with AllPop2
-                OldGeneration = New Double(NoRowsTableU2R, 11) {}
-                For PopIndex = 0 To NoRowsTableU2R - 1
-                    OldGeneration(PopIndex, 0) = CurrentGeneration(PopIndex, 0)
-                    OldGeneration(PopIndex, 1) = CurrentGeneration(PopIndex, 1)
-                   OldGeneration(PopIndex, 2) = CurrentGeneration(PopIndex, 2)
-                   OldGeneration(PopIndex, 3) = CurrentGeneration(PopIndex, 3)
-                   OldGeneration(PopIndex, 4) = CurrentGeneration(PopIndex, 4)
-                   OldGeneration(PopIndex, 5) = CurrentGeneration(PopIndex, 5)
-                   OldGeneration(PopIndex, 6) = CurrentGeneration(PopIndex, 6)
-                   OldGeneration(PopIndex, 7) = CurrentGeneration(PopIndex, 7)
-                   OldGeneration(PopIndex, 8) = CurrentGeneration(PopIndex, 8)
-                   OldGeneration(PopIndex, 9) = CurrentGeneration(PopIndex, 9)
-                   OldGeneration(PopIndex, 10) = CurrentGeneration(PopIndex, 10)
-                   OldGeneration(PopIndex, 11) = CurrentGeneration(PopIndex, 11)
-               Next
-               Generation = Generation + 1
-               ds.Clear()
-               dau2r.Fill(ds, TableNameu2r)
-               NoRowsTableU2R = ds.Tables(TableNameu2r).Rows.Count
-               ' ##### END of GENERATION
    ###############################################################
-           Loop
-           MsgBox("End")
-       End Sub
-   End Class
```

- Code for Testing for R2L-phf

```
Dim U2RCounter, TestCounter As Integer
        NoRowsTabler2l = ds.Tables(r2lname).Rows.Count
        NoRowsTableTest = ds.Tables("KDDtest$").Rows.Count
        For TestCounter = 0 To NoRowsTableTest - 1
            r6 = ds.Tables("KDDtest$").Rows(TestCounter).Item(6)
            r11 = ds.Tables("KDDtest$").Rows(TestCounter).Item(11)
            r12 = ds.Tables("KDDtest$").Rows(TestCounter).Item(12)
            r19 = ds.Tables("KDDtest$").Rows(TestCounter).Item(19)
            r22 = ds.Tables("KDDtest$").Rows(TestCounter).Item(22)

             For U2RCounter = 0 To NoRowsTabler2l - 1
                If ds.Tables(r2lname).Rows(U2RCounter).Item(1) = r6 Then
                    If ds.Tables(r2lname).Rows(U2RCounter).Item(2) = r11 Then
                        If ds.Tables(r2lname).Rows(U2RCounter).Item(3) = r12 Then
                       If ds.Tables(r2lname).Rows(U2RCounter).Item(4) = r19 Then
                           If ds.Tables(r2lname).Rows(U2RCounter).Item(5) = r22 Then
                               If ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "phf." Then
                                        match.Text = Val(match.Text) + 1
                        ElseIf ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "normal." Then
                                        Txtnormal.Text = Val(Txtnormal.Text) + 1
                                    Else
                                        mismatch.Text = Val(mismatch.Text) + 1
                                    End If
                                    Exit For
                                End If
                            End If
                        End If
                    End If
                End If
            Next
        Next
```