جـامـعـة الـشـرق الأوسـط
MIDDLE EAST UNIVERSITY

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

نموذج محسن للخوارزمية الجينية المستقرة لاكتشاف التطفل في الشبكات الحاسوبية

BY

Firas Mohammad Ahmad AlAbsi

Supervisor:

Prof. Reyadh Shaker Naoum

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

(Jul 2012)

Chapter Four: Experimental Results 39

4.1 Overview 39

4.2 KDD Cup 99 39

4.3 Selecting the most Significant Features 40

4.4 Fitness Function 43

4.4.1 Fitness Results 47

4.4.2 Discussion of Fitness Results 47

4.4.3 Comparing Strategy 48

4.4.4 Comparing Results 50

4.5 Tracing the Selection Process 50

4.5.1 Trace with Roulette Wheel Selection 52

4.5.2 Trace with Elitist Selection 53

4.5.3 Trace with Ranking Selection 54

4.5.4 Trace with Stochastic Universal Sampling Selection 56

4.5.5 Trace with Tournament Selection 57

4.6 Comparing between Selection and Crossover types 60

4.6.1 Comparing Strategy 60

4.6.2 Comparing Results and Discussion 61

4.7 The results of Detection Rate and False Positive Rate 62

4.8 Comparing thesis results with other results 63

4.9 How did this thesis achieve its objectives 64

Chapter Five: Conclusion and Future Work 65

5.1 Conclusion 65

5.2 Future Work 66

List of Abbreviations

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

security policies, acceptable use policies, or standard security practices. IDS is a software that automates the intrusion detection process".

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

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

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

1- Not all the internet accesses will be done through firewall.
2- Some of threats are originated inside the firewall
3- Firewalls are subject to attacks.

2.2.2 Intrusion Detection System Taxonomy Elements

1- Knowledge based System vs. Behavior based System:

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

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

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

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

Ex: Parent 1: 10010|000

Parent 2: 00101|101

After Crossover:

Offspring 1: 10010|101

Offspring 2: 00101|000

- Two Points

Step 1: randomly selects two points.

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

Ex: Parent 1: 100|100|00

Parent 2: 001|011|01

After Crossover:

Offspring 1: 100|011|00

Offspring 2: 001|100|01

- Uniform

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

Ex: Parent 1: 10010000

Parent 2: 00101101

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

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

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

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

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

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

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

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

Chapter Four

Experimental Results

4.1 Overview

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

4.2 KDD Cup 99

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

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

Table (4.1): Distribution of Attacks within Training Dataset

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

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

Where:

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

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

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

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

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

Where,

N: the number of records in the population.

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

Counter	Random Number	Selected individual
1	15	2
2	4	6
3	4	6
4	8.03	4
5	3.59	6
6	13.07	2
7	4.03	6
8	13.59	2
9	19.95	14
10	2.95	11
11	9.9	10
12	11.98	9
13	26	15
14	0.95	11
15	3.86	12
16	3.86	12

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

4.5.4 Trace with Stochastic Universal Sampling (SUS):

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

4.6 Comparing between Selection and Crossover types.

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

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

4.6.1 Comparing Strategy.

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

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

Table (4.22): The parameters used in the system

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

```
    Or ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "perl." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "rootkit." Then
-                                   TxtU2R4Count.Text = Val(TxtU2R4Count.Text) + 1
-                                   Else
-                                   TxtNotU2R4Count.Text = Val(TxtNotU2R4Count.Text) + 1
-                                     End If
-                                     Exit For
-                                End If
-                            End If
-                        End If
-                    End If
-                End If
-            Next
-
-            r6 = ds.Tables("KDDtest$").Rows(TestCounter).Item(6)
-            r11 = ds.Tables("KDDtest$").Rows(TestCounter).Item(11)
-            r12 = ds.Tables("KDDtest$").Rows(TestCounter).Item(12)
-            r19 = ds.Tables("KDDtest$").Rows(TestCounter).Item(19)
-            r22 = ds.Tables("KDDtest$").Rows(TestCounter).Item(22)
-
-            Dim R2LCounter As Integer
-            For R2LCounter = 0 To NoRowsTableR2L - 1
-                If ds.Tables("R2L").Rows(R2LCounter).Item(1) = r6 Then
-                    If ds.Tables("R2L").Rows(R2LCounter).Item(2) = r11 Then
-                        If ds.Tables("R2L").Rows(R2LCounter).Item(3) = r12 Then
-                            If ds.Tables("R2L").Rows(R2LCounter).Item(4) = r19 Then
-                                If ds.Tables("R2L").Rows(R2LCounter).Item(5) = r22 Then
-                                    If ds.Tables("KDDtest$").Rows(TestCounter).Item(42)
    = "ftp_write." Or ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "guess_passwd." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "imap." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "multihop." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "phf." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "spy." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "warezclient." Or
    ds.Tables("KDDtest$").Rows(TestCounter).Item(42) = "warezmaster." Then
-                                   TxtR2L4Count.Text = Val(TxtR2L4Count.Text) + 1
-                                   Else
-                                   TxtNotR2L4Count.Text = Val(TxtNotR2L4Count.Text) + 1
-                                     End If
-                                     Exit For
-                                End If
-                            End If
-                        End If
-                    End If
-                End If
-            Next
-        Next
```

- Code for calculating A and AB.

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

```
-                Ocmd.Parameters.AddWithValue("@AB", dAB)
-                Ocmd.CommandText = "updatedos1"
-                Try
-                    dos.Open()
-                    Ocmd.ExecuteNonQuery()
-                Catch ex As Exception
-                End Try
-                dos.Close()
-            Next
-            TxtDosDone.Text = "Done"
-        End Sub
-
-        Private Sub Button18_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button18.Click
-            Dim dados As New SqlDataAdapter("Select * from land", dos)
-            Dim ds As New DataSet
-            dados.Fill(ds, "land")
-            datrain.Fill(ds, "kddcup$")
-
-
-            Dim d7, d8, d12, d13, d23 As Double
-            Dim dA, dAB, did As Integer
-
-            Dim NoRowsTablekddcup As Integer
-            Dim NoRowsTableDos As Integer
-            Dim Attack As Integer
-
-            NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-            NoRowsTableDos = ds.Tables("land").Rows.Count
-
-            For Attack = 0 To NoRowsTableDos - 1
-
-                d7 = ds.Tables("land").Rows(Attack).Item(1)
-                d8 = ds.Tables("land").Rows(Attack).Item(2)
-                d12 = ds.Tables("land").Rows(Attack).Item(3
-                d13 = ds.Tables("land").Rows(Attack).Item(4
-                d23 = ds.Tables("land").Rows(Attack).Item(5)
-                dA = ds.Tables("land").Rows(Attack).Item(6)
-                dAB = ds.Tables("land").Rows(Attack).Item(7)
-                did = ds.Tables("land").Rows(Attack).Item(8)
-
-                For KddCounter = 0 To NoRowsTablekddcup - 1
-                    If d7 = ds.Tables("kddcup$").Rows(KddCounter).Item(6) Then
-                        If d8 = ds.Tables("kddcup$").Rows(KddCounter).Item(7) Then
-                            If d12 = ds.Tables("kddcup$").Rows(KddCounter).Item(11) Then
-                                If d13 = ds.Tables("kddcup$").Rows(KddCounter).Item(12) Then
-                                If d23 = ds.Tables("kddcup$").Rows(KddCounter).Item(22) Then
-                            If ds.Tables("kddcup$").Rows(KddCounter).Item(41) = "land." Then
-                                        dAB = dAB + 1
-                                    Else
-                                        dA = dA + 1
-                                    End If
-                                End If
-                            End If
-                        End If
-                    End If
-                Next
-
-                Dim Ocmd As New Data.SqlClient.SqlCommand
-                Ocmd.CommandType = CommandType.StoredProcedure
-                Ocmd.Connection = dos
-                Ocmd.Parameters.AddWithValue("@id", did)
-                Ocmd.Parameters.AddWithValue("@A", dA)
-                Ocmd.Parameters.AddWithValue("@AB", dAB)
-                Ocmd.CommandText = "updatedos2"
-                Try
-                    dos.Open()
-                    Ocmd.ExecuteNonQuery()
-                Catch ex As Exception
-                End Try
-                dos.Close()
```

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

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

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

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

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

```
-       Dim dau2r As New SqlDataAdapter("Select * from rootkit", U2R)
-       Dim steep1, steep2, steep3 As Integer
-       Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
-           dau2r.Fill(ds, "rootkit")
-       End Sub
-       Private Sub Button13_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button13.Click
-           If CmbSelection.SelectedItem = Nothing Then
-               MsgBox("Please Select Selection type")
-               Exit Sub
-           ElseIf CmbCrossover.SelectedItem = Nothing Then
-               MsgBox("Please Select Crossover type")
-               Exit Sub
-           ElseIf CmbReplacment.SelectedItem = Nothing Then
-               MsgBox("Please Select Replacement type")
-               Exit Sub
-           ElseIf TxtPopSize.Text = "" Then
-               MsgBox("Please determine the population size")
-               Exit Sub
-           End If
-           GAParameter = ""
-           If CmbSelection.SelectedIndex = 0 Then
-               GAParameter = GAParameter & "1"
-           ElseIf CmbSelection.SelectedIndex = 1 Then
-               GAParameter = GAParameter & "2"
-           ElseIf CmbSelection.SelectedIndex = 2 Then
-               GAParameter = GAParameter & "3"
-           ElseIf CmbSelection.SelectedIndex = 3 Then
-               GAParameter = GAParameter & "4"
-           ElseIf CmbSelection.SelectedIndex = 4 Then
-               GAParameter = GAParameter & "5"
-           End If
-           ' DataBase Definition
-           Dim NoRowsTableU2R As Integer
-           NoRowsTableU2R = ds.Tables("u2r").Rows.Count
-           ' Population definitions + parameter of population definition
-           Dim PopIndex, PopulationIndex As Integer
-           ' RWS -------------
-           Dim RndNum As Double
-           Dim SelectedIndividual As Integer
-           Dim SumFitness, AverageFitness, ExpectedFitness, SumExpectedFitness,
    SumExpectedFitness1 As Double
-           Dim j As Integer
-           Dim icount As Integer
-           ' Elitest -----------
-           Dim icount1, icount2, icount3 As Integer
-           Dim TempInt As Integer
-           Dim TempDouble As Double
-           ' Ranking -----------
-           Dim AllPopulation_Ranked As Integer(,) = New Integer(NoRowsTableU2R, 1) {}
-           Dim AllPopulation_FitnessRanked As Double(,) = New Double(NoRowsTableU2R, 1) {}
-           Dim min, max As Double
-           Dim y As Double
-           Dim x As Integer
-           ' Tournament -------------
-           Dim RndNum1, RndNum2, Difference, steep As Integer
-           Dim TableNameu2r As String
-           NoRowsTableU2R = ds.Tables("rootkit").Rows.Count
-           TableNameu2r = "rootkit"
-           dau2r.Fill(ds, TableNameu2r)
-           NoRowsTableU2R = ds.Tables(TableNameu2r).Rows.Count
-           Dim Generation As Integer
-           Generation = 1
-           Dim OldGeneration As Double(,) = New Double(NoRowsTableU2R, 11) {}
-           Dim LastGenerationNoRowsTabledos As Integer
-           LastGenerationNoRowsTabledos = 0
-
-           ' ##### Start of GENERATION   ########################################
-           Do While LastGenerationNoRowsTabledos < NoRowsTableU2R 'And Generation <> 16
-               ds.Clear()
-               datrain.Fill(ds, "kddcup$")
```

```
-               dau2r.Fill(ds, TableNameu2r)
-               NoRowsTableU2R = ds.Tables(TableNameu2r).Rows.Count
-               Dim AllPopulation_Old As Double(,) = New Double(NoRowsTableU2R, 1) {}
-               Dim AllPopulation_New As Double(,) = New Double(NoRowsTableU2R, 1) {}
-               Dim CrossedPopulation As Double(,) = New Double(NoRowsTableU2R, 4) {}
-               Dim CurrentGeneration As Double(,) = New Double(NoRowsTableU2R, 11) {}
-               Dim SelectedGeneration As Double(,) = New Double(NoRowsTableU2R, 11) {}
-               For PopulationIndex = 0 To NoRowsTableU2R - 1 Step Val(TxtPopSize.Text)
-                   FirstPopValue = PopulationIndex
-                   FinalPopValue = PopulationIndex + Val(TxtPopSize.Text) - 1
-                   If NoRowsTableU2R < FinalPopValue Then
-                       FinalPopValue = NoRowsTableU2R - 1
-                   End If
-                   If Generation = 1 Then
-                       For PopIndex = 0 To NoRowsTableU2R - 1
-                           OldGeneration(PopIndex, 0) = 0
-                           OldGeneration(PopIndex, 1) = 0
-                           OldGeneration(PopIndex, 2) = 0
-                           OldGeneration(PopIndex, 3) = 0
-                           OldGeneration(PopIndex, 4) = 0
-                           OldGeneration(PopIndex, 5) = 0
-                           OldGeneration(PopIndex, 6) = 0
-                           OldGeneration(PopIndex, 7) = 0
-                           OldGeneration(PopIndex, 8) = 0
-                           OldGeneration(PopIndex, 9) = 0
-                           OldGeneration(PopIndex, 10) = 0
-                           OldGeneration(PopIndex, 11) = 0
-                       Next
-                   End If
-                   ' ################################################
-                   ' Selection Process
-                   If CmbSelection.SelectedIndex = 0 Then
-                       ' RWS Selection
-                       SumFitness =
-                       For icount = FirstPopValue To FinalPopValue
-                       SumFitness = SumFitness + ds.Tables("rootkit").Rows(icount).Item(9)
-                       Next
-                       AverageFitness = SumFitness / (FinalPopValue - FirstPopValue + 1)
-                       SumExpectedFitness = 0
-                       For icount = FirstPopValue To FinalPopValue
-                       SumExpectedFitness = SumExpectedFitness +
    ds.Tables("rootkit").Rows(icount).Item(9) / AverageFitness
-                       Next
-                       For icount = FirstPopValue To FinalPopValue
-                           RndNum = Int((Rnd() * 100)) Mod SumExpectedFitness
-                           SumExpectedFitness1 = 0
-                           j = FirstPopValue
-                           While (j < FinalPopValue)
-                   ExpectedFitness = ds.Tables("rootkit").Rows(j).Item(9) / AverageFitness
-                               SumExpectedFitness1 = SumExpectedFitness1 + ExpectedFitness
-                               SelectedIndividual = ds.Tables("rootkit").Rows(j).Item(8)
-                               If SumExpectedFitness1 > RndNum Then
-                                   Exit While
-                               Else
-                                   j = j + 1
-                               End If
-                           End While
-   AllPopulation_New(icount, 0) = ds.Tables("rootkit").Rows(SelectedIndividual - 1).Item(8)
-   AllPopulation_New(icount, 1) = ds.Tables("rootkit").Rows(SelectedIndividual - 1).Item(9)
-                       Next
-                   ElseIf CmbSelection.SelectedIndex = 1 Then
-                       'Elitist Selection
-                       '*****************************************************
-                       For icount1 = FirstPopValue To FinalPopValue
-   AllPopulation_New(icount1, 0) = ds.Tables("rootkit").Rows(icount1).Item(8)
-   AllPopulation_New(icount1, 1) = ds.Tables("rootkit").Rows(icount1).Item(9)
-                       Next
-                       For icount2 = FirstPopValue To FinalPopValue
-                           For icount3 = FirstPopValue To FinalPopValue - 1
-   If AllPopulation_New(icount3 + 1, 1) > AllPopulation_New(icount3, 1) Then
-                                   TempInt = AllPopulation_New(icount3 + 1, 0)
-                                   TempDouble = AllPopulation_New(icount3 + 1, 1)
```

```
-    AllPopulation_New(icount3 + 1, 0) = AllPopulation_New(icount3, 0)
-    AllPopulation_New(icount3 + 1, 1) = AllPopulation_New(icount3, 1)
-                                    AllPopulation_New(icount3, 0) = TempInt
-                                    AllPopulation_New(icount3, 1) = TempDouble
-                                End If
-                            Next
-                        Next
-                        '*******************************************************
-                    ElseIf CmbSelection.SelectedIndex = 2 Then
-                        ' Ranking Selection
-                        For icount1 = FirstPopValue To FinalPopValue
-    AllPopulation_New(icount1, 0) = ds.Tables("rootkit").Rows(icount1).Item(8)
-    AllPopulation_New(icount1, 1) = ds.Tables("rootkit").Rows(icount1).Item(9)
-                        Next
-                        For icount2 = FirstPopValue To FinalPopValue
-                            For icount3 = FirstPopValue To FinalPopValue - 1
-    If AllPopulation_New(icount3 + 1, 1) > AllPopulation_New(icount3, 1) Then
-                                    TempInt = AllPopulation_New(icount3 + 1, 0)
-                                    TempDouble = AllPopulation_New(icount3 + 1, 1)
-    AllPopulation_New(icount3 + 1, 0) = AllPopulation_New(icount3, 0)
-    AllPopulation_New(icount3 + 1, 1) = AllPopulation_New(icount3, 1)
-                                    AllPopulation_New(icount3, 0) = TempInt
-                                    AllPopulation_New(icount3, 1) = TempDouble
-                                End If
-                            Next
-                        Next
-                        For PopIndex = FirstPopValue To FinalPopValue
-                          AllPopulation_Ranked(PopIndex, 0) = AllPopulation_New(PopIndex, 0)
-                            AllPopulation_Ranked(PopIndex, 1) = FinalPopValue - PopIndex + 1
-                        Next
-                        For PopIndex = FirstPopValue To FinalPopValue
-                            x = Rnd() * 100
-                            y = x / 100 + 1
-                            max = y
-                            If max = 1 Then
-                                max = 1.1
-                            End If
-                            min = 2 - max
-            AllPopulation_FitnessRanked(PopIndex, 0) = AllPopulation_Ranked(PopIndex, 0)
-                AllPopulation_FitnessRanked(PopIndex, 1) = max - (max - min) *
    ((AllPopulation_Ranked(PopIndex, 1) - 1) / (Val(TxtPopSize.Text) - 1))
-                        Next
-                        SumFitness = 0
-                        For icount = FirstPopValue To FinalPopValue
-                            SumFitness = SumFitness + AllPopulation_FitnessRanked(icount, 1)
-                        Next
-                        AverageFitness = SumFitness / (FinalPopValue - FirstPopValue + 1)
-                        SumExpectedFitness = 0
-                        For icount = FirstPopValue To FinalPopValue
-    SumExpectedFitness = SumExpectedFitness + ds.Tables("rootkit").Rows(icount).Item(9) /
    AverageFitness
-                        Next
-                        For icount = FirstPopValue To FinalPopValue
-                            RndNum = Int((Rnd() * 100)) Mod SumExpectedFitness
-                            SumExpectedFitness1 = 0
-                            j = FirstPopValue
-                            While (j < FinalPopValue)
-                    ExpectedFitness = ds.Tables("rootkit").Rows(j).Item(9) / AverageFitness
-                                SumExpectedFitness1 = SumExpectedFitness1 + ExpectedFitness
-                                SelectedIndividual = AllPopulation_FitnessRanked(j, 0)
-                                If SumExpectedFitness1 > RndNum Then
-                                    Exit While
-                                Else
-                                    j = j + 1
-                                End If
-                            End While
-                            AllPopulation_New(icount, 0) = AllPopulation_FitnessRanked(j, 0)
-                            AllPopulation_New(icount, 1) = AllPopulation_FitnessRanked(j, 1)
-                        Next
-                        ' ************************************************
-                    ElseIf CmbSelection.SelectedIndex = 3 Then
-                        ' SUS
```

```
-                For steep = 0 To NoRowsTableU2R - 1 Step 5
-                    RndNum = Int((Rnd() * 100)) Mod 5 + 1
-                    If RndNum = 1 Then
-                        If CrossedPopulation(steep, 0) = 0 Then
-                            CrossedPopulation(steep, 0) = 1
-                        Else
-                            CrossedPopulation(steep, 0) = 0
-                        End If
-                    End If
-                    If RndNum = 2 Then
-                        CrossedPopulation(steep, 1) = Int((Rnd() * 100)) Mod 4 + 1
-                    End If
-                    If RndNum = 3 Then
-                        If CrossedPopulation(steep, 2) = 0 Then
-                            CrossedPopulation(steep, 2) = 1
-                        Else
-                            CrossedPopulation(steep, 2) = 0
-                        End If
-                    End If
-                    If RndNum = 4 Then
-                        If CrossedPopulation(steep, 3) = 0 Then
-                            CrossedPopulation(steep, 3) = 1
-                        Else
-                            CrossedPopulation(steep, 3) = 0
-                        End If
-                    End If
-                    If RndNum = 5 Then
-                        CrossedPopulation(steep, 4) = Int(Rnd() * 100) / 100
-                    End If
-                Next
-                ' ##########################################
-                ' Evaluation
-                Dim NoRowsTablekddcup As Integer
-                Dim kddcounter As Integer
-                Dim dAB, dA As Integer
-                Dim maxAB, maxA As Double
-                Dim d7, d8, d12, d13, d23 As Double
-                Dim Attack As Integer
-                NoRowsTablekddcup = ds.Tables("kddcup$").Rows.Count
-                maxA = 0
-                maxAB = 0
-                For j = 0 To NoRowsTableU2R - 1
-                    If ds.Tables(TableNameu2r).Rows(j).Item(6) > maxA Then
-                        maxA = ds.Tables(TableNameu2r).Rows(j).Item(6)
-                    End If
-                    If ds.Tables(TableNameu2r).Rows(j).Item(7) > maxAB Then
-                        maxAB = ds.Tables(TableNameu2r).Rows(j).Item(7)
-                    End If
-                Next
-                Dim FitnessValue As Double
-                For Attack = 0 To NoRowsTableU2R - 1
-                    dAB = 0
-                    dA = 0
-                    d7 = CrossedPopulation(Attack, 0)
-                    d8 = CrossedPopulation(Attack, 1)
-                    d12 = CrossedPopulation(Attack, 2)
-                    d13 = CrossedPopulation(Attack, 3)
-                    d23 = CrossedPopulation(Attack, 4)
-                    For kddcounter = 0 To NoRowsTablekddcup - 1
-                        If d7 = ds.Tables("kddcup$").Rows(kddcounter).Item(6) Then
-                            If d8 = ds.Tables("kddcup$").Rows(kddcounter).Item(7) Then
-                                If d12 = ds.Tables("kddcup$").Rows(kddcounter).Item(11) Then
-                                If d13 = ds.Tables("kddcup$").Rows(kddcounter).Item(12) Then
-                                If d23 = ds.Tables("kddcup$").Rows(kddcounter).Item(22) Then
-                             If ds.Tables("kddcup$").Rows(kddcounter).Item(41) = "rootkit." Then
-                                                dAB = dAB + 1
-                                            Else
-                                                dA = dA + 1
-                                            End If
-                                        End If
-                                    End If
-                                End If
```