



An Enhancement of the Replacement Steady State Genetic Algorithm for Intrusion Detection

تحسين ابدال الخوارزمية الجينية لحالة الاستقرار لكشف التطفل

By:

ShathaAbd-Alhafid Aziz

Supervisor:

Prof. Reyadh Shaker Naoum

**A Thesis Submitted In Partial Fulfillment of the Requirements of the
Master Degree in Computer Information Systems
Faculty of Information Technology
Middle East University**

(May,2014)

Middle East University

Authorization Statement

I am, Shatha Abd-Alhafid Aziz, authorize Middle East University to supply hard 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: Shatha Abd-Alhafid Aziz

Date: 26 May 2014

Signature: 

جامعة الشرق الاوسط

التفويض

أنا شذى عبد الحافظ عزيز أفوض جامعة الشرق الأوسط بتزويد نسخ من رسالتي ورقيا وألكترونيا للمكتبات، أو المنظمات، أو الهيئات والمؤسسات المعنية بالأبحاث والدراسات العلمية عند طلبها.

الاسم: شذى عبد الحافظ عزيز

التاريخ: 26 أيار 2014

التوقيع: 

Middle East University
Examination Committee Decision

This is to certify that the thesis entitled "An Enhancement of the Replacement Steady State Genetic Algorithm for Intrusion Detection" was successfully defined and approved on 26 May 2014.

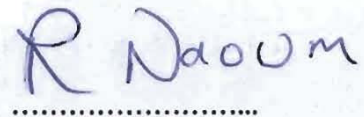
Examination Committee Members

Signature

Prof. Reyadh Shaker Naoum (Supervisor & Chairman)

Professor, Faculty of Information Technology

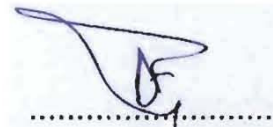
Middle East University (MEU)


.....

Dr. Ahmad Al Kayed (Member)

Dean of Faculty of Information Technology

Middle East University (MEU)


.....

Dr. Abdulsalam W. Alarabeyyat (External Member)

Vice Dean, Faculty of Prince Abdullah Ben Ghazi

For Information Technology

Al -Balqa Applied University (BAU)


.....

Dedication

To

My parents

My husband

My brothers and sisters

My friends

For their love, support and encouragement, without them nothing of this would have been possible. Thank you for everything.

Acknowledgments

At the end of this work, many people deserve to be acknowledged for their contribution to complete this work.

I would like to thank my parents; they always helped me in all my life, I am very thankful for my husband Wisam for his encouragement and support. I thank my children Abdullah, Rawan and fahed, because even if they don't know anything of what I'm doing, they tried to help me, I love you!

A very special thank to my advisor Prof. Reyadh Naoum for his helpful advices, motivation and knowledge he provided me throughout this work. He continually and convincingly pushes this research to the limit. Without his guidance and persistent help, this research would not be accomplished.

Finally, I would like to thank the Information Technology Faculty members and the management of Middle East University for providing adequate environment and supporting their students.

Table of Contents

Authorization Statement	II
التفويض	III
Examination Committee Decision	IV
Dedication	V
Acknowledgements	VI
Table of Contents	VII
List of tables	X
List of figures	XII
Abbreviations	XIII
Abstract	XIV
الملخص	XV
Chapter One: Introduction	
1.1 Preface	1
1.2 Problem Identification	2
1.3 Questions	2
1.4 Objectives	2
1.5 Significance	3
1.6 Limitations	3
1.7 Thesis Outline	4
Chapter Two: Literature Review & Related Work	
2.1 Overview	5
2.2 Literature Review related to Intrusion Detection Systems	5
2.3 Literature Review related to Genetic Algorithm	9
2.4 Literature Review related to Detection Systems based on GA	10
Chapter Three: Theoretical Background: Intrusion Detection & Genetic Algorithm	
3.1 Overview	14

3.2 Intrusion Detection	14
3.2.1 Computer Security	15
3.2.2 Firewall	16
3.2.3 Intrusion Detection Systems (IDS)	17
3.2.4 Components of intrusion detection system	18
3.2.5 Intrusion Detection Classification	19
3.2.6 Intrusion Detection Systems Taxonomy	20
3.2.7 Network Attacks	21
3.2.8 Signature and Signature Alarms	23
3.2.9 Intrusion Detection Evaluation	24
3.3 Genetic Algorithm (GA)	25
3.3.1 Working mechanism of Genetic Algorithm:	26
3.3.2 Steady State Genetic Algorithm (SSGA) elements	27
3.3.3 GA Forms	35
3.4 KDD Cup 99 Dataset	38
Chapter Four: Proposed Model & Methodology	41
4.1 Overview	41
4.2 Methodology	41
4.3 Proposed Model	42
4.4 The algorithm of Steady State Genetic Algorithm	43
4.5 Proposed Model Structure	43
4.5.1 Environment	43
4.5.1.1 Training Dataset	44
4.5.1.2 Testing Dataset	44
4.5.2 Classifier	44
4.5.3 Distinct Rules	45
4.5.4 Steady state Genetic Algorithm Unit	45
4.5.5 Rules Pool	47
4.5.6 Testing (Matching)	47

4.5.7 System Evaluation	47
4.6 Attack types and Features selection	47
Chapter Five: Experimental Results	49
5.1 Overview	49
5.2 Dataset	49
5.3 Classification	50
5.4 Features selection	51
5.5 Fitness Function	53
5.5.1 Why Fitness Function after using SSGA is equal to A	54
5.6 Tracking the genetic algorithm by values	56
5.7 The convergence	62
5.8 The Results of Detection Rate (DR) and False Positive Rate (FPR)	63
5.9 Tracking increasing of DR in U2R attack type with Double and Triple Replacement:	64
5.10 Comparing thesis results with other results:	66
Chapter Six: Conclusion and Future Work	68
6.1 Conclusion	68
6.2 Future Work	69
References	70
Appendix	75

List of Tables

3.1	The classification of main attacks categories	22
3.2	KDD'99 Feature Descriptions	39
4.1	Researches to determine the most significant features for each attack	48
5.1	Distribution of Attacks within Training Dataset	50
5.2	Sub attack types that appears in training dataset	50
5.3	Selected Features by Mukkamala, Sung and Abraham	51
5.4	Number of records before and after filtering for DoS attack	52
5.5	Number of records before and after filtering for R2L attack	52
5.6	Number of records before and after filtering for U2R attack	53
5.7	Number of records before and after filtering for Probe attack	53
5.8	Fitness value for each record in U2R	56
5.9	The values after selection process	57
5.10	Records form of U2R after crossover	57
5.11	The values of A for each record after mutation	58
5.12	Features values of U2R after crossover on the second generation	59
5.13	The values of A for each record after mutation in the second generation	59
5.14	Features values of U2R after crossover in the third generation	60
5.15	The values of A for each record after mutation on the third generation	60
5.16	Chromosomes value after Triple replacement process	61
5.17	Results of DR and FPR with Double Replacement	63
5.18	Results of DR and FPR with Triple Replacement	63

5.19	Number of records in the first 100 generations of U2R attack	64
5.20	Number of records after the 100 generations of U2R attack	65
5.21	First comparison	66
5.22	Second comparison	67

List of Figures

3.1 Misuse Intrusion Detection System	19
3.2 Anomaly Intrusion Detection System	20
3.3 Genetic Algorithm Structure	26
4.1 Enhanced Steady State Genetic Algorithm Model for Intrusion Detection System	42
5.1 The difference between double and triple replacement in the first 100 generation	64
5.2 The difference between double and triple replacement after the 100 generations	65

List of Abbreviations

<u>Abbreviation</u>	<u>Meaning</u>
AI:	Artificial Intelligent
BTR:	Binary Tournament Replacement
DOS:	Denial of Service
DR:	Detection Rate
FPR:	False Positive Rate
IDS:	Intrusion Detection System
ID:	Intrusion Detection
GA:	Genetic Algorithm
KDD:	Knowledge Discovery Database
NIDS:	Network Intrusion Detection System
NN:	Neural Network
R2L:	Remote to Local
RGA:	Recurrent Genetic Algorithm
SGA:	Simple Genetic Algorithm
SSGA:	Steady State Genetic Algorithm
TTR:	Triple Tournament Replacement
U2R:	User to Root

Abstract

In these days, Internet and computer systems face many intrusions without detection or prevention by security systems built for this purpose; large volume of important data is transferred through the networks such as personal data and credit cards information. Thus, the networks security has become more important than before. This led the researchers to make additional researches to develop Intrusion Detection Systems.

An Intrusion Detection System (IDS) is a system used to monitor the events and detect attacks; Steady State Genetic Algorithm is applied to support IDS by supplying the rule pool with additional data, these data can be used in testing phase to detect the attacks.

The main goal of this research is to enhance Replacement steady state genetic algorithm to detect intrusions. The enhancement has been achieved by comparing replacement methods.

This research provides that the Triple Tournament Replacement produces more accurate results than Binary Tournament Replacement in increasing Detection Rate (DR), but both of methods give the same values of False Positive Rate (FPR). The results of DR with Triple Replacement are 100% for three types of attack (DoS, Probe and R2L) and 53% for U2R. The average of DR of this system is 88.25%, and FPR is 1.48%. It's considered accepted results when comparing with the previous results.

الملخص

في هذه الايام،تواجه شبكة الانترنت ونظم الحاسوب العديد من التطفلالتبدون كشف او منع بواسطة نظم حماية مبنية لهذا الغرض، حجم كبير من البيانات المهمة تنقل خلال الشبكات مثل البيانات الشخصية ومعلومات بطاقات الائتمان، لذلك اصبح امن الشبكات اكثر اهمية من قبل، وهذا يقود الباحثين لمزيد من البحوث في هذا المجال لتطوير نظم كشف التطفل.

نظام كشف التطفل هو نظام مستخدملمراقبة الاحداث وكشف الهجمات، الخوارزمية الجينية لحالة الاستقرار مطبقة لدعم نظام كشف التطفل من خلال انتاج بيانات اضافية مستخدمة في مرحلة الاختبار لكشف الهجمات.

الهدف الرئيسي لهذا البحث هو تحسين ابدال الخوارزمية الجينية لحالة الاستقرار لكشف التطفل، وقد تم هذا التحسين من خلال مقارنة طرق الابدال. في هذا البحث توصلنا الى ان طريقة ابدال المجموعات الثلاثيةتعطي نتائج اكثر دقة من طريقة ابدال المجموعات الثنائية في زيادة معدل الكشف،لكن كل من الطريقتين اعطت نفس القيم لمعدل الكشف الايجابي.

حيث كان معدل الكشف معطريقة الابدال الثلاثية يساوي 100% لثلاث أنواع من التطفل وهي , DoS,Probe وR2L. بينما كان معدل الكشف للنوع U2R يساوي 53%.

وكان معدل الكشف الكلي88.25% ومعدل الخطأ الايجابي 1.48%. وهي تعتبر نتائج مقبولة بالمقارنة مع النتائج السابقة.

Chapter One

Introduction

1.1 Preface

Many studies and researches have worked hard to build smart and strong security systems to protect computer systems from intrusions (Ramakrishnan&Srinivasan, 2009).However, intrusion uses different ways to infiltrate into computer systems.

Security and trusted communicationof informationover internet and any other networks is always under threat of intrusions. So Intrusion Detection Systems have become a necessary component in terms of computer and network security. Generally, an intruder is defined as a system, program or person who tries to violate an information system or execute an illegal action,(Hoque, Mukit & Bikas, 2012).

Previous studies concentrated on developingprotection systems that detect the intrusion attack in early stage, and suggest cautions to the system administrator in order to take quick protection steps. On the other side, detecting intrusion must be done by using smart methods based on artificial intelligence techniques to detect the attacks.One of the most important approaches used to detect and prevent Intrusion is Genetic Algorithm (GA). Genetic Algorithms usually produce many solutions for the problem and thenselect the best solution based on two factors: Encoding and Evaluation Function(Jong, Chen, Su & Horng, 2005). The Genetic Algorithm evaluates each solution and then selects the most effective one.

1.2 Problem Identification

It is necessary to build a security system to protect the computer system against any attack. Since there is no perfect solution to prevent intrusions from violating system privacy, it is very important to be able to detect intrusions on time of occurrence and take actions to minimize the possible damage. Network Intrusion Detection System (NIDS) is one of the significant components in network security. NIDS must be more efficient to detect intrusions by enhancing replacement Steady State Genetic Algorithm to produce optimal IDS with high Detection Rate (DR) and low False Positive Rate (FPR).

1.3 Questions

The proposed system is expected to answer the following questions:

- Can the proposed system detect intrusions?
- Dose the proposed system detect intrusions with high detection rate (DR) and low False Positive Rate (FPR)?
- Does the proposed algorithm enhance the performance of intrusion detection system? Does it help to improve the results that are obtained previously?
- Is that optimal to use Steady State Genetic Algorithm (SSGA) with Misuse Detection?

1.4 Objectives

This research has many objectives that must be achieved to develop a smart and good intrusion detection system, as the following objectives:

- Enhancing Steady State Genetic Algorithm (SSGA) by comparing replacement methods to build a new system for intrusion detection.
- Increasing DR and decreasing FPR.
- Trying to find a new form for computing fitness function.
- Trying to enhance convergence problem.
- Comparing the results that will be obtained with the previous results produced from using steady state genetic Algorithm hoping that the proposed model will enhance the previous results.

1.5 Significance

The importance of this research is to enhance the existing IDSs. It deals with enhancement Steady State Genetic Algorithm with using Replacement approach.

The proposed model can lead to efficient intrusion detection system with high Detection Rate (DR) and low False Positive Rate (FPR).

1.6 Limitations

- The systems can be used for many actions to deal with intrusions: detection, prevention or both. This research deal only with detection system, since the prevention system also need to detection process at the beginning, then attempt to stop malicious activities.(Al-Rashdan, W.,2011)
- There are two ways to apply detection methodology: Misuse and Anomaly detection. In this research we performed Intrusion Detection System with misuse detection.

.1.7 Thesis Outline: The thesis is organized as follows:

- **Chapter One:** provides the Preface, Problem Identification, Questions, Objectives and the Significance of this work.
- **Chapter Two:** provides Literature Review and Related Work in the field of intrusion detection and genetic algorithm.
- **Chapter Three:** This chapter provides Theoretical Background about Intrusion Detection system. It will be described in details, including the concepts of computer security and firewall. Intrusion detection system classification and architecture will be explained. On the other side, this chapter presents knowledge about Genetic Algorithm by describing its components and operator types.
- **Chapter Four:** shows the stages of proposed Intrusion Detection System, and the elements of Steady State Genetic Algorithm. The methodology that has been followed to find the results will be discussed.
- **Chapter Five:** contains evaluation and experimental results for the proposed system, and comparing our method results with previous results.
- **Chapter Six:** In this chapter, the summarization of thesis and the future work related to this thesis domain have been presented.

Chapter Two

Literature Review & Related Work

2.1 Overview

This chapter introduces the previous researches related to using Genetic Algorithms for detecting intrusion. Many researches highlighted the intrusion argument and represented several detection systems to protect the user systems and programs from harmful intrusions. These studies also recommended some important methods and techniques for detecting intrusions that use many different methods and malwares. This chapter debates and highlights these methods and techniques according to the environments of intrusions.

2.2 Literature Review related to Intrusion Detection Systems:

Ramakrishnan and Srinivasan, (2009) explained that the intrusion operation forms are the greatest challenge in the Internet and they cause very difficult problems particularly when data theft which has been increased and it will never be decreased. The author claimed that there are several security systems for internet as human immune such as, intrusion detection which used to protect internal systems from any external attack.

In their paper, they used both technologies of intelligent agents and artificial immune system i.e., Agent Based Artificial Immune System (ABAIS) for security are presented.

However, the intrusion still challenge that faces security administrators who work hardly to decrease its attacks.

Ghali, (2009) introduced a new hybrid algorithm called Rough Set Neural Network Algorithm (RSNNA); for reducing the number of computer resources required for

detecting an attack. However, they used Knowledge Discovery Database(KDD cup99) in testing and the result indicated that the model had ability 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 Intrusion Detection, and did not test it on Misuse as this research did.

Some intrusions use malwares to infiltrate into user network and then into user systems and data. Ye, Li&Chen, (2010) proposed a principled cluster ensemble framework for combining individual clustering solutions based on the consensus partition. The researchers developed an automatic malware categorization system (AMCS) to group malware samples automatically. In their research, a hybrid hierarchical clustering algorithm is proposed to combine the merits of hierarchical clustering and k-medoids algorithms as a first part and a weighted subspace K-medoids algorithm as a second part for generating base clustering. The results of categorizations of the proposed system can be utilized to produce signatures for malware families that are beneficial for detecting malware. The proposed system was successful in clustering malware as it was concluded from the results of the practical experiments.

Moreover, Cesare& Xiang, (2010), highlighted that security threats like malware and intrusion are the most sophisticated. Also, these programs have been increased quickly because of the good environment for them. The researchers highlighted one of most important detection types “Signature-based” which is considered as beneficial system for dealing with malwares. In their research, two methods were used for dealing with malware (Static and Dynamic Analysis). The static analysis deals with malware at

source code level. Dynamic analysis deals with malware by extracting features for recognizing the malwares.

Chau ,Nachenberg, Wilhelm, Wright & Faloutsos (2010), introduced an approach called “Polonium” technique for detecting intrusion. However, the researchers claimed that the proposed technique can detect the malwares in an accurate scale. The proposed system calculates the file reputation depending on the fast and scalable belief propagation algorithm. The proposed method attained 85% true positive rate and the true positive rate further improves 2% in the case of more iterations. The researchers also detailed an important design for the proposed method that enables its successful application on their dataset. However, the researchers exploited the fact that some files exist together on a computer. Besides that, the research introduced a good and robust technique for classifying malwares. The researchers represented empirical observations on the largest file submissions dataset ever published which span over 60 terabytes of disk space. They detailed an important design and implementation features of the proposed method which enable its successful application on our large dataset.

The proposed polonium algorithm removed one path and took half of the storage. The proposed system has weaknesses in detecting malware because it depends on the file reputation. However, this system does not benefit in providing new smart methods and techniques compared with other researches.

Intrusions always reach to user systems by infiltration; which is considered the most danger case that may appear in many systems. However, there is no full protection that can keep user systems from intruders who use malware as one of their tools for reaching to user data. In the context with this argument the researchers Ashoor, and Gore, (2011)

highlighted that the intruder's activity on the network poses risk attacks user data. Furthermore, the researchers proposed some techniques and methods that help in detecting intrusion activity and malware infiltration into network or systems. Neither intrusion nor malware has a specific form or a regular activity to be identified by users. However, this article aimed to explain and indicate the stages of the evolution of the IDS idea and its importance to researchers. On the other side, the researchers categorized IDS into Host based IDS, Network based IDS and Hybrid based IDS. However, all the mentioned techniques in this article are familiar and used in many security systems. Finally, the researchers concluded that the intrusion detection systems and malware detection systems are not distinguished because they aimed to use the same elements and techniques of protection.

Modi et al,(2013) represented intrusion detection techniques that are used in cloud to protect cloud systems from intrusion. They mentioned several types of intrusion used to violate the cloud systems such as: Insider attacks, Flooding attacks, User to Root attacks, Port Scanning, attacks on Virtual Machine (VM) and Back door Channel Attacks. These types are used to violate the weakness points in these situations. For this purpose, the authors represent the security techniques that used to protect cloud system from these attacks by detecting the intrusion activity in specific layer. These techniques include: Firewall which is considered as a common solution for intrusion. Firewall works by denying or allowing protocols, ports and IP's addresses. Besides that, there are other techniques mentioned by authors such as: signature based detection, Anomaly detection, Artificial Neural Network detection and Genetic Algorithm based Intrusion Detection. Authors concluded that, Firewall is not enough to solve cloud security issues.

2.3 Literature Review related to Genetic Algorithm:

Many researchers studied the Genetic Algorithms (GA) and exploited its capabilities in designing smart systems and solving problems.

Haupt and Haupt,(2004) explained practical GA in four chapters and indicated the concept of finding the best solution by optimizing GA. The author explained that GA can be used for many purposes and objectives in solving different problems. Also, the authors indicated the concept of parallel GA which is an important and advanced way in solving a problem, but it needs more work as the author mentioned. Also, the study explained the encoding and decoding processes that are used in GA to select the best chromosomes. There are many advanced applications represented in the study that depend on GA such as: Decoding and Secret Message, Stealth design and Repot Trajectory planning.

Al-Sabbah, (2012),presenteda model for enhancing the color image using the steady state genetic algorithm (SSGA), andused modified fitness function to achieve more accurate results with less noise. The main objective of this research is to process the image so that the result is more suitable than the original image.The researcher developed three models for enhancing the colourful of the image with variety types of input - output and different type of parameter. The advantage of this research is combining the chromaity components of the image; the image that has been enhanced from three perspectives.

2.4 Literature Review related to Intrusion Detection Systems (IDS) based on Genetic Algorithm (GA):

Some studies involved GA in their proposed systems in order to get the best solution for intrusion detection.

Selvakani, Rajesh, (2007) executed GA-based approach for Network Misuse Detection. Also, they used a technique for generating rules for R2L and DoS attacks. Besides that, they used KDD Cup 99 data set for finding the probability of Detection as well as they found the overall performance = 59% Detection Rate and False Alarm Rate=0.1%. In details, they used GA-based approach for Network Misuse Detection for framing the rules required. They worked just with R2L and DoS attacks, they have Low DR. But in our research, we deal with four types of attacks: R2L, DoS, U2R and Probe with Detection Rate 88.25%.

Goyal and Kumar, (2009) studied current intrusion types that attack user systems and they proposed machine learning approach based on using GA. Their algorithm takes into consideration the network protocol and IP's types. Besides that, the proposed algorithm depends on the type of network service on the destination and the connection status. They implanted their algorithm on KDD 99 datasets in order to produce a rule set that can be applied on IDS. Authors claimed that they succeeded in applying their algorithm and generated the rule that classify all Smurf type of attack connections. The rule set classifies Probe attacks into 52 attack connections. To check the algorithm efficiency, positive false rate which gives result about 0.2% and they calculated accuracy that gives result about 100%. They used simple genetic algorithm, but in this research we used Steady State Genetic Algorithm.

On the other hand, Al-Sharafat, (2009) highlighted the techniques used for detecting intrusion using Genetic Algorithm (GA). The author represented the intrusions that attack the network, so the security must be placed on specific situations on the network.

However, the author used Steady State Genetic Algorithm (SSGA) with Anomaly based IDS. While our research used SSGA with Misuse IDS, depending on another fitness function.

The author got detection rate (DR) reaches to (97.45)%. She also compared between the algorithms that usually used for detecting intrusion using DR parameter. On the other side, the author concluded that SSGA needs more modifications by using binary encoding to find out its impacts.

Stewart, (2009) modified GA and NN in order to propose a new combination of GA and NN to be able of tuning not only the weighting of a NN, but also its size and connection. The enhanced 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.

Agravat&Rao, (2011) explained two objectives of fuzzy Genetic-based learning algorithms and debated its usage for detecting intrusion in a computer network. However, the objectives aim for minimizing the number of fuzzy rules and maximizing the classification rate. The researchers used 10% labeled data for training and testing the GA, and they used 20 features instead of 41 from the KDD Cup 99 dataset. But in our research we used 5 features instead of 41 from the same dataset.

Hoque, Mukit & Bikas, (2012) proposed detection system using GA and applying their algorithm on specific datasets of intrusions detection (KDD'99) which is based on 1998 DARPA. Their implementation is based on defining datasets as chromosomes and then making GA produces the parents and children to select the best chromosome. However, they chose range value about 0.125 for selection purpose, and they conducted crossover and mutation in next stages. 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.

In addition to that, Naoum, Abid & Al-Sultani, (2012), classified intrusions by using an enhanced Resilience Back Propagation Neural Network. After they had completed their system and testing the proposed system, they got 94.7% average Detection Rate, and 15.7% False Positive Rate. But our thesis used Genetic Algorithm instead of Resilience Back Propagation Neural Network, and got another results.

Kshirsagar, Tidke & Vishnu, (2012) used GA with Data mining technique to develop intrusion detection system. The proposed system depended on data mining methods and they proposed their system in a model called "Data Mining Hybrid IDS". The model contained three sensors located at network sensor manager that is connected with data "warehouse unit". The data is transferred between warehouse and network sensors and pattern mining. The proposed system enabled the administrator to prevent the intruder activity by data mining techniques and sensor snort located on the network.

Mostaque, (2013) designed an Intrusion Detection System (IDS), by applying genetic algorithm (GA) and fuzzy logic to detect different types of attacks within a network efficiently. The proposed fuzzy logic-based system could be able to detect the

intrusive activities of the computer networks as the rule base holds a better set of rules. The environment of his proposed system was KDD Cup 99 dataset. He used Simple Genetic Algorithm (SGA) with Anomaly detection, but in this research we used Steady State genetic Algorithm (SSGA) with Misuse detection. The experimental results illustrated that the proposed system got higher accuracy in determining whether the action are normal or attack, and got a good detection rate.

Chapter Three

Theoretical Background Intrusion Detection & Genetic Algorithm

3.1 Overview

This chapter introduces the knowledge about Intrusion Detection system and its components, Intrusion Detection taxonomy and evaluation. On the other side, this chapter explains the elements of Steady State Genetic Algorithm, and its working mechanism.

Finally, it will explain the population that will be used in this research.

3.2 Intrusion Detection

It is important to build a system that has the ability to protect user system and data from intruders.

Intrusion can be denoted as any set of actions that try to compromise the integrity, confidentiality or availability of a computer resource. This act can be referred as intrusion detection, (Hoque, Mukit & Bikas, 2012).

Intrusion detection (ID) is a process of discovering the intrusion activity. It can be classified into two types Anomaly and Misuse intrusion detection (Ramakrishn & Srinivasan, 2009). In our research we used Misuse Detection.

3.2.1 Computer Security

The security is a process where management must apply a security program which includes attention, prevention, detection and management to minimize the dangers. It is a process that cannot be perfect, but it helps to reduce attacks and manage the risks. The aim is to make controls that are used and put into a place to defend against various attacks. To implement network security there must be a security policy which is the basic of that process. (Siva, Vinay & Babu, 2013)

Computer System Security can be defined as a process of protecting main aspects for any computer system security. Those aspects are: confidentiality, integrity and availability, which are referred in the abbreviation CIA. (Bishop, 2005)

These three aspects are defined as follows:-

Confidentiality is the hiding of information or resources. The need for keeping information secret comes from the use of computers in sensitive fields such as government, the military field and all the types of origination that keep personnel information secret. Confidentiality also implements to the existence data as well as hiding resources.

Integrity refers to the trustiness of data or resources, and it is usually formulated in terms of wrong preventing or unauthorized change. Integrity includes data integrity (the content of the information) and origin integrity (the source of the data). The source of the information may relate its accuracy and believability.

Working with integrity is very different from working with confidentiality. With confidentiality, the data are either corrected or not, but integrity includes both the correctness and the trustiness of the data.

Availability refers to the ability of using the information or resource required. Availability is a significant aspect of credibility as well as of system design because an unavailable system is at least as bad as no system at all. The aspect of availability that is appropriate to security is that someone may try to prevent access to data or to a service by making it unavailable. Attempts to obstruct availability, called Denial of Service attacks (DoS). (Bishop, 2005)

Importance of those properties varied and connected to the company's business.

3.2.2 Firewall

Firewall is a combination of hardware and software that separates an organization's internal network from the Internet; and it is a protective system. (Kurose & Ross, 2010)

Firewalls play a significant role on any network as they supply a protective block against most forms of attack coming from the outside.

The function of a firewall is to traffic from outside to inside, and vice versa, passes through the firewall.

Only authorized traffic, as defined by a local security policy, will be allowed to pass the firewall itself should be secure against permeation. The firewall itself is a device connected to the network, If not designed or installed properly, it could be vulnerable. (Kurose & Ross, 2010)

3.2.3 Intrusion Detection Systems (IDS)

In latest years, Intrusion Detection System (IDS) has become one of the hottest research areas in Computer Security. It is a significant detection technology and it is used to maintain data integrity and system availability during an intrusion, (Li, 2004).

An Intrusion Detection System (IDS) is defined as a security system that monitors computer systems and analyzes that traffic for possible attacks coming from outside the organization and also for system misuse or attacks from inside the organization, (SANS, 2001).

The authors define an Intrusion Detection System (IDS) as a software or hardware or combination of software and hardware that are used to detect the intruder activity (Rehman, 2003).

That is IDS produces an alert to the security administrators, to notify about the attack (Scarfone & Mell, 2007).

Bishop, (2005) mentioned Intrusion Detection System goals as follows:

1. **Detect a wide variety of intrusions:** Intrusions from within the site, as well as those from outside the site, are of concern. In addition, both known and unknown violations should be detected. This suggests a mechanism for learning or adapting new kinds of attacks.
2. **Detect intrusions in a timely manner:** "Timely" here is to distinguish the intrusion within a short period of time not exactly in real time.
3. **Present the analysis in a simple, easy- to- understand format:** This should alert the system green if there is no detection and red when an attack is detected.

So intrusion detection mechanisms must present more complex data to a site security officer.

4. **Accurate:** A false positive happens when an intrusion detection system discovers an attack, but no attack is underway. False positives reduce the trust in the correctness of the results and increase the quantity of work involved.

False negative is worse because it happens when an intrusion detection system fails to discover an ongoing attack, where the purpose of these systems is to report attacks. The object of an intrusion detection system is to decrease both types of errors.

3.2.4 Components of intrusion detection system

An intrusion detection system usually consists of three functional elements. The first element of an intrusion detection system is a *data source*. Data sources can be classified into four categories: Host-based monitors, Network-based monitors, Application-based monitors and Target-based monitors. The second element of an intrusion detection system is known as the *analysis engine*. This element takes data from data source and analyses the data to attacks. The analysis engine can use Misuse or Anomaly detection.

The third element of an intrusion detection system is the *response manager*. In basic terms, the response manager will only act when lacking of precision (possible intrusion attacks) is exist on the system, by informing someone or something in the form of a response. (Mostaque, 2013)

3.2.5 Intrusion Detection Classification

Intrusion Detection can be classified into two categories: misuse intrusion detection and anomaly intrusion detection. (Li, 2004), (Ramakrishn& Srinivasan, 2009)

-Misuse (signature) intrusion detection: refers to the techniques that recognize known violations as ‘pattern’ or ‘signature’, and compared them with the observed events to detect unknown threat.

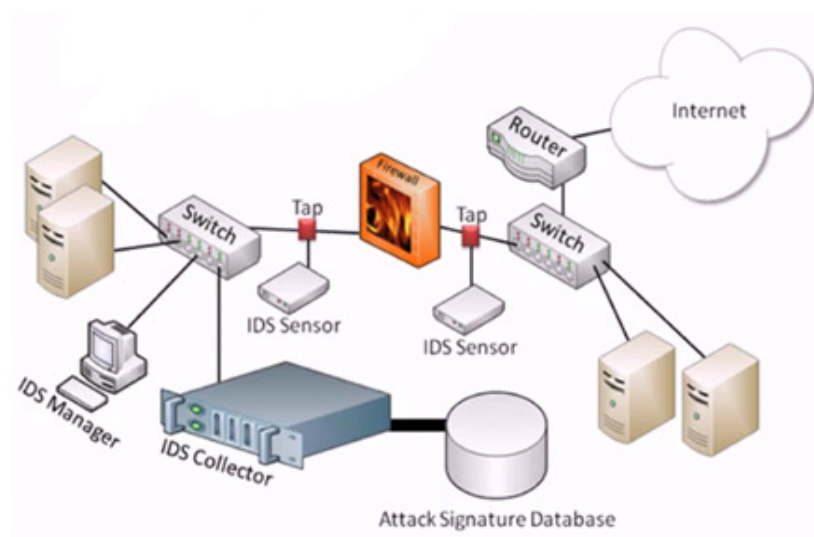


Figure (3.1) Misuse Intrusion Detection System (Gadbois, 2011)

-Anomaly Intrusion Detection System:(behavior detection)refers to the techniques that characterize normal actions of the system;actions that deviate from the expected normal action are considered intrusions.

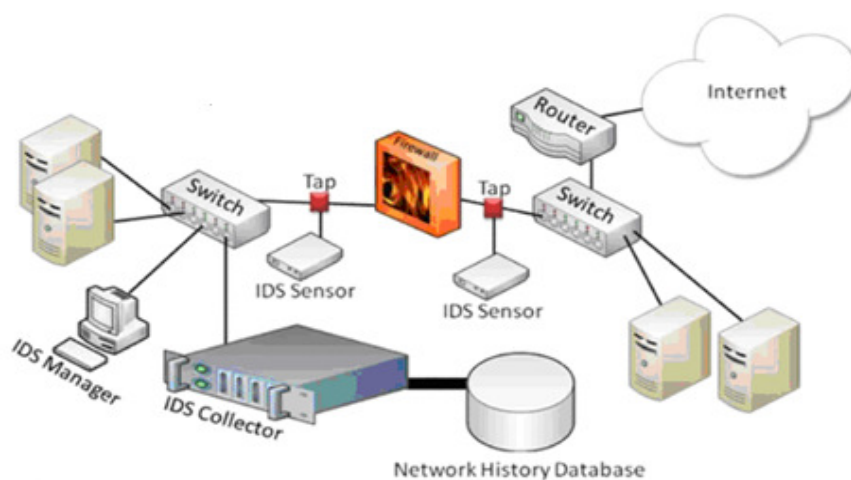


Figure (3.2) Anomaly Intrusion Detection System (Gadbois, 2011)

3.2.6. Intrusion Detection Systems Taxonomy

a. Knowledge based system vs. behavior based system:

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 which means that the system will have very low false alarm rates. 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. However, this method has many disadvantages .The first one is that the information must be up to date to have a good and an effective knowledge based IDS. The second one is that the information must be gathered after a detailed analytical process over each attack. So there is difficulty in gathering information. This method faces a generalization issue. Misuse IDS, sometimes called signature, its considered as knowledge based IDS, (Al-Sharafat, 2009)

b.Host based System vs. Network based System

For recognizing and catching the attack and infiltration, IDS usually uses Network based IDS or Host based IDS. However, both of them look for specific pattern which considered as signature to explain malicious activities or policy violations. If the recognition and indication are conducted by using IDS over network traffic, such as traffic volume, protocol usage..etc, then it will be called Network based IDS,

Also, Haupt,& Haupt (2004) indicated that these two technologies are similar in the root, but they are different in their operational use. However, Network Intrusion Detection is used for analyzing network packets and examines events as information packets, so it demonstrates the abuse of vulnerabilities, but the detection engine in this technology cannot detect encrypted.The current trend in intrusion detection is to combine both host based and network based information to develop hybrid systems that have more efficiency in security level. (Jaiganesh, Mangayarkarasi& Dr. Sumathi, 2013)

3.2.7Network Attacks:

Attacks types are classified according to the following categories: (Tavallae, Bagheri, & Ghorbani, 2009)

- 1) **Denial of Service Attack (DoS):** Is an attack in which the attacker makes some computing or memory resources too busy or too full to prevent legitimate users access to system. DoS attacks are classified based on the services which an attacker renders unavailable to legitimate users.
- 2) **User to Root Attack (U2R):** occurs when anattacker starts out withaccess to a normaluser account on the systemand is able to exploit some vulnerability to gain rootaccess to the system.

3) **Remote to Local Attack (R2L):** occurs when an attacker has the ability to send packets to a machine over a network, and then exploits the machine's vulnerability to lawlessly gain local access as a user.

4) **Probing Attack:** Is an attempt to learn information about a network of computers to look for exploitation. Probing is a class of attacks where an attacker scans a network to gather information or to find known vulnerabilities. There are different types of probes. Some of them exploit the computer's legitimate features and some of them use social engineering techniques. This class of attacks is the most commonly used and requires a little technical expertise. Lahre, Diwan, Kumar and Agrawal, (2013), displayed a table about classification of the main categories of attacks into sub attacks types as the following table:

Attack type	Sub attacks types
Normal	Normal
DoS	smurf, teardrop, pod, back, land, apache2, udpstorm, mailbomb, processtable, Neptune
Probe	ipsweep, portsweep, nmap, satan, saint, mscan
R2L	dictionary, ftp_write, guess_password, imap, named, sendmail, spy, xlock, xsnoop, snmpgetattack, httptunnel, worm, snmpguess, multihop, phf, wraezclient, wraezmaster
U2R	perl, ps, xterm, loadmodule, eject, buffer_overflow, sqlattack

Table (3.1): the classification of main attacks categories

3.2.8 Signature and Signature Alarms:

A **signature** is a set of rules that an IDS uses to detect intruder action, such as U2R attacks.

Sensors scan network packets, using signatures to detect known violations and compare with predefined actions. A virulent packet flow has a specific type of action and signature, and an IDS sensor analyzes the data flow using many variant signatures.

Signature Alarms:

The efficiency of IDS sensors to detect an attack in accurate manner and generate an alarm is critical to the role of the sensors. Attacks can generate the following types of alarms:

- **False positive:** A false positive is an alarm occurred by normal traffic or non-malicious action. For example a wrong password entered mistakenly by a real user may result in false positive. The sensor cannot distinguish between a hacker user and a mistaken user; false positives can be minimized by regulating the sensors.
- **False negative:** A false negative occurs when a signature is not fired, when disturbing traffic is detected. A false negative should be regarded a software bug only if the IDS has a signature that has been designed to detect the disturbing traffic.
- **True positive:** A true positive occurs when an IDS signature is correctly fired, and an alarm is generated, when disturbing traffic is detected.
- **True negative:** A true negative occurs when a signature is not fired. when benign traffic is captured and examined. In other words, the sensor does not fire an alarm when it captures and examines “normal” network traffic. (Singh, 2013)

3.2.9 Intrusion Detection Evaluation:

Network Intrusion Detection System must be accurate to increase the trust of the system. Thus we must increase Detection Rate DR and decrease False Positive Rate FPR. Kuang, (2007) defined (DR) and (FPR) as following:

Detection Rate (DR) is “the ratio of correctly classified intrusive examples to the total number of intrusive examples”.

False Positive Rate (FPR) is “the ratio of incorrectly classified normal examples (false alarms) to the total number of normal examples”.

$$DR = \frac{\sum \text{no. of detected attacks}}{\sum \text{no. of total attacks}} \quad (3.1)$$

$$FPR = \frac{\sum \text{no. of false alarms}}{\sum \text{no. of total normal records}} \quad (3.2)$$

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

$$\text{Precision} = TP / (TP + FP) \quad (3.3)$$

$$\text{Recall} = TP / (TP + FN) \quad (3.4)$$

$$\text{Overall Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (3.5)$$

Where:

TP = True Positive, TN = True Negative

FN = False Negative, FP = False Positive

3.3 Genetic Algorithm (GA)

GA depends on producing Chromosomes to produce more advanced chromosomes that considered advanced solutions. At each generation, the best chromosome will be selected based on computing fitness function of each chromosome, and then chromosomes are selected according to their fitness values.

Genetic Algorithm is “an adaptive heuristic search method based on population genetics”.

Genetic algorithm is a search method based on the principles of natural selection and genetics (Kumar, Husian, Upreti & Gupta, 2010).

Genetic Algorithm is based on Darwin’s principles in optimizing the chromosome population of candidate solutions (Li, W., 2004).

Most of the traditional methods of the problem optimization, for complex problems have suffered from many problems and difficulties.

Therefore, the researchers try to use Genetic Algorithm properties as a suitable technique.

In addition, genetic algorithm has the following properties: (Goldberg, D., 1989)

1. GA works with the parameters code, not with the parameters themselves.
2. GA searches for a set of points, not for a single point.
3. GA uses available information, not derivative or other auxiliary knowledge.
4. GA uses probabilistic transition rules, not deterministic ones.

These are the reasons which encourage researchers to use GA as a General Search method depending on natural selection genes principles which are developed by Holland in 1975 in his book "Adaptation in Natural and Artificial Systems" (Mitchell, M. 1996).

3.3.1 Working mechanism of Genetic Algorithm:

The algorithms start with random population of chromosomes (individuals) which have a determined length, then evaluate fitness of chromosomes by measuring the value of fitness function for each one, then generate a new generation by applying genetic operators such as selection, crossover and mutation frequently, until the stopping criterions achieved and the best chromosome presented.

The flowchart below shows the stages of GA.

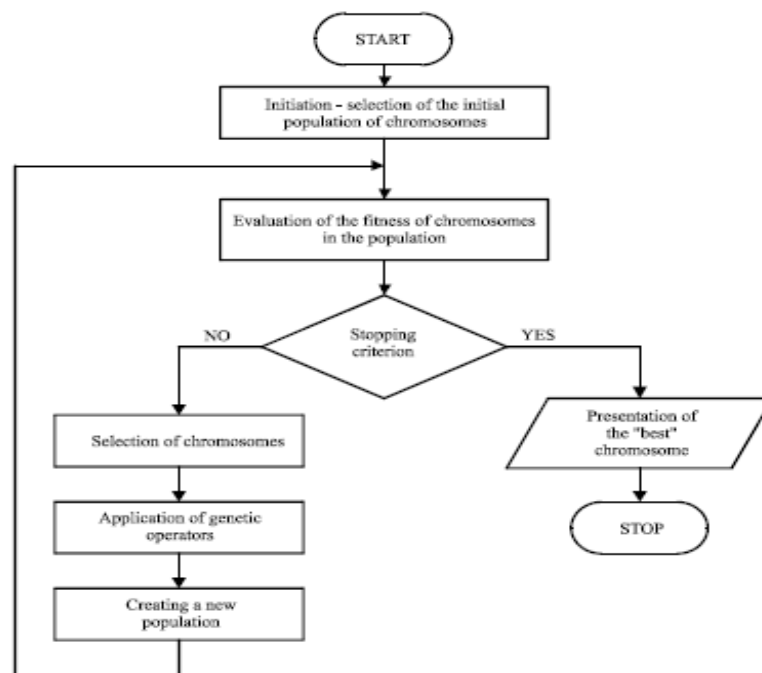


Figure (3.3) Genetic Algorithm flowchart(Al-Sharafat,2009)

3.3.2 Steady State Genetic Algorithm (SSGA) elements

Steady state genetic algorithm consists of a set of elements:

-Population:

Population is a set of individuals(chromosomes)of a specified size. The size of population is determined according to the nature of the problem. Generally the population is generated randomly (Kumar, Husian, Upreti& Gupta, 2010).

The population is the result of a single iteration of genetic algorithm. However, iteration can create new population which contains a set of chromosomes, and each chromosome is one complete possible solution to the problem to be solved using genetic algorithms, and more generations are needed for finding optimal solutions.

-Evaluation:

Each individual has a fitness value used to evaluate the fitness for each one. Fitness value evaluates the quality of each chromosome, so the high fitness value gives the chromosome high probability to be selected in the selection stage (Mitchell M., 1996).

There are many equations used for evaluation such as:

$$\text{Fitness} = 1 / (1 + f(x_1, x_2 \dots x_n)) \quad (3.6)$$

$f(\dots)$:objective function

n :number of the variables in the objective function

Al-Sharafat, (2009) developed two new functions, fitness function and strength function as shown below:

New Fitness Function:

$$Fw_i = \left(\frac{f_i}{\sum_{i=1}^n f_i} \right) * \left(\frac{DR}{n} - \frac{FPR}{n} \right) \quad (3.7)$$

New Strength Function:

$$Sw_i = f_i * \text{age } i \quad (3.8)$$

Where:

n: number of rules

DR: Detection Rate

FPR: False positive rate

f_i : fitness for rule i

age i : age of rule i in classifier

Sw_i : The result of new strength of the rule i .

Fw_i : The result of new fitness function for the rule i

Alabsi and Naoum, (2012), presented new Reward Penalty Fitness Function as the following equation:

$$\text{Fitness} = 2 + \frac{AB - A}{AB + A} + \frac{AB}{X} - \frac{A}{Y} \quad (3.9)$$

The values of A and AB depend on the condition and action parts, when the condition and action of the selected record equal to the condition and action of the Compared record then the value of AB of the selected record will increase by one, else the value of A of the selected record will be increased by one.

X = the maximum value of AB in the population.

Y = the maximum value of A in the population.

-Encoding:

Encoding is one of the significant processes in Genetic Algorithm to represent solutions.

The gene in genetic algorithm is considered the problem parameter which can be encoded as one of the encoding methods.

There are different methods for encoding such as:

- **Binary Encoding:** is a common encoding method in GA.

This method converts the value of parameters into binary value (0, 1) e.g., chromosome = 10000101.

- **Integer encoding:** in this method, Chromosome is represented by using integer numbers.

- **Real encoding:** This method uses actual real values of chromosome; we used this method of encoding in our research.

1.2	2.2	0.4	0.672
-----	-----	-----	-------	-------

To enhance the efficiency of GA in solving problems, suitable representation method must be chosen.

-Selection

Selection is a process of choosing individuals (parents) from current population to implement operations of crossover and mutation on them to generate new individuals, (Mitchell M., 1996). There are different methods of selection:

- **Roulette wheel Selection:**

It is a selection method used in SGA for selecting individuals with high fitness value.

Roulette wheel selection works by calculating expected probability for each individual "e_i" as following:

$$e_i = \frac{F_i}{\bar{F}} \quad (3.10)$$

$$\bar{F} = \Sigma F_i / n$$

Where:

F_i: Individual fitness,

\bar{F} : Average fitness,

n: number of individuals in generation

In general, there are many problems in this method such as: Premature convergence; where convergence happens in GA at the beginning the processes. Individuals that have high fitness value will be dominant on others. In addition, the dominant individuals will stay in next generation. That means; individuals with low fitness value will be eliminated early from the population.

Slow finishing; slow finishing happens at the end of GA processes. At the end of the execution, there will be a large number of individuals with similar fitness values. In this case, which one will be selected? And we cannot be able to recognize between them.

Finally, there is no guarantee to keep the best individual for next generations. To improve this problem, we can use Elitist selection method.

• ***Elitist Selection:***

This method keeps a finite number of the best individuals in each generation, since these individuals may not be chosen or disrupted during crossover and mutation.

• ***Ranking selection:***

There are many ways to perform this method, but the simplest way is linear ranking method. This method is proposed by Backer in 1985, where individuals of a population are ordered descending (from 1 to N_{pop}) according to their fitness value. An individual with the highest fitness value takes the rank=1, the next takes the rank=2 and so on (Schmidt, & Stidsen 1997).

$$F' = \max - (\max - \min) \left[\frac{(\text{rank}-1)}{(N_{pop} - 1)} \right] \quad (3.11)$$

Where:

$$1 < \max \leq 2 \text{ \& rank } \in \{1, 2, \dots, n\} \text{ \& min} = 2 - \max$$

• ***Stochastic Universal Sampling:***

In this method, the Roulette Wheel can be spin just once instead of spinning it n times as it was described in Roulette Wheel selection.

This method tries to reduce the difference between the actual value and the expected one.

According to the following relation, probability of selected individuals must be within the expected and actual value of that individual (Mitchell, M. 1996).

$e_i \geq$ probability of $i \geq e_i$ (3.12)

Where:

e_i : actual value for i

e_i : expected value for i

This method still suffers from premature convergence and slow finishing problem such as roulette method.

• **Tournament selection:**

There are many ways to perform this method; the simplest and the most common one is Binary Tournament Selection.

We can formulize this method as shown:

$$\text{Select}_n = \begin{cases} \text{ind}_i & \text{if } F(\text{ind}_i) > F(\text{ind}_j) \\ \text{ind}_j & \text{otherwise} \end{cases} \quad (3.13)$$

For $n = \{1, 2\}$, random $i, j \in \{1, 2, \dots, N_{\text{pop}}\}$, $i \neq j$

Where:

Select_n : selected individual that has number n ,

$F(\text{ind}_i)$: fitness of individual i ,

$F(\text{ind}_j)$: fitness of individual j .

-GA Operator

There are two main types of operator which are used to reproduce new individuals in the next generations: Crossover and Mutation.

Crossover: it is the process of convert of genes between two individuals to reproduce new individuals (offspring's) which inherit their parent's behavior.

There are many types of crossover:

- **Single-point crossover (1x)**

Single Point method aims to select crossover point and interchange the two parent's chromosomes after this point for producing two new offspring's.

- **Two-point crossover (2x)**

Two Points method aims to select two crossover points and interchange the two parents chromosomes between these points.

- **Uniform crossover (UX)**

Uniform method differs from other methods where genes are randomly exchanged, in order to gain a high diversity in populations.

Mutation is a randomly changing of genes in chromosome. One of the strong features of mutation is creating new individuals different from the existing ones, the probability of the occurrence of mutation is assumed ($0 \leq P_m \leq 1$). However, mutation has many types: Flip bit, Boundary and Uniform.

-Replacement

It is a process performed on the worst individuals to be replaced by new better individuals. There are two methods of replacement:

- **Binary Tournament Replacement (BTR)**

It will choose the best chromosome from two according to their fitness values.

$$\text{Replace}(n) \begin{cases} \text{ind } i & \text{if } F(\text{ind } i) < F(\text{ind } j) \\ \text{ind } j & \text{otherwise} \end{cases} \quad (3.14)$$

For $n = \{1,2\}$, random numbers $i, j \in \{1,2,\dots,N_{pop}\}, i \neq j$

Where:

Replace(n): individual n that will be replaced

$F(ind\ i)$: fitness of individual i

$F(ind\ j)$: fitness of individual j

• *Triple Tournament Replacement (TTR)*

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

$$\text{Replace}(n) \begin{cases} ind\ i & \text{if } F(ind\ i) < F(ind\ j) \text{ and } F(ind\ i) < F(ind\ k) \\ ind\ j & \text{if } F(ind\ j) < F(ind\ i) \text{ and } F(ind\ j) < F(ind\ k) \\ ind\ k & \text{otherwise} \end{cases} \quad (3.15)$$

For $n = \{1,2\}$, random numbers $i, j, k \in \{1,2,\dots,N_{pop}\}, i \neq j, i \neq k, j \neq k$

Where:

$F(ind\ i)$: fitness of individual i

$F(ind\ j)$: fitness of individual j

-**Stopping criteria**

There is repeating in the evolution process of GA, until satisfaction of the Stopping condition. There are numbers of criteria such as: *On-line performance*

$$\text{On-line}(T) = \frac{1}{T} \sum_{t=1}^T F(t) \quad (3.16)$$

Where

T: number of times to find fitness,

F(t): binary evaluation for fitness values.

(Kumar, Husian, Upreti & Gupta 2010) cited common Stopping condition such as:

- A solution is found that satisfies minimum criteria.
- Reaching to the fixed number of generations.
- Allocating budget (time, money) reached.
- Sequential iterations no longer produce better results.

3.3.3 GA Forms

There are many forms of GA such as: Simple Genetic Algorithm (SGA), Steady State Genetic Algorithm (SSGA) and Recurrent Genetic Algorithm (RGA).

1. Simple Genetic Algorithm (SGA)

This form of GA focuses on the reproduction of a new generation with whole replacement of the previous one. So, there is no intersection between them. SGA works as follows: (Mitchell M., 1996)

Algorithm of SGA:

```

Begin
t=0
Initialize P (t) // P(t) = Population
Evaluate P (t)
While (Termination condition is not satisfied) do
    Selection P(t+1) from P(t)
    Perform Crossover on P(t+1)
    Perform Mutation on P(t+1)
    Evaluate P(t+1)
    t=t+1
End

```

2. Steady State Genetic Algorithm (SSGA)

In SSGA, generation is changing gradually by replacing unwanted individuals partially; a part of the population is transferred to the next generation without any changes. So, there are intersected generations where a set of individuals are replaced by new set.

Algorithm of SSGA:

```

Begin
t=0
Initialize P (t) // P(t) = Population
Evaluate P (t)
While (Termination condition is not satisfied) do
    Selection P(t+1) from P(t)
    Perform Crossover on P(t+1)
    Perform Mutation on P (t+1)
    Evaluate P (t+1)
    Replacement (P(t), P(t+1))
    t=t+1
End

```

SSGA has a set of characteristics which excels SGA such as:

- The ability to find solutions by using small samples contrary to SGA, whereas SSGA can select a new good individual as soon as they are generated , while SGA must examine all population to select good individuals.
- SSGA avoids repeating the same individual in population, so generation will contain plenty of different solution.
- SSGA has more steadiness against genetic divergence). For example, if we suppose that an individual is constructed from good genes, this individual has a high fitness value. These individual genes will be torn up by using crossover operator to produce new

individuals. The original individual (the parent) will still be a part of a current generation and can be selected to cross over again unlike SGA.

3. Recurrent Genetic Algorithm (RGA)

It is a new strategy that enhances the GA; this strategy concentrated on Crossover and Mutations stages. In Fakeih, A., Kattan, A., (2012) Authors supposed t'_i as an intermediate population located between t_i and t_{i+1} to be feedback population. However, the algorithm uses fitness to reward parents in population t_i . They also provided equations of *Fitness Reward Function* (FRF) for crossover and mutation stages in order to determine the process of iteration on these stages.

For each crossover operator that parents P_x and P_y joined, where $x, y \in \{1, 2, \dots, \text{population size}\}$, we use the following *FRF*:

$$FRF(\text{Parent}_x \text{Fitness}) = \text{Offspring Fitness} \times \text{Parent}_x \text{ contribution}$$

$$FRF(\text{Parent}_y \text{Fitness}) = \text{Offspring Fitness} \times \text{Parent}_y \text{ contribution}$$

Where, *Offspring Fitness* is the fitness value of the generated offspring, *Parent x contribution* and *Parent y contribution* are real numbers from the interval(0, 1) to represent the proportion of genetic materials that each parent contributed when generating the offspring. Note that *Parent x contribution* + *Parent y contribution* = 1.

For each mutation operator that parent P_x joined, we use the following *FRF*:

$$FRF(\text{Parent}_x \text{Fitness}) = (\text{Offspring Fitness} \times \text{Parent contribution}_x)$$

Here, because the mutation operator is based on single parent, *Parent x contribution* is calculated as the amount of genetic materials that passed from the parent into the offspring.

3.4 KDD Cup 99 Dataset

In 1998, an “Intrusion Detection Evaluation Program (IDEP)” managed by the Lincoln Laboratory at the Massachusetts Institute of Technology. This program was achieved to build a data set that would help evaluate different intrusion detection systems (IDS) in order to evaluate their strengths and weaknesses. This data set is popularly known as DARPA 1998 data set. (Darpa, 1998)

This data set named in the literature as The Knowledge Discovery in Databases (KDD) and later used in KDD Cup 1999 data set, which was introduced for the development of intrusion detection systems through a suite of pattern recognition and machine learning algorithms for main attack categories: namely Probing, Denial of Service (DoS), User-to-Root (U2R) and Remote-to-Local (R2L). (Sabhnani and Serpen, 2004)

Many studies are interested in KDD Cup 99 dataset because of its role in the field of intrusion detection systems and benchmarking analysis of attack over the network in the last decade. Many researchers contributed their efforts in analyzing the dataset using several techniques. (Siddiqui & Naahid, 2013)

KDD CUP 99 dataset consists of 4,940,210 records; each record has 41 features labeled as either normal or an attack, with feature number 42 which determines the type of attack. Sathya, Ramani, and Sivaselvi, (2011) displayed the 41 features and their description as the following table:

Feature No.	Feature Name	Description
1	Count	Number of connections to the same host as the current connection in the past two seconds
2	Destination bytes	Bytes sent from destination to source
3	Diff srvrate	% of connections to different services
4	Dst host count	Count of connections having the same destination host
5	Dsthost diff srv rate	% of different services on the current host
6	Dst host rerror rate	% of connections to the current host that have an RST error
7	Dsthost same src port rate	% of connections to the current host having the same src port
8	Dsthost same srv rate	% of connections having the same destination host and using the same service
9	Dsthost serror rate	% of connections to the current host that have an S0 error
10	Dsthost srv count	Count of connections having the same destination host and using the same service
11	Dsthost srv diff host rate	% of connections to the same service coming from different hosts
12	Dsthost srv rerror rate	% of connections to the current host and specified service that have an RST error
13	Dst host srv serror rate	% of connections to the current host and specified service that have an S0 error
14	Duration	Duration of the connection.
15	Flag	Status flag of the connection
16	Hot	Number of "hot" indicators
17	Isguest login	1 if the login is a "guest" login; 0 Otherwise
18	Ishost login	1 if the login belongs to the "host"
19	Land	1 if connection is from/to the samehost/port; 0 otherwise
20	logged in	1 if successfully logged in; 0 otherwise
21	Numaccess files	Number of operations on access control files
22	Numcompromised	Number of "compromised" conditions
23	Numfailed logins	Number of failed logins
24	Numfile creations	Number of file creation operations
25	Num outbound cmds	Number of outbound commands in an ftp session
26	Num root	Number of "root" accesses
27	Num shells	Number of shell prompts
28	Protocol type	Protocol type
29	Rerror rate	% of connections that have "REJ" Errors
30	Root shell	1 if root shell is obtained; 0 otherwise

31	Same srv rate	% of connections to the same service
32	Serror rate	% of connections that have "SYN" Errors
33	Service	Destination service (e.g. telnet, ftp)
34	Src bytes	Bytes sent from source to destination
35	Srv count	Number of connections to the same service as the current connection in the past two seconds
36	Srv_diff host rate	% of connections to different hosts
37	Srv error rate	% of connections that have "REJ" errors
38	Srv serror rate	% of connections that have "SYN" Errors
39	Su attempted	1 if "su root" command attempted; 0 otherwise
40	Urgent	Number of urgent packets
41	Wrong_fragment	Number of wrong fragments

Table (3.2): KDD'99 Features Description. (Sathya, Ramani, and Sivaselvi, 2011)

Chapter Four

Proposed Model & Methodology

4.1 Overview

This chapter will clarify the methodology of this research and the proposed model structure in details. It will also clarify how to implement the system and how to get the results.

4.2 Methodology

This thesis deals with the enhanced Steady State Genetic Algorithm (SSGA) for Intrusion Detection (ID), through applying the best method of replacement by comparing replacement methods: Binary Tournament Replacement and Triple Tournament Replacement, then choose the best one, we was hopping to improve the previous results.

The environment will be the KDD Cup 99 dataset and we will use 10% of the environment as a population sample. The system uses two datasets; training dataset which is 9% and testing dataset which is 1% of KDD Cup 99.

The proposed model contains three phases: classification phase, steady state Genetic Algorithm phase and Matching phase.

After using (SSGA), the resulted rules will be stored in the rules pool, to be used at the Matching phase to examine the testing dataset.

Finally, evaluating the system success, strength and trustworthy by calculating Detection Rate (DR) and False Positive Rate (FPR). The effective IDS must have high Detection Rate (DR) and low False Positive Rate (FPR).

4.3 Proposed Model

The following figure displays our own proposed model:

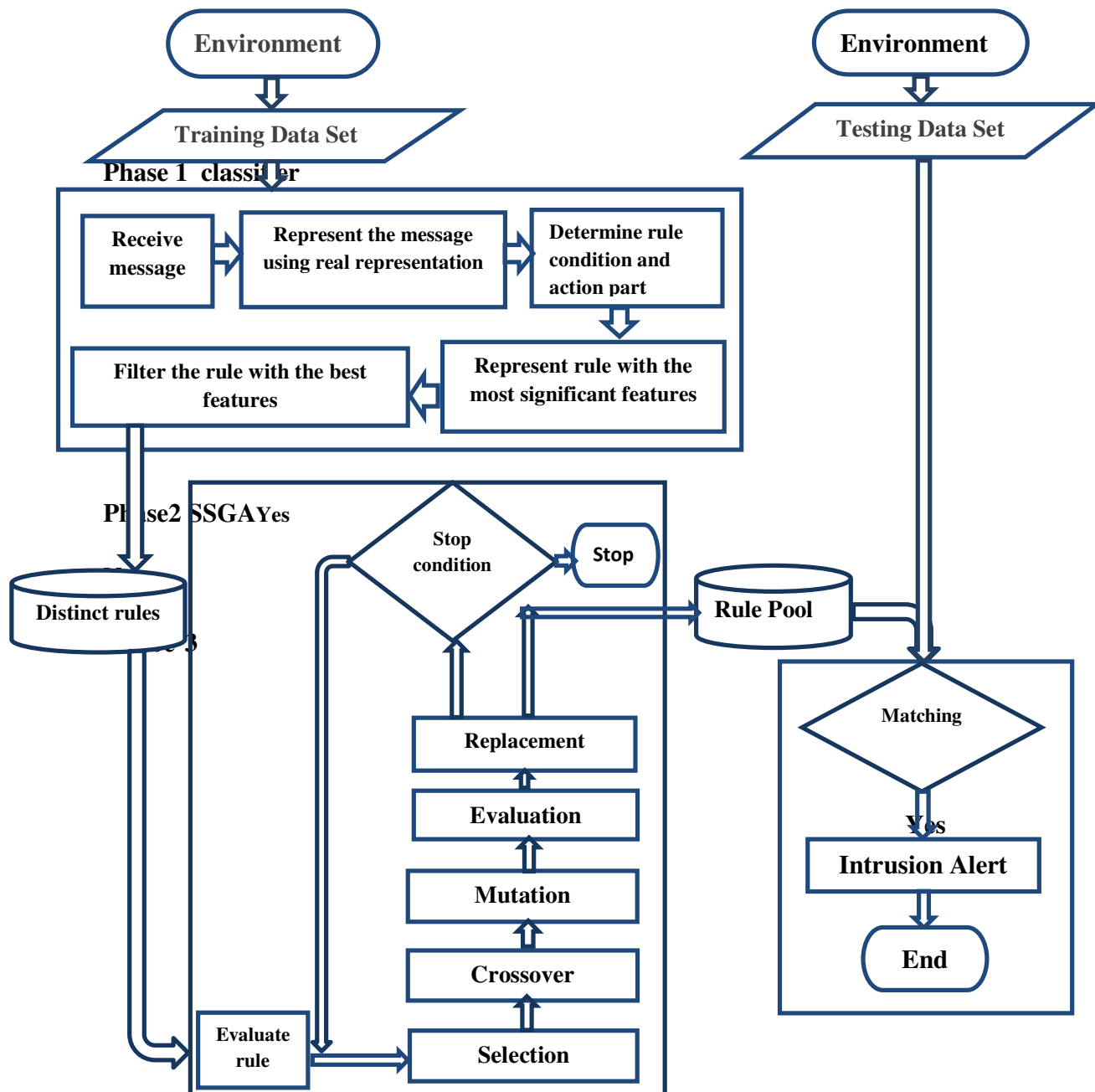


Figure (4.1)
Enhanced Steady State Genetic Algorithm Model for Intrusion Detection System

4.4 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 Elitist Selection.

Step (4): Apply Uniform Crossover.

Step (5): Apply Flip Bit Mutation.

Step (6): Evaluate the chromosomes.

Step (7): Apply Binary and Triple 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.

4.5 Proposed Model Structure:

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

4.5.1 Environment:

The KDD CUP 99 datasets will be the environment for implementing the proposed model. KDD CUP 99 has been built based on the data captured in DARPA'98 IDS evaluation program.

In this research 10% of original dataset will be used as a sample for training and testing, that is approximately 494,021 records.

4.5.1.1 Training Dataset:

The system will start receiving data from KDD Cup 99, in this research we used 9% as a training dataset, which is approximately 444618 records.

4.5.1.2 Testing Dataset:

Part of the dataset will be used to examine the matching between packet and the rules stored in the rule pool; KDD Cup 99 will be the environment for testing dataset, the researcher will use 1% as testing dataset, which is approximately 49403 records.

4.5.2 Classifier:

Classifier used for classifying the data; in this phase, the message that comes from training dataset will be received and then represent the message by using real representation to produce the rule as a chromosome which will be used later in determining the most significant features. The values of the most significant features are varied between binary and real numbers, so the real representation is the best, because it includes both types: binary and real numbers.

Also, the classifier will use a message to determine the condition and action part of the rule. The condition part of the rule has a combination of values related to set of features, those values of that features made the condition to cause such attack.

The relation between the condition values and the type of attack represents the action part, which means if those features have those values then the type of attack will be as it is in the feature number 42.

To represent a rule with the most significant features, there are many researches for finding the most significant features that are sufficient to recognize the type of attack, as mentioned in table (4.1).

The last part of classifier is to filter the message from redundancy.

4.5.3 Distinct Rules:

This database will be produced after the following operations:

- Classifying data to four classes (DoS, Probe, R2L and U2R).
- Creating the rules as Condition-Action form.
- Determining the rules with the most significant features.
- Removing the redundant rules from the rules dataset.

4.5.4 Steady state Genetic Algorithm Unit:

In this unit, genetic algorithm will be used to generate new chromosomes from the existing data. This phase will help in selecting the rules by replacing the poorest one, so the rules pool should contain the best elitist rules.

-Evaluation:

The chromosomes will be evaluated by computing Fitness value for each chromosome in order to be selected in the next stage.

This research will use Reward Penalty Fitness Function(3.9) proposed by (Alabsi and Naoum, 2012).

-Selection:

Selecting the appropriate individual can be done by using Elitist Selection.

Elitist Selection gets the best results when it used together with Uniform Crossover within Steady State Genetic Algorithm, depending on (Alabsi, 2012).

-Crossover:

At this stage, Uniform Crossover (UX) will be used, where genes are randomly exchanged at random points within a chromosome to produce two new offsprings.

-Mutation:

Applying mutation for each child produced from the last stage, Mutation will use flip bit, by flipping the value of a gene that was chosen randomly, to be equal to a random number of specific range.

-Evaluation:

This stage will be used to evaluate the generated chromosomes by using Reward Penalty Fitness Function. Evaluating the chromosomes at this stage helps in applying Replacement stage.

-Replacement:

Applying Replacement by comparing the Replacement methods, and then decide which one gives the best result.

-Check the Stopping Criteria:

Checking the stopping criteria can be done if there are no additional new rules to be produced, then the Genetic Algorithm will be stopped; otherwise, the Genetic Algorithm creates additional generation.

4.5.5 Rules Pool:

It will contain the rules that are collected from the training data and steady state Genetic Algorithm unit in order to be used in the testing phase.

4.5.6 Testing (Matching):

In this phase, the proposed system will try to match the received data with the existent rules in the rules pool, in order to recognize the data and detect the intrusions.

If it satisfied the matching condition, then the alarm of intrusion detection will be appeared, otherwise it is normal behavior.

4.5.7 System Evaluation:

Evaluating the proposed system will be done by calculating Detection Rate (DR) and False Positive Rate (FPR) then comparing the results with others.

4.6. Attack types and Features selection:

Attacks types are classified according to the following categories: (Tavallae, Bagheri, & Ghorbani 2009)

1. Denial of Service (DoS)
2. User to Root (U2R)
3. Remote to Local (R2L)
4. Probing

To determine each type of attacks, the feature selection issue must be taken in consideration.

There are many papers determined the most significant features responsible for identifying the type of attack. The following table presents the papers and the most significant features for each attack:

Attacks Papers	DoS	Probe	U2R	R2L
Mukkamala and Sung, (2003)	f1,f5,f6,f23,f24,f25 f26,f32,f36,f38,f39	f1,f2,f3,f4,f5,f6, f23,f24,f29,f32,f33	f1,f2,f3,f5,f6,f12, f23,f24,f32,f33	f1,f3,f5,f6,f32, f33
Chou, Yen, and Luo, (2008)	f1,f2,f3,f4,f5,f6,f12, f23,f24,f31,f32,f37	f1,f2,f3,f4,f12,f16, f25,f27,f28,f29,f30, f40	f1,f2,f3,f10,f16	f1,f2,f3,f4,f5,f10, f22
Zainal, Maarof, Shamsuddin, and Abraham, (2008)	f5,f10,f24,f29,f33, f34,f38,f40	f2,f3,f23,f34, f36,f40	f3,f4,f6,f14, f17,f22	f3,f4,f10,f23, f33,f36
Mukkamala, Sung and Abraham,(2004)	f7,f8,f12,f13,f23	f3,f12,f27,f31,f35	f14,f17,f25,f36,f38	f6,f11,f12,f19,f22

Table (4.1):Researches to determine the most significant features for each attack

In this research, the results ofMukkamala, Sung and Abraham,(2004)will be adopted in the stage of representing rules with the most significant features, because this research has been tested in many previous studies such as (Al-Sharafat, 2009) and (Alabsi, 2012), and it gives good results.

Chapter Five

Experimental Results

5.1 Overview

In this chapter, we present the experimental results through the execution of proposed Intrusion Detection System, which has been supported by Steady State Genetic Algorithm.

5.2 Dataset

All experiments and evaluations will be done over Knowledge Discovery in databases KDD Cup 99 dataset, by using Vb.Net 2010 and SQL server 2008. KDD Cup 99 includes a wide variety of intrusions, so it is described as the most widely used dataset in the field of IDSs evaluation.

The whole data of KDD Cup 99 is 4940210 records; it's available on the KDD official website. This research used 10% of the whole data as training and testing dataset, which means 494021 records were used in this research.

10% of the whole data have been divided into two groups 9% as training dataset which contains 444618 records (including 78416 normal and 366202 attacks) and 1% as testing dataset which contains 49403 records. Table (5.1) presents the distribution of attacks in the training dataset:

Attack	No. of Rows	Percentage
DoS	361243	98.64%
R2L	1125	0.31%
U2R	37	0.01%
Probe	3797	1.04%
Total	366202	100%

Table (5.1):Distribution of Attacks within Training Dataset

5.3 Classification:

We have been dealing with data by classifying them into four databases: DoS, R2L, U2R and Probe. Each database contains many tables of sub attack types, and then we have classified training dataset according to the attack type and sub attack types.

The training dataset is made up of 22 different sub attack types for the known attack categories .The following table shows the sub attack types that have been found within(9%) training dataset:

attack	Sub attack types
DoS	Back, land, Neptune, pod, smurf, teardrop
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
U2R	Buffer_overflow, loadmodule, perl, rootkit
Probe	Ipsweep, nmap, portsweep, satan

Table (5.2):Sub attack types in training dataset

5.4 Features selection:

The training dataset contains 444618 records. Each record has 41 features with feature number 42 which determines the type of attack.

To judge that the record of testing dataset belongs to a particular classification of attack, the tested record must have the same features values as at least one record of the training dataset. The process of testing record with the whole data in the training dataset is achieved by comparing 41 features. To make this process easier with less time; we depend on the most significant features that determine the type of attack without needing to the other features.

In this stage, Mukkamala, Sung and Abraham, (2004) have been adopted to represent record with the most significant features; they tried to remove the useless features to increase the accuracy until they got five significant features for each record, as the following table:

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

Table (5.3): Selected Features by Mukkamala, Sung and Abraham

After selecting five features for each record, many duplicate records will be appeared; no need to use the duplicate records. So we tried to filter data and eliminate the duplicate records.

In the following tables, we show the number of records before and after filtering for each category of attack:

Sub attack types	No. of records before filtering	No. of records after filtering	No. of repeated records
Back	2103	22	2081
Land	18	2	16
Neptune	86031	302	85729
Pod	242	22	220
Smurf	271970	351	271619
Teardrop	879	202	677
Total records	361243	901	360342

Table (5.4): Number of records before and after filtering for DoS attack

Sub attack types	No. of records before filtering	No. of records after filtering	No. of repeated records
ftp_write	8	8	0
Guess_passwd	53	3	50
Imap	12	4	8
Multihop	7	7	0
Phf	3	1	2
Spy	2	2	0
Warezclient	1020	26	994
Warezmaster	20	20	0
Total records	1125	71	1054

Table (5.5): Number of records before and after filtering for R2L attack

Sub attack types	No. of records before filtering	No. of records after filtering	No. of repeated records
Buffer_overflow	17	8	9
Loadmodule	9	6	3
Perl	3	2	1
Rootkit	8	4	4
Total records	37	20	17

Table (5.6): Number of records before and after filtering for U2R attack

Sub attack types	No. of records before filtering	No. of records after filtering	No. of repeated records
Ipsweep	1118	24	1094
Nmap	231	33	198
Portsweep	860	183	677
Satan	1588	256	1332
Total records	3797	496	3301

Table (5.7): Number of records before and after filtering for Probe attack

5.5 Fitness Function:

To apply steady state genetic algorithm each chromosome must be evaluated by computing its Fitness value. Fitness value evaluates the quality of each chromosome, so the high fitness value will give chromosome high probability to be selected in the selection stage. Reward Penalty Fitness Function has been used to determine fitness value for each record as the following equation:

$$\text{Fitness} = 2 + \frac{AB - A}{AB + A} + \frac{AB}{X} - \frac{A}{Y}$$

The values of A and AB depend on the condition and action parts, when the condition and action of the selected record are equal to the condition and action of the Compared record then the value of AB of the selected record will be increased by one, else the value of A of the selected record will be increased by one.

X = the maximum value of AB in the population.

Y = the maximum value of A in the population.

5.5.1 Why Fitness Function after using SSGA is equal to A:

Reward Penalty based Fitness Function has been built to evaluate each chromosome in the population depending on four different values (A value, AB value, Maximum A value in the population and Maximum AB value in the population). So, the chromosome will be strong if it has a high value of AB, and it will be stronger if its AB value is more close to the maximum AB value in the population and it will be the strongest chromosome if it has the maximum AB value in the population.

On the other side, the chromosome will be weak if it has a high value of A, it will be more weakness if its A value is more close to the maximum A value in the population and it will be the weakest chromosome if it has the maximum A value in the population.

As mentioned before, we depend on the value of AB which means if there is a chromosome with a set of genes value, how much this chromosome has the same action in a set of data which is called Training dataset.

But after using Genetic Algorithm, there are a huge number of chromosomes that is created but its AB value is equal to Zero. So, we have some cases that Fitness Value is equal to (NULL) that means the value is not accepted mathematically.

So, this Fitness Function does not fit to be used in evaluating a chromosome created by GA and reapplying GA over it, but it will be fit if we apply GA many times on the same original population, without creating a new chromosome by applying GA on a created chromosome.

This problem must be solved in our case, because we must evaluate created chromosomes before applying the Replacement phase.

The problem has been solved by just depending on A value. If it is equal to zero, that means the generated chromosome is completely new but if there is A value which is more than 0, this will mean there are two cases, the first case is that the created chromosome is same to the rules which stored in Rule pool, that means, we don't need this created chromosome anymore.

The second case is that the created chromosome is the same to the rule of another attack which means that this created chromosome will not help us in detection.

After determining A value for each record, the Replacement phase has been applied. By using Triple Replacement, we compare three generations and take just the chromosome with A values equal zero or approach to zero. For example, if we have a population with size = 300, we have three generations, each one generates another 300 chromosomes and the result is 900 chromosomes, we just take the best 300 chromosomes of 900 chromosomes.

5.6 Tracking the genetic algorithm by values:

We try to track genetic algorithm processes with values as an example. Since there are a large number of records in each type of attack, we will track GA processes on U2R attack which contains the least number of record (20 records) as mentioned in tables (5.6).

After determining the fitness value by using Reward Penalty Fitness Function for each record with the first generation, we get the following values:

ID	Fitness value	ID	Fitness value
1	0.75	11	3.33
2	1.917	12	3.33
3	1.75	13	1.667
4	3.5	14	2.667
5	3.25	15	4
6	3.33	16	3
7	3.75	17	1.9
8	3.5	18	0.25
9	1.33	19	1.368
10	1	20	3.5

Table (5.8):fitness value for each record in U2R

Notice from the table above that the highest value for fitness is 4 of the record number 15, and the lowest value for fitness is 0.25 of the record number 18. We will use these values in the selection stage.

After elitist selection process, we get the following table:

ID	Fitness value	ID	Fitness value
15	4	14	2.66
7	3.75	2	1.917
4	3.5	17	1.901
8	3.5	3	1.75
20	3.5	13	1.667
6	3.33	19	1.368
11	3.33	9	1.33
12	3.33	10	1
5	3.25	1	0.75
16	3	18	0.25

Table (5.9): the values after selection process in U2R

Now, uniform crossover will be applied, where features are randomly exchanged at random points within a chromosome to produce two new records.

The features of each record after crossover as the following values:

No.	Features values	No.	Features values
1	1 2 0 0.5 0	11	1 1 0 1 0
2	1 1 0 0 0	12	0 4 0 0.5 0
3	0 4 0 1 0	13	0 1 0 0 0
4	1 1 0 1 0	14	0 0 0 1 0
5	1 1 0 0 0	15	1 1 0 1 0
6	1 0 0 1 0	16	0 0 0 0 0
7	0 1 0 0.2 0	17	0 0 0 0.17 0
8	0 4 0 0.25 0	18	0 0 0 1 0
9	1 2 0 0.5 0	19	0 0 0 1 0
10	1 0 0 0.01 0	20	0 0 0 1 0

Table (5.10): records form of U2R after crossover

On the next stage, mutation has been applied on the first chromosome with feature number three. But the chromosome stayed with the same form because all the values of the feature number three are the same and equal to zero.

Another mutation has been applied on the chromosome number eleven with feature number two. The form of the chromosome before mutation was 1 1 0 1 0, after mutation became 1 2 0 1 0.

Now, we need to evaluate each chromosome before replacement process depending on the value of A. The following table shows the value of A for each record:

No.	A value	No.	A value
1	0	11	0
2	2	12	0
3	2	13	17
4	2	14	277273
5	2	15	2
6	6	16	27424
7	1	17	891
8	1	18	277273
9	0	19	277273
10	4	20	277273

Table (5.11): the values of A for each record after mutation

From the table above, notice that there is a difference between the values of A.

The record with the value of A equal to zero, is stronger than others, and vice versa, the record with high value of A is weaker than others.

Now, in the second generation after applying crossover operation, we get the following values:

No.	Features values	No.	Features values
1	1 1 0 0 0	11	1 1 0 0.5 0
2	1 2 0 0.5 0	12	0 4 0 1 0
3	0 4 0 1 0	13	0 0 0 0 0
4	1 1 0 1 0	14	0 1 0 1 0
5	1 0 0 0 0	15	1 0 0 0 0
6	1 1 0 1 0	16	0 1 0 1 0
7	0 1 0 0.2 0	17	0 0 0 1 0
8	0 4 0 0.25 0	18	0 0 0 0.17 0
9	1 0 0 0.01 0	19	0 0 0 1 0
10	1 2 0 0.5 0	20	0 0 0 1 0

Table (5.12): Features values of U2R after crossover on the second generation

Mutation has been applied on the first chromosome with feature number two. The form of chromosome before mutation was 1 1 0 0 0, after mutation became 1 2 0 0 0. Another mutation has been applied on the chromosome number eleven on the first feature. The form of chromosome before mutation was 1 1 0 0.5 0, after mutation became 0 1 0 0.5 0.

Evaluating each chromosome depending on the value of A, as the following table:

No.	A value	No.	A value
1	2	11	3
2	0	12	2
3	2	13	27424
4	2	14	4
5	8	15	8
6	2	16	4
7	1	17	277273
8	1	18	891
9	4	19	277273
10	0	20	277273

Table (5.13): the values of A for each record after mutation in the second generation

Now, in the third generation after crossover operation, we get the following values:

No.	Features values	No.	Features values
1	1 2 0 0.5 0	11	1 1 0 0.5 0
2	1 10 00	12	0 4 0 1 0
3	0 4 0 1 0	13	0 1 0 0 0
4	1 1 0 1 0	14	0 0 0 1 0
5	1 1 0 0 0	15	1 0 0 0 0
6	1 0 0 1 0	16	0 1 0 1 0
7	0 1 0 0.2 0	17	0 0 0 0.17 0
8	0 4 0 0.25 0	18	0 0 0 10
9	1 2 0 0.5 0	19	0 0 0 1 0
10	1 0 0 0.01 0	20	0 0 0 1 0

Table (5.14): Features values of U2R after crossover on the third generation

Mutation has been applied on the first chromosome with feature number five, but there is no change on the feature values; because all the values of this feature equal zero. Another mutation has been applied on the chromosome number eleven on the feature number four. The form of the chromosome before mutation was 1 1 0 0.5 0, after mutation became 1 1 0 0.2 0.

Evaluating each chromosome depending on the value of A, The following table shows the values of A of each record:

No.	A value	No.	A value
1	0	11	0
2	2	12	2
3	2	13	17
4	2	14	277273
5	2	15	8
6	6	16	4
7	1	17	891
8	1	18	277273
9	0	19	277273
10	4	20	277273

Table (5.15): the values of A for each record after mutation in the third generation

The last process of GA is the Replacement; triple replacement was applied every three generations according to the values of A.

So, after triple replacement process we will get the following chromosomes:

No.	F1	F2	F3	F4	F5
1	1	2	0	0.5	0
2	1	2	0	0.5	0
3	1	1	0	1	0
4	0	4	0	0.5	0
5	1	2	0	0.5	0
6	1	2	0	0.5	0
7	1	2	0	0.5	0
8	1	2	0	0.5	0
9	1	1	0	0.2	0
10	0	1	0	0.2	0
11	0	4	0	0.25	0
12	0	1	0	0.2	0
13	0	4	0	0.25	0
14	0	1	0	0.2	0
15	0	4	0	0.25	0
16	1	1	0	0	0
17	0	4	0	1	0
18	1	1	0	1	0
19	1	1	0	0	0
20	1	1	0	1	0

Table (5.16): Chromosomes value after triple replacement process

Actually from the replacement results, we got 8 distinct chromosomes after eliminate repeated records to be saved in the rule pool.

This has been done for the first three generations. Other generations will be proceeding until achieving the stopping criteria.

Stopping criteria of genetic algorithm will be done when there are no additional new records to be produced.

5.7 The convergence:

The convergence to an accurate solution refers to the procedures that are followed to access of the solution for specific problem. That means, you may have different solutions but one of those solutions is the optimal, according to specific criterion.

- In this research, we notice that if there are two cases with detection rate equal 100% as in probe attack type, but the first case with rule pool which has 1000 chromosomes and the second case with rule pool which has 2000 chromosomes then the first case has a faster convergence, by comparing the requested time for two cases.

- Stopping Criterion can be considered as an indicator to GA convergence. In our research, when there are no additional new rules to be produced, then the Genetic Algorithm will be stopped. Otherwise, the Genetic Algorithm creates an additional generation.

That means, the convergence happened when the difference between the number of records in generation (n) and the number of records in generation (n+1) approaching to zero.

5.8 The Results of Detection Rate (DR) and False Positive Rate (FPR):

As a result of this research, Intrusion Detection System has been built and supported with Steady State Genetic Algorithm. Detection Rate has been calculated by using equation (3.1) and False Positive Rate was calculated by using equation (3.2).

The goal is to get high DR and low FPR. After system execution, we get the following results of DR and FPR with Double Replacement for each type of attack:

	DoS	Probe	U2R	R2L
DR	97%	100%	40%	100%
FPR	0.0232	1.84	1.866	2.195

Table (5.17): Results of DR and FPR with Double Replacement

And the results of DR and FPR with Triple Replacement for each type of attack:

	DoS	Probe	U2R	R2L
DR	100%	100%	53%	100%
FPR	0.0232	1.84	1.866	2.195

Table (5.18): Results of DR and FPR with Triple Replacement

Thus we conclude that:

- Triple replacement method produced more accurate results in DR than double replacement.
- Both of Double and Triple Replacement have the same values of FPR, but they are different in the value of DR.
- The increasing in the number of generations leads to increase DR, but the values of FPR are not affected with this increasing.

5.9 Tracking increasing of DR in U2R attack with Double and Triple Replacement:

Our results in tables (5.17) and (5.18) show that the Triple Replacement has an effect on the SSGA and lead to an optimal solution by comparing it with double replacement, this comparison is clear with DR of U2R attack.

To explain that, from our results we found that SSGA with double replacement in the first 100 generations produced 240 records, but with triple replacement produced 177 records.

Generation	No. of record with double	No. of record with triple
10	28	16
20	46	24
30	87	44
40	138	71
50	180	87
60	232	106
70	235	126
80	237	134
90	238	155
100	240	177

Table (5.19): number of records in the first 100 generations of U2R attack

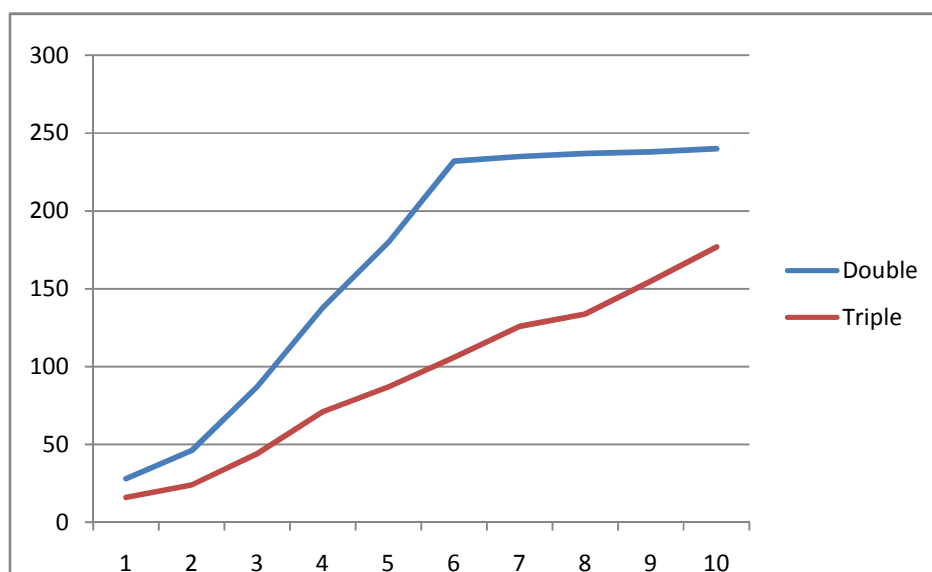


Figure (5.1) the difference between double and triple replacement in the first 100 generation

While, SSGA with triple replacement until generation number 600 produced 1096 records, but with double replacement produced 288 records as shown in the following table:

generation	No. of record with double	No. of record with triple
100	260	197
200	267	382
300	273	557
400	281	750
500	285	920
600	288	1096

Table(5.20): number of records after the 100 generations of U2R attack

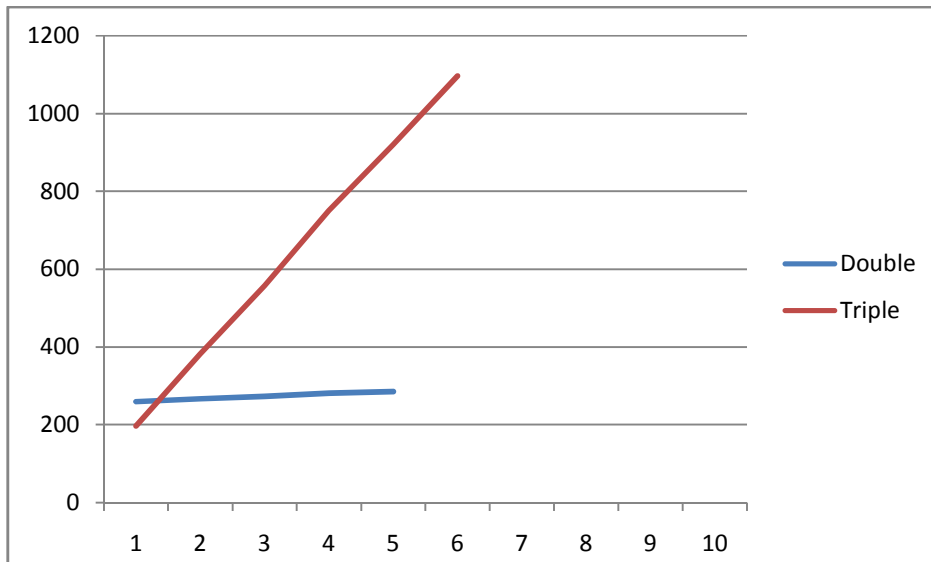


Figure (5.2) the difference between double and triple replacement after the 100 generations

From figures (5.1) and (5.2), we note that X-axes present the generations and Y-axes present the number of records. Our results showed that at the beginning generations, SSGA with double replacement produce more chromosomes than SSGA with triple replacement, and this is because double replacement saves a new chromosome at the rule

pool just once a time for each two generations and triple replacement saves a new chromosome at the rule pool just once a time for each three generations.

After 100 generation, double replacement stopped from producing new chromosomes, but triple replacement continues to produce them and improve the value of DR in the system.

5.10 Comparing thesis results with other results:

The first comparing procedure will be between our results when we use Triple Replacement, and (Alabsi, 2012) results he used double replacement in Genetic Algorithm with Misuse Intrusion Detection System. But in Alabsi thesis it has been used 5% as a sample of training dataset, while our research used 9% as a sample of training dataset.

The criterion of comparing is DR of each attack, as the following table:

	DoS	Probe	U2R	R2L
Our research	100%	100%	53%	100%
Alabsi	94%	100%	86%	100%

Table (5.21): First comparison

From table (5.21), we note that the DR of U2R attack in our research is very low comparing with other attacks. To explain the reason of this case, since the number of records of U2R attack is 20 records in training dataset and 15 records in testing dataset. The question here: Are the 20 records that distributed on 9% of dataset being able to

detect 15 records distributed on 1% of dataset? Then, distribution of the sample leads to decrease the value of DR of U2R.

While, for example with DoS attack there are multiple numbers of records distributed on training dataset that be able to detect small number of records distributed on testing dataset. Then, the value of DR of DoS was high.

The second comparing procedure will be between our results with Stewart and alsharafat result. The criteria of this comparison will be the average of DR and FPR.

(Stewart, L. 2009) found that the average of DR is equal to 79.67%, and FPR is 2.69%, which is worse than the results of our research.

(Al-Sharafat, 2009) got the average of DR equal to 98.9%, and the average of FPR equal to 0.094% which is better than the results of this research.

The following table shows these results:

	Average of DR	Average of FPR
Our result	88.25%	1.48%
Alsharafat	98.9%	0.094%
Stewart	79.67%	2.69%

Table (5.22): Second comparison

Stewart used hybrid system of Genetic Algorithm with Neural Networks for Intrusion Detection System. While Al-Sharafat used Steady State Genetic Algorithm with Anomaly based IDS.

From this comparison we note that this research achieved better results than (stewart, 2009), but worse results than (alsharafat, 2009).

Chapter six

Conclusion and Future Work

6.1 Conclusion:

In this thesis an intrusion detection system was built and we focus on enhancing the replacement of steady state genetic algorithm to increase the detection rate.

The environment of this research was the KDD Cup 99 dataset. All experiments and evaluations are performed by using 10% of the whole dataset. The inputs to the system are two subsets; training dataset which is 9% and testing dataset which is 1% of KDD Cup 99 dataset.

We conclude from our research that by using the replacement, it enhanced the SSGA by comparing Replacement methods. Also, we found that triple replacement produced more accurate results than double replacement, according to the value of DR, number of generations and the number of new chromosomes.

Also, we found that Triple Replacement produced DR is equal to 100% for the following attack types (DoS, Probe, and R2L), but U2R attack produced a result of DR equal to 53% which is still better than U2R in Double Replacement which produced DR equal to 40%.

This research improves Fitness Function to be fit for finding a Fitness Value that is proper for new chromosomes that produced from GA.

Also, we found that Triple Replacement enhanced the convergence to the solution and improved the efficiency of SSGA for producing new chromosomes.

6.2 Future Work:

1. Genetic Algorithm can be applied in two ways: the first way is applying GA on the main attack types such as DoS, R2L, U2R and Probe. The second one is applying GA on each sub attack types. To determine which way is the best, additional studies must be done to know which one is better.
2. Additional researches about selecting the best rate of mutation. The used mutation rate is 0.1, thus we need to develop an algorithm to find optimal rate.
3. Need more studies on how to keep higher DR and lower FPR with Misuse detection.
4. Need further researches to choose the appropriate population size and determine which is the best size of population, big or small? What is the difference between them?
5. Additional researches must be done to find the complicity of steady state genetic algorithm with a different type of replacement such as random replacement.
6. Need more researches and studies about fitness function to reach the formula gives more accurate results.
7. Using hybrid algorithm which adopt a neural computing with GA as a procedure for finding optimal solution for Intrusion Detection.

References:

- 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 chennai, india. Available at: <http://airccj.org/CSCP/vol1/csit1226.pdf>
- Alabsi, F., Naoum, R., (2012), Comparison of Selection Methods and Crossover Operations using Steady State Genetic Based Intrusion Detection System, *Journal of Emerging Trends in Computing and Information Sciences*. VOL. 3, NO.7, ISSN 2079-8407
- 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.
- Alabsi, F. (2012), *An Enhanced Steady State Genetic Algorithm Model for Misuse Network Intrusion Detection System*, (master thesis). Middle East University, Amman, Jordan.
- Al-Rashdan, W., (2011). *A Hybrid Artificial Neural Network Model for effective network intrusion detection system*, (doctorate dissertation) , The Arab Academy for banking and financial sciences, Amman, Jordan.
- Al-Sabbah, A. A. (2012). *The Color Image Enhancement Using SSGA Steady State Genetic Algorithm*, (master dissertation). Middle East University, Amman, Jordan.
- Al-Sharafat, W.S. (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.
- Ashoor, A., Gore, S., (2011), Importance of Intrusion Detection System (IDS), *International Journal of Scientific & Engineering Research*, Vol. 2, Issue 1, ISSN 2229-5518.
- Bishop, M. (2005). *Introduction to Computer Security*. Boston: Pearson Education.
- Cesare, S. & Xiang, Y., (2010), Classification of Malware Using Structured Control Flow, *Proceeding in Eighth Australasian Symp, Parallel and Distributed Computing*.
- Chau, D., Nachenberg, C., Wilhelm, J., Wright, A., Faloutsos, C., (2010), Polonium: Tera-Scale Graph Mining for Malware Detection, *Journal of ACM digital library*.

- 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).
- DARPA 1998 data set,
http://www.ll.mit.edu/IST/ideval/data/1998/1998_data_index.html, cited August 2003.
- Fakeih, A. & Kattan, A. (2012). Recurrent Genetic Algorithm Sustaining Evolvability. *Journal of Springer Link*, (230-242).
- Gadbois, P., (2011). "Train Signal's CompTIA Security+ Intrusion Detectio", YouTube, (On Line) , available: <http://www.youtube.com/watch?v=O2Gz-v8WswQ>
- 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.
- Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.
- Goyal, A. & Kumar, C. (2009). GA-NIDS: A Genetic Algorithm based Network Intrusion Detection System, *Journal of Elsevier*.
- Haupt, R., Haupt, S., (2004). *Practical Genetic Algorithm*, published by John Wiley, available at:
<https://os.cloudme.com/v1/webshares/12885457505/CloudComputing/Cloud%20Computing/Artificial%20Genetic%20Algorithms/Practical%20Genetic%20Algorithms%20-%20Randy%20L.%20Haupt,%20Sue%20Ellen%20Haupt.pdf>
- Hoque M. S., Mukit A. & Bikas A. (2012). An Implementation of Intrusion Detection System Using Genetic Algorithm, *International Journal of Network Security and its applications*, Vol.4, NO.2, 109-120.
- Jong, G. J., Chen, S. M., Su, T. J., & Horng, G. J. (2005). A combined LMS with RGA algorithm of the co-channel separation system. In *Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems. ISPACS 2005*, 285-288, IEEE.
- 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>

- KDD-CUP 1999 Data, Available at:
<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- Kshirsagar, V. K., Tidke, S. M. & Vishnu, S. (2012). Intrusion Detection System using Genetic Algorithm and Data Mining: An Overview. *International Journal of Computer Science and Informatics* ISSN (PRINT), 2231-5292.
- Kuang, L. (2007). DNIDS: A dependable network intrusion detection system using the CSI-KNN algorithm.
- 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>
- Kurose, J. & Ross, K. (2010). *Computer Networking A Top-Down Approach* (5thed.). Boston: Pearson Education.
- Lahre, K., Diwan, T., Kumar, S. Agrawal,P., (2013).Analyze Different approaches for IDS using KDD 99 Data Set. *International Journal on Recent and Innovation Trends in Computing and Communication*, Volume: 1 Issue: 8, ISSN 2321 – 8169, pages 645 – 651.
- Li, W. (2004). Using genetic algorithm for network intrusion detection. *Proceedings of the United States Department of Energy Cyber Security Group*, 1-8.
- Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press. Cambridge, Massachusetts. London, England.
- Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1), 42-57.
- Mostaque Md.,(2013). Network intrusion detection system using genetic algorithm and fuzzy logic.*International Journal of Innovative Research in Computer and Communication Engineering.(An ISO 3297: 2007 Certified Organization)* Vol. 1, Issue 7.
- Mukkamala, S. & Sung, A (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
- 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).
- Ramakrishnan, S. & Srinivasan, S. (2009). Intelligent agent based artificial immune system for computer security—a review. *Artificial Intelligence Review*, 32(1-4), 13-43.
- Reeves, C., (2000), *Genetic Algorithms*, third chapter, School of Mathematical and Information Sciences, available at:
<http://sci2s.ugr.es/docencia/metah/bibliografia/GeneticAlgorithms.pdf>,
- Rehman, R. U. (2003). *Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID*. Prentice Hall Professional, at:
<http://ptgmedia.pearsoncmg.com/images/0131407333/downloads/0131407333.pdf>.
- Sabhnani, M. and Serpen, G., (2004). Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set. *Journal of Intelligent Data Analysis*, Volume 8 Issue 4, pages 403-415.
- Sathya, S., Ramani, R., Sivaselvi, K., (2011) .Discriminant Analysis based Feature Selection in KDD Intrusion Dataset, *International Journal of Computer Applications*, (0975 – 8887) Volume 31– No.11
- (SysAdmin, Audit, Network, Security) Institute, (2001). Intrusion detection systems; definition, need and challenge. Available at:
http://www.sans.org/reading_room/whitepapers/detection/intrusion-detection-systems-definition-challenges_343
- 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>
- Schmidit, M. & Stidsen, T. (1997). *Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms*. Kluwer Academic Publisher.
- Selvakani, S., Rajesh, R., (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.
- Siddiqui, M., Naahid, S., (2013). Analysis of KDD CUP 99 Dataset using Clustering based Data Mining, *International Journal of Database Theory and*

Application, Vol.6 No.5, available at:
http://www.sersc.org/journals/IJDTA/vol6_no5/3.pdf.

- Singh, S. (2013). Intrusion detection system (IDS) and intrusion prevention system (IPS) for network security: a critical analysis. *International Journal of Research in Engineering & Applied Sciences*. Volume 3, Issue 3 ISSN: 2249-3905
- Siva, M. V., Vinay A. B & Babu, K. R. (2013). An intrusion detection system architecture based on neural networks and genetic algorithms. *International Journal of Computer Science and Management Research*. Vol 2 Issue 1, ISSN 2278-733X.
- 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_MS_c.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>
- Jaiganesh, V., Mangayarkarasi, S. & Dr. Sumathi, P. (2013). Intrusion Detection Systems: A Survey and Analysis of Classification Techniques. *International Journal of Advanced Research in Computer and Communication Engineering* Vol. 2, Issue 4, ISSN(Online): 2320-9801 ok
- Ye, Y., Li, T., Chen, Y., (2010), Automatic Malware Categorization Using Cluster Ensemble, *Journal of ACM*.
- 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 calculating A and AB

```

- Private conTrain AsNewSqlConnection("Data Source=(local);Initial Catalog=KDD;Integrated Security=True")
- Private datrain AsNewSqlDataAdapter("Select * from training", conTrain)
- Private conU2R AsNewSqlConnection("Data Source=(local);Initial Catalog=S_U2R;Integrated Security=True")
- PrivateSub btnbuffer_overflow_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
  btnbuffer_overflow.Click
-
- Dim dabuffer_overflow AsNewSqlDataAdapter("Select * from buffer_overflow", conU2R)
- Dim ds AsNewDataSet
-   dabuffer_overflow.Fill(ds, "buffer_overflow")
-   datrain.Fill(ds, "training")
-
- Dim f14, f17, f25, f36, f38 AsDouble
- Dim A, AB, id AsInteger
- Dim NoRowsTablekddcup AsInteger
- Dim NoRowsTablebuffer_overflow AsInteger
- Dim Attack AsInteger
-   NoRowsTablekddcup = ds.Tables("training").Rows.Count
-   NoRowsTablebuffer_overflow = ds.Tables("buffer_overflow").Rows.Count
-
- For Attack = 0 To NoRowsTablebuffer_overflow - 1
-   id = ds.Tables("buffer_overflow").Rows(Attack).Item(0)
-   f14 = ds.Tables("buffer_overflow").Rows(Attack).Item(1)
-   f17 = ds.Tables("buffer_overflow").Rows(Attack).Item(2)
-   f25 = ds.Tables("buffer_overflow").Rows(Attack).Item(3)
-   f36 = ds.Tables("buffer_overflow").Rows(Attack).Item(4)
-   f38 = ds.Tables("buffer_overflow").Rows(Attack).Item(5)
-   A = ds.Tables("buffer_overflow").Rows(Attack).Item(6)
-   AB = ds.Tables("buffer_overflow").Rows(Attack).Item(7)
-
-   For KddCounter = 0 To NoRowsTablekddcup - 1
-     If f14 = ds.Tables("training").Rows(KddCounter).Item(9) Then
-     If f17 = ds.Tables("training").Rows(KddCounter).Item(10) Then
-     If f25 = ds.Tables("training").Rows(KddCounter).Item(14) Then
-     If f36 = ds.Tables("training").Rows(KddCounter).Item(18) Then
-     If f38 = ds.Tables("training").Rows(KddCounter).Item(19) Then
-     If ds.Tables("training").Rows(KddCounter).Item(21) = buffer_overflow."Then"
-       AB = AB + 1
-     Else
-       A = A + 1
-     EndIf
-   EndIf
- EndIf
- EndIf
- EndIf
- EndIf
- Next
- Dim Ocmd AsNew Data.SqlClient.SqlCommand
-   Ocmd.CommandType = CommandType.StoredProcedure
-   Ocmd.Connection = conU2R
-   Ocmd.Parameters.AddWithValue("@id", id)
-   Ocmd.Parameters.AddWithValue("@A", A)
-   Ocmd.Parameters.AddWithValue("@AB", AB)
-   Ocmd.CommandText = "updatebuffer_overflow"
- Try
-   conU2R.Open()
-   Ocmd.ExecuteNonQuery()
- Catch ex AsException
- EndTry
-   conU2R.Close()
- Next

```

```

-         txtbuffer_overflow.Text = "Done"
-     EndSub
-
-     PrivateSub btnloadmodule_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
-         Handles btnloadmodule.Click
-         Dim daloadmodule As New SqlDataAdapter("Select * from loadmodule", conU2R)
-         Dim ds As New DataSet
-         daloadmodule.Fill(ds, "loadmodule")
-         datrain.Fill(ds, "training")
-
-         Dim f14, f17, f25, f36, f38 As Double
-         Dim A, AB, id As Integer
-         Dim NoRowsTablekddcup As Integer
-         Dim NoRowsTableloadmodule As Integer
-         Dim Attack As Integer
-         NoRowsTablekddcup = ds.Tables("training").Rows.Count
-         NoRowsTableloadmodule = ds.Tables("loadmodule").Rows.Count
-
-         For Attack = 0 To NoRowsTableloadmodule - 1
-             id = ds.Tables("loadmodule").Rows(Attack).Item(0)
-             f14 = ds.Tables("loadmodule").Rows(Attack).Item(1)
-             f17 = ds.Tables("loadmodule").Rows(Attack).Item(2)
-             f25 = ds.Tables("loadmodule").Rows(Attack).Item(3)
-             f36 = ds.Tables("loadmodule").Rows(Attack).Item(4)
-             f38 = ds.Tables("loadmodule").Rows(Attack).Item(5)
-             A = ds.Tables("loadmodule").Rows(Attack).Item(6)
-             AB = ds.Tables("loadmodule").Rows(Attack).Item(7)
-
-             For KddCounter = 0 To NoRowsTablekddcup - 1
-                 If f14 = ds.Tables("training").Rows(KddCounter).Item(9) Then
-                 If f17 = ds.Tables("training").Rows(KddCounter).Item(10) Then
-                 If f25 = ds.Tables("training").Rows(KddCounter).Item(14) Then
-                 If f36 = ds.Tables("training").Rows(KddCounter).Item(18) Then
-                 If f38 = ds.Tables("training").Rows(KddCounter).Item(19) Then
-                 If ds.Tables("training").Rows(KddCounter).Item(21) = "loadmodule." Then
-                     AB = AB + 1
-                 Else
-                     A = A + 1
-                 EndIf
-                 EndIf
-                 EndIf
-                 EndIf
-                 EndIf
-                 EndIf
-                 Next
-                 Dim Ocmd As New Data.SqlClient.SqlCommand
-                 Ocmd.CommandType = CommandType.StoredProcedure
-                 Ocmd.Connection = conU2R
-                 Ocmd.Parameters.AddWithValue("@id", id)
-                 Ocmd.Parameters.AddWithValue("@A", A)
-                 Ocmd.Parameters.AddWithValue("@AB", AB)
-                 Ocmd.CommandText = "updateloadmodule"
-
-                 Try
-                     conU2R.Open()
-                     Ocmd.ExecuteNonQuery()
-                 Catch ex As Exception
-                 EndTry
-                 conU2R.Close()
-                 Next
-                 txtloadmodule.Text = "Done"
-             EndSub
-
-             PrivateSub btnperl_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnperl.Click
-                 Dim daperl As New SqlDataAdapter("Select * from perl", conU2R)
-                 Dim ds As New DataSet

```

```

-         daperl.Fill(ds, "perl")
-         datrain.Fill(ds, "training")
-
-     Dim f14, f17, f25, f36, f38 AsDouble
-     Dim A, AB, id AsInteger
-     Dim NoRowsTablekddcup AsInteger
-     Dim NoRowsTableperl AsInteger
-     Dim Attack AsInteger
-     NoRowsTablekddcup = ds.Tables("training").Rows.Count
-     NoRowsTableperl = ds.Tables("perl").Rows.Count
-
-     For Attack = 0 To NoRowsTableperl - 1
-         id = ds.Tables("perl").Rows(Attack).Item(0)
-         f14 = ds.Tables("perl").Rows(Attack).Item(1)
-         f17 = ds.Tables("perl").Rows(Attack).Item(2)
-         f25 = ds.Tables("perl").Rows(Attack).Item(3)
-         f36 = ds.Tables("perl").Rows(Attack).Item(4)
-         f38 = ds.Tables("perl").Rows(Attack).Item(5)
-         A = ds.Tables("perl").Rows(Attack).Item(6)
-         AB = ds.Tables("perl").Rows(Attack).Item(7)
-
-     For KddCounter = 0 To NoRowsTablekddcup - 1
-     If f14 = ds.Tables("training").Rows(KddCounter).Item(9) Then
-     If f17 = ds.Tables("training").Rows(KddCounter).Item(10) Then
-     If f25 = ds.Tables("training").Rows(KddCounter).Item(14) Then
-     If f36 = ds.Tables("training").Rows(KddCounter).Item(18) Then
-     If f38 = ds.Tables("training").Rows(KddCounter).Item(19) Then
-     If ds.Tables("training").Rows(KddCounter).Item(21) = "perl." Then
-         AB = AB + 1
-     Else
-         A = A + 1
-     EndIf
-     EndIf
-     EndIf
-     EndIf
-     EndIf
-     Next
-     Dim Ocmd AsNew Data.SqlClient.SqlCommand
-     Ocmd.CommandType = CommandType.StoredProcedure
-     Ocmd.Connection = conU2R
-     Ocmd.Parameters.AddWithValue("@id", id)
-     Ocmd.Parameters.AddWithValue("@A", A)
-     Ocmd.Parameters.AddWithValue("@AB", AB)
-     Ocmd.CommandText = "updateperl"
-
-     Try
-         conU2R.Open()
-         Ocmd.ExecuteNonQuery()
-     Catch ex AsException
-     EndTry
-     conU2R.Close()
-     Next
-     txtperl.Text = "Done"
- EndSub
-
- PrivateSub btnrootkit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
- btnrootkit.Click
-     Dim darootkit AsNew SqlDataAdapter("Select * from rootkit", conU2R)
-     Dim ds AsNew DataSet
-     darootkit.Fill(ds, "rootkit")
-     datrain.Fill(ds, "training")
-
-     Dim f14, f17, f25, f36, f38 AsDouble
-     Dim A, AB, id AsInteger

```

```

- Dim NoRowsTablekddcup AsInteger
- Dim NoRowsTablerootkit AsInteger
- Dim Attack AsInteger
- NoRowsTablekddcup = ds.Tables("training").Rows.Count
- NoRowsTablerootkit = ds.Tables("rootkit").Rows.Count
-
- For Attack = 0 To NoRowsTablerootkit - 1
-     id = ds.Tables("rootkit").Rows(Attack).Item(0)
-     f14 = ds.Tables("rootkit").Rows(Attack).Item(1)
-     f17 = ds.Tables("rootkit").Rows(Attack).Item(2)
-     f25 = ds.Tables("rootkit").Rows(Attack).Item(3)
-     f36 = ds.Tables("rootkit").Rows(Attack).Item(4)
-     f38 = ds.Tables("rootkit").Rows(Attack).Item(5)
-     A = ds.Tables("rootkit").Rows(Attack).Item(6)
-     AB = ds.Tables("rootkit").Rows(Attack).Item(7)
-
- For KddCounter = 0 To NoRowsTablekddcup - 1
-     If f14 = ds.Tables("training").Rows(KddCounter).Item(9) Then
-     If f17 = ds.Tables("training").Rows(KddCounter).Item(10) Then
-     If f25 = ds.Tables("training").Rows(KddCounter).Item(14) Then
-     If f36 = ds.Tables("training").Rows(KddCounter).Item(18) Then
-     If f38 = ds.Tables("training").Rows(KddCounter).Item(19) Then
-     If ds.Tables("training").Rows(KddCounter).Item(21) = "rootkit." Then
-         AB = AB + 1
-     Else
-         A = A + 1
-     EndIf
- EndIf
- EndIf
- EndIf
- EndIf
- EndIf
- Next
- Dim Ocmd AsNew Data.SqlClient.SqlCommand
-     Ocmd.CommandType = CommandType.StoredProcedure
-     Ocmd.Connection = conU2R
-     Ocmd.Parameters.AddWithValue("@id", id)
-     Ocmd.Parameters.AddWithValue("@A", A)
-     Ocmd.Parameters.AddWithValue("@AB", AB)
-     Ocmd.CommandText = "updaterootkit"
- Try
-     conU2R.Open()
-     Ocmd.ExecuteNonQuery()
- Catch ex AsException
- EndTry
-     conU2R.Close()
- Next
-     txtrootkit.Text = "Done"
- EndSub
-
-

```

Code for calculating Fitness Function:

```

- PrivateSub btnbuffer_overflow_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
  btnbuffer_overflow.Click
- Dim dabuffer_overflow AsNewSqlDataAdapter("Select * from buffer_overflow", conU2R)
- Dim ds AsNewDataSet
-     dabuffer_overflow.Fill(ds, "buffer_overflow")
- Dim dosA, dosAB, dosid AsInteger
- Dim dosFitnessValue AsDouble
- Dim NoRowsTabledos AsInteger
-     NoRowsTabledos = ds.Tables("buffer_overflow").Rows.Count
- Dim FirstPopValue, FinalPopValue, maxA, maxAB AsInteger
- For i = 0 To NoRowsTabledos - 1 Step 400
-     maxA = 0

```

```

-         maxAB = 0
-         FirstPopValue = i
-         FinalPopValue = i + 399
-     If NoRowsTabledos < FinalPopValue Then
-         FinalPopValue = NoRowsTabledos - 1
-     EndIf
-     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)
-     EndIf
-     If ds.Tables("buffer_overflow").Rows(j).Item(7) > maxAB Then
-         maxAB = ds.Tables("buffer_overflow").Rows(j).Item(7)
-     EndIf
-     Next
-     For y = FirstPopValue To FinalPopValue
-         dosA = ds.Tables("buffer_overflow").Rows(y).Item(6)
-         dosAB = ds.Tables("buffer_overflow").Rows(y).Item(7)
-         dosid = ds.Tables("buffer_overflow").Rows(y).Item(0)
-         dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
-         dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
-     Dim Ocmd AsNew Data.SqlClient.SqlCommand
-         Ocmd.CommandType = CommandType.StoredProcedure
-         Ocmd.Connection = conU2R
-         Ocmd.Parameters.AddWithValue("@id", dosid)
-         Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
-         Ocmd.CommandText = "fitnessbuffer_overflow"
-     Try
-         conU2R.Open()
-         Ocmd.ExecuteNonQuery()
-     Catch ex AsException
-         MsgBox(ex.Message)
-     EndTry
-     conU2R.Close()
-     Next
-     Next
-
-     txtbuffer_overflow.Text = "Done"
- EndSub
-
- PrivateSub btnloadmodule_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
- btnloadmodule.Click
-     Dim daloadmodule AsNewSqlDataAdapter("Select * from loadmodule", conU2R)
-
-     Dim ds AsNewDataSet
-         daloadmodule.Fill(ds, "loadmodule")
-     Dim dosA, dosAB, dosid AsInteger
-     Dim dosFitnessValue AsDouble
-     Dim NoRowsTabledos AsInteger
-         NoRowsTabledos = ds.Tables("loadmodule").Rows.Count
-     Dim FirstPopValue, FinalPopValue, maxA, maxAB AsInteger
-     For i = 0 To NoRowsTabledos - 1 Step 400
-         maxA = 0
-         maxAB = 0
-         FirstPopValue = i
-         FinalPopValue = i + 399
-     If NoRowsTabledos < FinalPopValue Then
-         FinalPopValue = NoRowsTabledos - 1
-     EndIf
-     For j = FirstPopValue To FinalPopValue
-     If ds.Tables("loadmodule").Rows(j).Item(6) > maxA Then
-         maxA = ds.Tables("loadmodule").Rows(j).Item(6)
-     EndIf
-     If ds.Tables("loadmodule").Rows(j).Item(7) > maxAB Then
-         maxAB = ds.Tables("loadmodule").Rows(j).Item(7)
-     EndIf
-     Next
-     Next

```

```

- Next
- For y = FirstPopValue To FinalPopValue
-     dosA = ds.Tables("loadmodule").Rows(y).Item(6)
-     dosAB = ds.Tables("loadmodule").Rows(y).Item(7)
-     dosid = ds.Tables("loadmodule").Rows(y).Item(0)
-     dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
-     dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
- Dim Ocmd AsNew Data.SqlClient.SqlCommand
-     Ocmd.CommandType = CommandType.StoredProcedure
-     Ocmd.Connection = conU2R
-     Ocmd.Parameters.AddWithValue("@id", dosid)
-     Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
-     Ocmd.CommandText = "fitnessloadmodule"
- Try
-     conU2R.Open()
-     Ocmd.ExecuteNonQuery()
- Catch ex AsException
-     MsgBox(ex.Message)
- EndTry
- conU2R.Close()
- Next
- Next
- txtloadmodule.Text = "Done"
- EndSub
-
- PrivateSub btnperl_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnperl.Click
- Dim daperl AsNewSqlDataAdapter("Select * from perl", conU2R)
-
- Dim ds AsNewDataSet
-     daperl.Fill(ds, "perl")
- Dim dosA, dosAB, dosid AsInteger
- Dim dosFitnessValue AsDouble
- Dim NoRowsTabledos AsInteger
-     NoRowsTabledos = ds.Tables("perl").Rows.Count
- Dim FirstPopValue, FinalPopValue, maxA, maxAB AsInteger
- For i = 0 To NoRowsTabledos - 1 Step 400
-     maxA = 0
-     maxAB = 0
-     FirstPopValue = i
-     FinalPopValue = i + 399
- If NoRowsTabledos < FinalPopValue Then
-     FinalPopValue = NoRowsTabledos - 1
- EndIf
- For j = FirstPopValue To FinalPopValue
- If ds.Tables("perl").Rows(j).Item(6) > maxA Then
-     maxA = ds.Tables("perl").Rows(j).Item(6)
- EndIf
- If ds.Tables("perl").Rows(j).Item(7) > maxAB Then
-     maxAB = ds.Tables("perl").Rows(j).Item(7)
- EndIf
- Next
- For y = FirstPopValue To FinalPopValue
-     dosA = ds.Tables("perl").Rows(y).Item(6)
-     dosAB = ds.Tables("perl").Rows(y).Item(7)
-     dosid = ds.Tables("perl").Rows(y).Item(0)
-     dosFitnessValue = 2 + ((dosAB - dosA) / (dosA + dosAB)) + dosAB / maxAB - dosA / maxA
-     dosFitnessValue = Double.Parse(dosFitnessValue.ToString("#0.000"))
- Dim Ocmd AsNew Data.SqlClient.SqlCommand
-     Ocmd.CommandType = CommandType.StoredProcedure
-     Ocmd.Connection = conU2R
-     Ocmd.Parameters.AddWithValue("@id", dosid)
-     Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
-     Ocmd.CommandText = "fitnessperl"
- Try
-     conU2R.Open()

```

```

-         Ocmd.ExecuteNonQuery()
-     Catch ex As Exception
-         MsgBox(ex.Message)
-     EndTry
-     conU2R.Close()
- Next
- Next
-     txtperl.Text = "Done"
- EndSub
-
- PrivateSub btnrootkit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
  btnrootkit.Click
-     Dim darootkit As New SqlDataAdapter("Select * from rootkit", conU2R)
-
-     Dim ds As New DataSet
-     darootkit.Fill(ds, "rootkit")
-     Dim dosA, dosAB, dosid As Integer
-     Dim dosFitnessValue As Double
-     Dim NoRowsTabledos As Integer
-     NoRowsTabledos = ds.Tables("rootkit").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
-     EndIf
-     For j = FirstPopValue To FinalPopValue
-     If ds.Tables("rootkit").Rows(j).Item(6) > maxA Then
-         maxA = ds.Tables("rootkit").Rows(j).Item(6)
-     EndIf
-     If ds.Tables("rootkit").Rows(j).Item(7) > maxAB Then
-         maxAB = ds.Tables("rootkit").Rows(j).Item(7)
-     EndIf
-     Next
-     For y = FirstPopValue To FinalPopValue
-         dosA = ds.Tables("rootkit").Rows(y).Item(6)
-         dosAB = ds.Tables("rootkit").Rows(y).Item(7)
-         dosid = ds.Tables("rootkit").Rows(y).Item(0)
-         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 = conU2R
-     Ocmd.Parameters.AddWithValue("@id", dosid)
-     Ocmd.Parameters.AddWithValue("@FitnessValue", dosFitnessValue)
-     Ocmd.CommandText = "fitnessrootkit"
-
-     Try
-         conU2R.Open()
-         Ocmd.ExecuteNonQuery()
-     Catch ex As Exception
-         MsgBox(ex.Message)
-     EndTry
-     conU2R.Close()
- Next
- Next
-     txtrootkit.Text = "Done"
- EndSub

```


Code for using Steady State Genetic Algorithm

```

Private conTrain AsNewSqlConnection("Data Source=(local);Initial Catalog=KDD;Integrated Security=True")
Private conU2R AsNewSqlConnection("Data Source=(local);Initial Catalog=AllAttacks;Integrated Security=True")
Private datrain AsNewSqlDataAdapter("Select * from training", conTrain)
Dim daU2R AsNewSqlDataAdapter("Select * from U2R", conU2R)
Dim ds AsNewDataSet
Dim FirstPopValue, FinalPopValue AsInteger
Dim steep1, steep2, steep3 AsInteger

PrivateSub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

' DataBase Definition
Dim NoRowsTableU2R AsInteger

' Population definitions + parameter of population definition
Dim PopulationIndex AsInteger
Dim icount1, icount2, icount3 AsInteger
Dim TempInt AsInteger
Dim TempDouble AsDouble
Dim RndNum, RndNum1, RndNum2, steepAsInteger
Dim TableNameu2r AsString

    TableNameu2r = "U2R"
    NoRowsTableU2R = ds.Tables(TableNameu2r).Rows.Count
Dim Generation AsInteger
    Generation = 9
Dim Counter AsInteger
    Counter = 0
Dim RepG1 AsDouble(,) = NewDouble(NoRowsTableU2R, 5) {}
Dim RepG2 AsDouble(,) = NewDouble(NoRowsTableU2R, 5) {}
Dim RepG3 AsDouble(,) = NewDouble(NoRowsTableU2R, 5) {}
Dim AfterRep AsDouble(,) = NewDouble(NoRowsTableU2R, 5) {}
Dim AllPopulation_Old AsDouble(,) = NewDouble(NoRowsTableU2R, 1) {}
Dim AllPopulation_New AsDouble(,) = NewDouble(NoRowsTableU2R, 1) {}
Dim CrossedPopulation AsDouble(,) = NewDouble(NoRowsTableU2R, 4) {}

' ##### Start of GENERATION #####
Dim GenerationCounter, LastGenerationCounter AsInteger
DoWhile Counter < NoRowsTableU2R 'And Generation Mod 2 = 0
    GenerationCounter = 0
    LastGenerationCounter = ComboBox1.SelectedIndex + 1

    For j = GenerationCounter To LastGenerationCounter
        ds.Clear()
        daU2R.Fill(ds, TableNameu2r)
        datrain.Fill(ds, "training")
        NoRowsTableU2R = ds.Tables(TableNameu2r).Rows.Count
        For PopulationIndex = 0 To NoRowsTableU2R - 1 Step 250
            FirstPopValue = PopulationIndex
            FinalPopValue = PopulationIndex + 249
            If NoRowsTableU2R < FinalPopValue Then
                FinalPopValue = NoRowsTableU2R - 1
            EndIf
            MsgBox("Generation = " & Generation & "    norowstableu2r = " & NoRowsTableU2R)

' Elitist Selection
'*****
For icount1 = AllPopulation_New.GetLowerBound(0) To AllPopulation_New.GetUpperBound(0) - 1
    AllPopulation_New(icount1, 0) = ds.Tables("U2R").Rows(icount1).Item(0)
    AllPopulation_New(icount1, 1) = ds.Tables("U2R").Rows(icount1).Item(6)
Next
For icount2 = AllPopulation_New.GetLowerBound(0) To AllPopulation_New.GetUpperBound(0) - 1
For icount3 = AllPopulation_New.GetLowerBound(0) To AllPopulation_New.GetUpperBound(0) - 2
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

EndIf
Next
Next

' #####
' Crossover Process
Dim Cross11, Cross21, Cross12, Cross22, Cross13, Cross23, Cross14, Cross24, Cross15, Cross25, Change As Double
For icount1 = FirstPopValue To FinalPopValue - 2 Step 2

    RndNum1 = Int((Rnd() * 100) Mod 5 + 1)
    RndNum2 = Int((Rnd() * 100) Mod 5 + 1)
    If RndNum1 = 1 Then
        Cross11 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(1)
        Cross21 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(1)
    Else
        Cross11 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(1)
        Cross21 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(1)
    EndIf
    If RndNum1 = 2 Then
        Cross12 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(2)
        Cross22 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(2)
    Else
        Cross12 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(2)
        Cross22 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(2)
    EndIf
    If RndNum1 = 3 Then
        Cross13 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(3)
        Cross23 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(3)
    Else
        Cross13 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(3)
        Cross23 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(3)
    EndIf
    If RndNum1 = 4 Then
        Cross14 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(4)
        Cross24 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(4)
    Else
        Cross14 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(4)
        Cross24 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(4)
    EndIf
    If RndNum1 = 5 Then
        Cross15 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(5)
        Cross25 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(5)
    Else
        Cross15 = ds.Tables("U2R").Rows(AllPopulation_New(icount1, 0)).Item(5)
        Cross25 = ds.Tables("U2R").Rows(AllPopulation_New(icount1 + 1, 0)).Item(5)
    EndIf
    If RndNum2 = 1 Then
        Change = Cross11
        Cross11 = Cross21
        Cross21 = Change
    EndIf
    If RndNum2 = 2 Then
        Change = Cross12
        Cross12 = Cross22
        Cross22 = Change
    EndIf
    If RndNum2 = 3 Then
        Change = Cross13

```

```

        Cross13 = Cross23
        Cross23 = Change
    EndIf
    If RndNum2 = 4 Then
        Change = Cross14
        Cross14 = Cross24
        Cross24 = Change
    EndIf
    If RndNum2 = 5 Then
        Change = Cross15
        Cross15 = Cross25
        Cross25 = Change
    EndIf
    CrossedPopulation(icount1, 0) = Cross11
    CrossedPopulation(icount1, 1) = Cross12
    CrossedPopulation(icount1, 2) = Cross13
    CrossedPopulation(icount1, 3) = Cross14
    CrossedPopulation(icount1, 4) = Cross15
    CrossedPopulation(icount1 + 1, 0) = Cross21
    CrossedPopulation(icount1 + 1, 1) = Cross22
    CrossedPopulation(icount1 + 1, 2) = Cross23
    CrossedPopulation(icount1 + 1, 3) = Cross24
    CrossedPopulation(icount1 + 1, 4) = Cross25
Next

''' #####
''' Mutation Process
For steep = FirstPopValue To FinalPopValue Step 10
    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
        EndIf
    EndIf
    If RndNum = 2 Then
        CrossedPopulation(steep, 1) = Int((Rnd() * 100)) Mod 4 + 1
    EndIf
    If RndNum = 3 Then
    EndIf
    If RndNum = 4 Then
        CrossedPopulation(steep, 3) = Int(Rnd() * 100) / 100
    EndIf
    If RndNum = 5 Then
    EndIf
Next

''' #####
''' Evaluation
Dim NoRowsTablekddcup AsInteger
Dim dA AsInteger
Dim d14, d17, d25, d36, d38 AsDouble
Dim Attack AsInteger
    NoRowsTablekddcup = ds.Tables("training").Rows.Count

For Attack = FirstPopValue To FinalPopValue
    dA = 0
        d14 = CrossedPopulation(Attack, 0)
        d17 = CrossedPopulation(Attack, 1)
        d25 = CrossedPopulation(Attack, 2)
        d36 = CrossedPopulation(Attack, 3)
        d38 = CrossedPopulation(Attack, 4)

For kddcounter = 0 To NoRowsTablekddcup - 1

```

```

If d14 = ds.Tables("training").Rows(kddcounter).Item(9) Then
If d17 = ds.Tables("training").Rows(kddcounter).Item(10) Then
If d25 = ds.Tables("training").Rows(kddcounter).Item(14) Then
If d36 = ds.Tables("training").Rows(kddcounter).Item(18) Then
If d38 = ds.Tables("training").Rows(kddcounter).Item(19) Then
dA = dA + 1
EndIf
EndIf
EndIf
EndIf
EndIf
Next
If ComboBox1.SelectedIndex = 0 Then
If Generation Mod 2 = 1 Then
' save array1
RepG1(Attack, 0) = d14
RepG1(Attack, 1) = d17
RepG1(Attack, 2) = d25
RepG1(Attack, 3) = d36
RepG1(Attack, 4) = d38
RepG1(Attack, 5) = dA

ElseIf Generation Mod 2 = 0 Then
' save array2
RepG2(Attack, 0) = d14
RepG2(Attack, 1) = d17
RepG2(Attack, 2) = d25
RepG2(Attack, 3) = d36
RepG2(Attack, 4) = d38
RepG2(Attack, 5) = dA

EndIf
ElseIf ComboBox1.SelectedIndex = 1 Then
If Generation Mod 3 = 1 Then
' save array1
RepG1(Attack, 0) = d14
RepG1(Attack, 1) = d17
RepG1(Attack, 2) = d25
RepG1(Attack, 3) = d36
RepG1(Attack, 4) = d38
RepG1(Attack, 5) = dA

ElseIf Generation Mod 3 = 2 Then
' save array2
RepG2(Attack, 0) = d14
RepG2(Attack, 1) = d17
RepG2(Attack, 2) = d25
RepG2(Attack, 3) = d36
RepG2(Attack, 4) = d38
RepG2(Attack, 5) = dA

ElseIf Generation Mod 3 = 0 Then
' save array3
RepG3(Attack, 0) = d14
RepG3(Attack, 1) = d17
RepG3(Attack, 2) = d25
RepG3(Attack, 3) = d36
RepG3(Attack, 4) = d38
RepG3(Attack, 5) = dA
EndIf
EndIf
Next ' Values of Fitness Function A's
Dim TempDoubl0, TempDoubl1, TempDoubl2, TempDoubl3, TempDoubl4, TempDoubl5 AsDouble
' arrangement + replacemet + Saving Data

```

```

If ComboBox1.SelectedIndex = 0 And Generation Mod 2 = 0 Then
'Next
For icount2 = FirstPopValue To FinalPopValue
For icount3 = FirstPopValue To FinalPopValue - 1
If RepG1(icontains3 + 1, 5) < RepG1(icontains3, 5) Then
    TempDoub10 = RepG1(icontains3, 0)
    TempDoub11 = RepG1(icontains3, 1)
    TempDoub12 = RepG1(icontains3, 2)
    TempDoub13 = RepG1(icontains3, 3)
    TempDoub14 = RepG1(icontains3, 4)
    TempDoub15 = RepG1(icontains3, 5)

RepG1(icontains3, 0) = RepG1(icontains3 + 1, 0)
RepG1(icontains3, 1) = RepG1(icontains3 + 1, 1)
RepG1(icontains3, 2) = RepG1(icontains3 + 1, 2)
RepG1(icontains3, 3) = RepG1(icontains3 + 1, 3)
RepG1(icontains3, 4) = RepG1(icontains3 + 1, 4)
RepG1(icontains3, 5) = RepG1(icontains3 + 1, 5)

RepG1(icontains3 + 1, 0) = TempDoub10
RepG1(icontains3 + 1, 1) = TempDoub11
RepG1(icontains3 + 1, 2) = TempDoub12
RepG1(icontains3 + 1, 3) = TempDoub13
RepG1(icontains3 + 1, 4) = TempDoub14
RepG1(icontains3 + 1, 5) = TempDoub15

EndIf
Next
Next
For icount2 = FirstPopValue To FinalPopValue
For icount3 = FirstPopValue To FinalPopValue - 1
If RepG2(icontains3 + 1, 5) < RepG2(icontains3, 5) Then
    TempDoub10 = RepG2(icontains3, 0)
    TempDoub11 = RepG2(icontains3, 1)
    TempDoub12 = RepG2(icontains3, 2)
    TempDoub13 = RepG2(icontains3, 3)
    TempDoub14 = RepG2(icontains3, 4)
    TempDoub15 = RepG2(icontains3, 5)

RepG2(icontains3, 0) = RepG2(icontains3 + 1, 0)
RepG2(icontains3, 1) = RepG2(icontains3 + 1, 1)
RepG2(icontains3, 2) = RepG2(icontains3 + 1, 2)
RepG2(icontains3, 3) = RepG2(icontains3 + 1, 3)
RepG2(icontains3, 4) = RepG2(icontains3 + 1, 4)
RepG2(icontains3, 5) = RepG2(icontains3 + 1, 5)

RepG2(icontains3 + 1, 0) = TempDoub10
RepG2(icontains3 + 1, 1) = TempDoub11
RepG2(icontains3 + 1, 2) = TempDoub12
RepG2(icontains3 + 1, 3) = TempDoub13
RepG2(icontains3 + 1, 4) = TempDoub14
RepG2(icontains3 + 1, 5) = TempDoub15
EndIf
Next
Next
Dim RepPointer1, RepPointer2, RepPointer3 As Integer
    RepPointer3 = 0
    RepPointer1 = 0
    RepPointer2 = 0
For RepPointer3 = FirstPopValue To FinalPopValue
If RepG1(RepPointer1, 5) <= RepG2(RepPointer2, 5) Then
AfterRep(RepPointer3, 0) = RepG1(RepPointer1, 0)
AfterRep(RepPointer3, 1) = RepG1(RepPointer1, 1)
AfterRep(RepPointer3, 2) = RepG1(RepPointer1, 2)
AfterRep(RepPointer3, 3) = RepG1(RepPointer1, 3)

```

```

AfterRep(RepPointer3, 4) = RepG1(RepPointer1, 4)
AfterRep(RepPointer3, 5) = RepG1(RepPointer1, 5)
    RepPointer1 = RepPointer1 + 1
ElseIf RepG1(RepPointer1, 5) > RepG2(RepPointer2, 5) Then
AfterRep(RepPointer3, 0) = RepG2(RepPointer2, 0)
AfterRep(RepPointer3, 1) = RepG2(RepPointer2, 1)
AfterRep(RepPointer3, 2) = RepG2(RepPointer2, 2)
AfterRep(RepPointer3, 3) = RepG2(RepPointer2, 3)
AfterRep(RepPointer3, 4) = RepG2(RepPointer2, 4)
AfterRep(RepPointer3, 5) = RepG2(RepPointer2, 5)
    RepPointer2 = RepPointer2 + 1
EndIf
Next
Dim CheckCounter As Integer
Dim Redundant As Integer
Dim Feature1, Feature2, Feature3, Feature4, Feature5 As Double
    NoRowsTableU2R = ds.Tables(TableNames2r).Rows.Count
For PopIndex = FirstPopValue To FinalPopValue
    Feature1 = AfterRep(PopIndex, 0)
    Feature2 = AfterRep(PopIndex, 1)
    Feature3 = AfterRep(PopIndex, 2)
    Feature4 = AfterRep(PopIndex, 3)
    Feature5 = AfterRep(PopIndex, 4)
    Redundant = 0
For CheckCounter = 0 To NoRowsTableU2R - 1
If Feature1 = ds.Tables(TableNames2r).Rows(CheckCounter).Item(1) And Feature2 =
ds.Tables(TableNames2r).Rows(CheckCounter).Item(2) And Feature3 =
ds.Tables(TableNames2r).Rows(CheckCounter).Item(3) And Feature4 =
ds.Tables(TableNames2r).Rows(CheckCounter).Item(4) And Feature5 =
ds.Tables(TableNames2r).Rows(CheckCounter).Item(5) Then
    Redundant = Redundant + 1
Exit For
EndIf
Next
Dim Ocmd As New SqlCommand
If Redundant = 0 Then
    Ocmd.CommandType = CommandType.StoredProcedure
    Ocmd.Connection = conU2R
Ocmd.Parameters.AddWithValue("@f14", Feature1)
Ocmd.Parameters.AddWithValue("@f17", Feature2)
Ocmd.Parameters.AddWithValue("@f25", Feature3)
Ocmd.Parameters.AddWithValue("@f36", Feature4)
Ocmd.Parameters.AddWithValue("@f38", Feature5)
Ocmd.Parameters.AddWithValue("@FitnessValue", AfterRep(PopIndex, 5))
Ocmd.Parameters.AddWithValue("@generation", Generation)
    Ocmd.CommandText = "SaveU2RDouble"
Try
conU2R.Open()
Ocmd.ExecuteNonQuery()
Catch ex As Exception
MsgBox(ex.Message)
EndTry
conU2R.Close()
EndIf
Next
ElseIf ComboBox1.SelectedIndex = 1 And Generation Mod 3 = 0 Then
For icount2 = FirstPopValue To FinalPopValue
For icount3 = FirstPopValue To FinalPopValue - 1
If RepG1(icount3 + 1, 5) < RepG1(icount3, 5) Then
    TempDoub10 = RepG1(icount3, 0)
    TempDoub11 = RepG1(icount3, 1)
    TempDoub12 = RepG1(icount3, 2)
    TempDoub13 = RepG1(icount3, 3)
    TempDoub14 = RepG1(icount3, 4)
    TempDoub15 = RepG1(icount3, 5)

```

```

RepG1(icontains3, 0) = RepG1(icontains3 + 1, 0)
RepG1(icontains3, 1) = RepG1(icontains3 + 1, 1)
RepG1(icontains3, 2) = RepG1(icontains3 + 1, 2)
RepG1(icontains3, 3) = RepG1(icontains3 + 1, 3)
RepG1(icontains3, 4) = RepG1(icontains3 + 1, 4)
RepG1(icontains3, 5) = RepG1(icontains3 + 1, 5)
RepG1(icontains3 + 1, 0) = TempDoubl0
RepG1(icontains3 + 1, 1) = TempDoubl1
RepG1(icontains3 + 1, 2) = TempDoubl2
RepG1(icontains3 + 1, 3) = TempDoubl3
RepG1(icontains3 + 1, 4) = TempDoubl4
RepG1(icontains3 + 1, 5) = TempDoubl5

EndIf
Next
Next
For icount2 = FirstPopValue To FinalPopValue
For icount3 = FirstPopValue To FinalPopValue - 1
If RepG2(icontains3 + 1, 5) < RepG2(icontains3, 5) Then
    TempDoubl0 = RepG2(icontains3, 0)
    TempDoubl1 = RepG2(icontains3, 1)
    TempDoubl2 = RepG2(icontains3, 2)
    TempDoubl3 = RepG2(icontains3, 3)
    TempDoubl4 = RepG2(icontains3, 4)
    TempDoubl5 = RepG2(icontains3, 5)

RepG2(icontains3, 0) = RepG2(icontains3 + 1, 0)
RepG2(icontains3, 1) = RepG2(icontains3 + 1, 1)
RepG2(icontains3, 2) = RepG2(icontains3 + 1, 2)
RepG2(icontains3, 3) = RepG2(icontains3 + 1, 3)
RepG2(icontains3, 4) = RepG2(icontains3 + 1, 4)
RepG2(icontains3, 5) = RepG2(icontains3 + 1, 5)

RepG2(icontains3 + 1, 0) = TempDoubl0
RepG2(icontains3 + 1, 1) = TempDoubl1
RepG2(icontains3 + 1, 2) = TempDoubl2
RepG2(icontains3 + 1, 3) = TempDoubl3
RepG2(icontains3 + 1, 4) = TempDoubl4
RepG2(icontains3 + 1, 5) = TempDoubl5
EndIf
Next
Next
For icount2 = FirstPopValue To FinalPopValue
For icount3 = FirstPopValue To FinalPopValue - 1
If RepG3(icontains3 + 1, 5) < RepG3(icontains3, 5) Then
    TempDoubl0 = RepG3(icontains3, 0)
    TempDoubl1 = RepG3(icontains3, 1)
    TempDoubl2 = RepG3(icontains3, 2)
    TempDoubl3 = RepG3(icontains3, 3)
    TempDoubl4 = RepG3(icontains3, 4)
    TempDoubl5 = RepG3(icontains3, 5)

RepG3(icontains3, 0) = RepG3(icontains3 + 1, 0)
RepG3(icontains3, 1) = RepG3(icontains3 + 1, 1)
RepG3(icontains3, 2) = RepG3(icontains3 + 1, 2)
RepG3(icontains3, 3) = RepG3(icontains3 + 1, 3)
RepG3(icontains3, 4) = RepG3(icontains3 + 1, 4)
RepG3(icontains3, 5) = RepG3(icontains3 + 1, 5)

RepG3(icontains3 + 1, 0) = TempDoubl0
RepG3(icontains3 + 1, 1) = TempDoubl1
RepG3(icontains3 + 1, 2) = TempDoubl2
RepG3(icontains3 + 1, 3) = TempDoubl3
RepG3(icontains3 + 1, 4) = TempDoubl4

```

```

RepG3(icount3 + 1, 5) = TempDoub15
EndIf
Next
Next
Dim RepPointer1, RepPointer2, RepPointer3, RepPointer4 AsInteger
    RepPointer3 = 0
    RepPointer1 = 0
    RepPointer2 = 0
    RepPointer4 = 0
For RepPointer3 = FirstPopValue To FinalPopValue
IfRepG1(RepPointer1, 5) <= RepG2(RepPointer2, 5) And RepG1(RepPointer1, 5) <= RepG3(RepPointer4, 5) Then
AfterRep(RepPointer3, 0) = RepG1(RepPointer1, 0)
AfterRep(RepPointer3, 1) = RepG1(RepPointer1, 1)
AfterRep(RepPointer3, 2) = RepG1(RepPointer1, 2)
AfterRep(RepPointer3, 3) = RepG1(RepPointer1, 3)
AfterRep(RepPointer3, 4) = RepG1(RepPointer1, 4)
AfterRep(RepPointer3, 5) = RepG1(RepPointer1, 5)
    RepPointer1 = RepPointer1 + 1
ElseIfRepG2(RepPointer2, 5) <= RepG1(RepPointer1, 5) And RepG2(RepPointer2, 5) <= RepG3(RepPointer4, 5) Then
AfterRep(RepPointer3, 0) = RepG2(RepPointer2, 0)
AfterRep(RepPointer3, 1) = RepG2(RepPointer2, 1)
AfterRep(RepPointer3, 2) = RepG2(RepPointer2, 2)
AfterRep(RepPointer3, 3) = RepG2(RepPointer2, 3)
AfterRep(RepPointer3, 4) = RepG2(RepPointer2, 4)
AfterRep(RepPointer3, 5) = RepG2(RepPointer2, 5)
    RepPointer2 = RepPointer2 + 1
ElseIfRepG3(RepPointer4, 5) <= RepG1(RepPointer1, 5) And RepG3(RepPointer4, 5) <= RepG2(RepPointer2, 5) Then
AfterRep(RepPointer3, 0) = RepG3(RepPointer4, 0)
AfterRep(RepPointer3, 1) = RepG3(RepPointer4, 1)
AfterRep(RepPointer3, 2) = RepG3(RepPointer4, 2)
AfterRep(RepPointer3, 3) = RepG3(RepPointer4, 3)
AfterRep(RepPointer3, 4) = RepG3(RepPointer4, 4)
AfterRep(RepPointer3, 5) = RepG3(RepPointer4, 5)
    RepPointer4 = RepPointer4 + 1
EndIf
Next
EndIf
    Generation = Generation + 1
Next' Result of Population
Next
    Counter = NoRowsTableU2R
ds.Clear()
daU2R.Fill(ds, TableNameu2r)
    NoRowsTableU2R = ds.Tables(TableNameu2r).Rows.Count
    NoRowsTableU2R = NoRowsTableU2R + 1
''' ##### END of GENERATION #####
Loop
MsgBox("End")
EndSub
PrivateSub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
daU2R.Fill(ds, "U2R")
datrain.Fill(ds, "training")
EndSub

PrivateSub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
Dim RepG1 AsDouble(,) = NewDouble(5, 8) {}
MsgBox(RepG1.GetLowerBound(1))
MsgBox(RepG1.GetUpperBound(1))
EndSub
EndClass

```

Code for Testing for R2L

```

private contest AsNewSqlConnection("Data Source=(local);Initial Catalog=KDD;Integrated Security=True")
Private conU2R AsNewSqlConnection("Data Source=(local);Initial Catalog=AllAttacks;Integrated Security=True")

```



```

Private datest AsNewSqlDataAdapter("Select * from testing", contest)
Dim daU2R AsNewSqlDataAdapter("Select * from U2R", conU2R)
Dim NoRowsTabler21 AsInteger
Dim NoRowsTableTest AsInteger
Dim ds AsNewDataSet
Dim r2Iname AsString

PrivateSub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
Try
daU2R.Fill(ds, "U2R")
datest.Fill(ds, "testing")
    r2Iname = "U2R"
Dim U2RCounter, TestCounter AsInteger
    NoRowsTabler21 = ds.Tables(r2Iname).Rows.Count
    NoRowsTableTest = ds.Tables("testing").Rows.Count
Dim r6, r11, r12, r19, r22 AsDouble

For TestCounter = 0 To NoRowsTableTest - 1

    r6 = ds.Tables("testing").Rows(TestCounter).Item(9)
    r11 = ds.Tables("testing").Rows(TestCounter).Item(10)
    r12 = ds.Tables("testing").Rows(TestCounter).Item(14)
    r19 = ds.Tables("testing").Rows(TestCounter).Item(18)
    r22 = ds.Tables("testing").Rows(TestCounter).Item(19)
For U2RCounter = 0 To NoRowsTabler21 - 1
Ifds.Tables(r2Iname).Rows(U2RCounter).Item(1) = r6 Then
Ifds.Tables(r2Iname).Rows(U2RCounter).Item(2) = r11 Then
Ifds.Tables(r2Iname).Rows(U2RCounter).Item(3) = r12 Then
Ifds.Tables(r2Iname).Rows(U2RCounter).Item(4) = r19 Then
Ifds.Tables(r2Iname).Rows(U2RCounter).Item(5) = r22 Then
Ifds.Tables("testing").Rows(TestCounter).Item(21) = "buffer_overflow."Ords.Tables("testing").Rows(TestCounter).Item(21)
= "perl."Ords.Tables("testing").Rows(TestCounter).Item(21) =
"loadmodule."Ords.Tables("testing").Rows(TestCounter).Item(21) = "rootkit."Then
    match.Text = Val(match.Text) + 1
ElseIfds.Tables("testing").Rows(TestCounter).Item(21) = "normal."Then
    txtnormal.Text = Val(txtnormal.Text) + 1
Else
    mismatch.Text = Val(mismatch.Text) + 1
EndIf
Exit For
EndIf
EndIf
EndIf
EndIf
EndIf
Next
Next
MsgBox("END")
Catch ex AsException

EndTry
EndSub
PrivateSub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
EndSub
EndClass

```