

GOLIATH.NET OBFUSCATOR 5.X (some features)

goliath assembly.ext /renaming /call /fakevar /fakemath /fakejmp

```
<ObfuscationAttribute(Exclude:=True)>
Private Sub ChangeNextNodeLocation(ByVal btnNode As ButtonNode, ByVal spacing As Integer)

    Dim parent As ButtonNode = btnNode.Parent
    '<<<<
    If (parent.Parent Is Nothing) Then

        For i As Integer = (btnNode.NodePosition + 1) To parent.Count - 1

            Dim node2 As ButtonNode = CType(parent.Item(i), ButtonNode)
            '<<<<
            Me.ChangeVerticalLocation(node2, spacing)

        Next i

    Else

        For j As Integer = (btnNode.NodePosition + 1) To parent.Count - 1

            Dim node3 As ButtonNode = CType(parent.Item(j), ButtonNode)
            '<<<<
            Me.ChangeVerticalLocation(node3, spacing)

        Next j
        Me.ChangeNextNodeLocation(btnNode.Parent, spacing)

    End If

End Sub
```

ORIGINAL METHOD

GOLIATH .NET OBFUSCATOR 5.X (some features)

goliath assembly.ext /renaming /call /fakevar /fakemath /fakejmp

The screenshot shows the Red Gate's .NET Reflector interface. On the left, the class hierarchy for `ButtonNode` is displayed, including methods like `ChangeNextNodeLocation`. The main window shows the disassembled code for `ChangeNextNodeLocation` with obfuscation attributes. The code is as follows:

```
<Obfuscation(Exclude:=True)> _
Private Sub ChangeNextNodeLocation(ByVal btnNode As ButtonNode, ByVal spacing As Integer)
    Dim parent As ButtonNode = btnNode.Parent
    If (parent.Parent Is Nothing) Then
        Dim num3 As Integer = (parent.Count - 1)
        Dim i As Integer = (btnNode.NodePosition + 1)
        Do While (i <= num3)
            Dim node2 As ButtonNode = DirectCast(parent.Item(i), ButtonNode)
            Me.ChangeVerticalLocation(node2, spacing)
            i += 1
        Loop
    Else
        Dim num4 As Integer = (parent.Count - 1)
        Dim j As Integer = (btnNode.NodePosition + 1)
        Do While (j <= num4)
            Dim node3 As ButtonNode = DirectCast(parent.Item(j), ButtonNode)
            Me.ChangeVerticalLocation(node3, spacing)
            j += 1
        Loop
    End If
    Me.ChangeNextNodeLocation(btnNode.Parent, spacing)
End Sub
```

At the bottom of the interface, there is a yellow banner with the text: "Step through this assembly in the Visual Studio debugger using .NET Reflector Pro" and a button labeled "Install VS add-in".

SPY ORIGINAL METHOD
WITH .NET REFLECTOR

GOLIATH .NET OBFUSCATOR 5.X (some features)

goliath assembly.ext /renaming /call /fakevar /fakemath /fakejmp

The screenshot shows the Red Gate's .NET Reflector interface. On the left, the 'Derived Types' tree lists various methods, with 'ChangeNextNodeLocation(a, Int32) : Void' selected. The main window displays the decompiled code for this method, which is obfuscated. The code includes attributes like '<Obfuscation(Exclude:=True), MethodImpl(MethodImplOptions.NoInlining)> _' and a 'Private Sub ChangeNextNodeLocation' with several nested loops and conditional statements. Handwritten text in the bottom right corner reads 'OBFUSCATED METHOD WITH: /RENAMING'. At the bottom of the window, there is a yellow banner with an information icon and the text 'Step through this assembly in the Visual Studio debugger using .NET Reflector Pro' and an 'Install VS add-in' button.

```
Private Sub ChangeNextNodeLocation(ByVal a As a, ByVal A As Integer)
    Dim a2 As a = a.F
    If (a2.F Is Nothing) Then
        Dim num3 As Integer = (a2.Count - 1)
        Dim i As Integer = (a.f + 1)
        Do While (i <= num3)
            Dim a3 As a = DirectCast(a2.Item(i), a)
            Me.a(a3, A)
            i += 1
        Loop
    Else
        Dim num4 As Integer = (a2.Count - 1)
        Dim j As Integer = (a.f + 1)
        Do While (j <= num4)
            Dim a4 As a = DirectCast(a2.Item(j), a)
            Me.a(a4, A)
            j += 1
        Loop
        Me.ChangeNextNodeLocation(a.F, A)
    End If
End Sub
```

OBFUSCATED METHOD WITH:
/RENAMING

Step through this assembly in the Visual Studio debugger using .NET Reflector Pro

GOLIATH .NET OBFUSCATOR 5.X (some features)

goliath assembly.ext /renaming /call /fakevar /fakemath /fakejmp

The screenshot displays the Red Gate's .NET Reflector application. The left pane shows a tree view of the assembly's types, with `ChangeNextNodeLocation(a, Int32) : Void` selected. The right pane, titled "Disassembler", shows the following code:

```
<Obfuscation(Exclude:=True, MethodImpl(MethodImplOptions.NoInlining)> _
Private Sub ChangeNextNodeLocation(ByVal a As a, ByVal A As Integer)
  Dim a2 As a = a.a(a)
  If (a.a(a2) Is Nothing) Then
    Dim num3 As Integer = (a.a(DirectCast(a2, ArrayList)) - 1)
    Dim i As Integer = (a.A(a) + 1)
    Do While (i <= num3)
      Dim a3 As a = DirectCast(a.a(DirectCast(a2, ArrayList), i), a)
      a.a(Me, a3, A)
      i += 1
    Loop
  Else
    Dim num4 As Integer = (a.a(DirectCast(a2, ArrayList)) - 1)
    Dim j As Integer = (a.A(a) + 1)
    Do While (j <= num4)
      Dim a4 As a = DirectCast(a.a(DirectCast(a2, ArrayList), j), a)
      a.a(Me, a4, A)
      j += 1
    Loop
  End If
End Sub
```

The bottom pane shows the following metadata for the selected method:

```
Private Sub ChangeNextNodeLocation(ByVal a As a, ByVal A As Integer)
Declaring Type: WindowsApplication1.a
Assembly: ButtonTreeView, Version=1.0.0.0
```

At the bottom of the window, there is a yellow banner with the text: "Step through this assembly in the Visual Studio debugger using .NET Reflector Pro" and a button labeled "Install VS add-in".

OBFUSCATED METHOD WITH:
/RENAMING
/CALL

GOLIATH .NET OBFUSCATOR 5.X (some features)

goliath assembly.ext /renaming /call /fakevar /fakemath /fakejmp

The screenshot shows the Red Gate's .NET Reflector interface. The left pane displays a tree view of the assembly's types, including various methods and classes. The right pane shows the disassembled code for the method `ChangeNextNodeLocation`. The code is heavily obfuscated, using many temporary variables (e.g., `a5`, `a6`, `a7`, `a8`, `a9`, `a10`, `a11`) and complex expressions. The disassembler options are set to `<MethodImpl(MethodImplOptions.NoInlining), Obfuscation(Exclude:=True)> _`. Below the disassembled code, there are handwritten notes in black ink on a yellow background that read: OBFUSCATED METHOD WITH:
/RENAMING
/CALL
/FAKEVAR

Red Gate's .NET Reflector

File View Tools Help

Visual Basic

Disassembler

```
<MethodImpl(MethodImplOptions.NoInlining), Obfuscation(Exclude:=True)> _
Private Sub ChangeNextNodeLocation(ByVal a As a, ByVal A As Integer)
    Dim a5 As a = a.d(a.a(a))
    If (a.a(a.a5)) Is Nothing Then
        Dim a6 As A = a.b(CInt((a.a(DirectCast(a.a(a5), ArrayList)) - 1)))
        Dim a7 As A = a.b(CInt((a.A(a) + 1)))
        Do While (a.a(a7) <= a.a(a6))
            Dim a8 As a = a.d(DirectCast(a.a(DirectCast(a.a(a5), ArrayList), a.a(a7)), a))
            a.a(Me, a.a(a8), A)
            a7 = a.b(CInt((a.a(a7) + 1)))
        Loop
    Else
        Dim a9 As A = a.b(CInt((a.a(DirectCast(a.a(a5), ArrayList)) - 1)))
        Dim a10 As A = a.b(CInt((a.A(a) + 1)))
        Do While (a.a(a10) <= a.a(a9))
            Dim a11 As a = a.d(DirectCast(a.a(DirectCast(a.a(a5), ArrayList), a.a(a10)), a))
            a.a(Me, a.a(a11), A)
            a10 = a.b(CInt((a.a(a10) + 1)))
        Loop
    End If
End Sub
```

Private Sub ChangeNextNodeLocation(ByVal a As a, ByVal A As Integer)

Declaring Type: WindowsApplication1.a

Assembly: ButtonTreeView, Version=1.0.0.0

Step through this assembly in the Visual Studio debugger using .NET Reflector Pro

Install VS add-in

GOLIATH .NET OBFUSCATOR 5.X (some features)

goliath assembly.ext /renaming /call /fakevar /fakemath /fakejmp

The screenshot shows the Red Gate's .NET Reflector interface. On the left, a tree view lists various methods and properties. The main window displays the disassembled code for the method `ChangeNextNodeLocation`. The code is heavily obfuscated, using variables like `a5`, `a6`, `a7`, `a8`, `a9`, `a10`, and `a11`. The disassembler options are set to `<MethodImpl(MethodImplOptions.NoInlining), Obfuscation(Exclude:=True)> _`. Handwritten notes in black ink on the right side of the disassembler window list the obfuscation features used: OBFUSCATED METHOD WITH:, `/RENAMING`, `/CALL`, `/FAKEVAR`, and `/FAKEMATH`. The bottom status bar indicates that the assembly can be stepped through in the Visual Studio debugger using .NET Reflector Pro.

```
<MethodImpl(MethodImplOptions.NoInlining), Obfuscation(Exclude:=True)> _
Private Sub ChangeNextNodeLocation(ByVal a As a, ByVal A As Integer)
    Dim a5 As a = a.d(a.a(a))
    If (a.a(a.a5)) Is Nothing Then
        Dim a6 As A = a.b(a.A(a.a(DirectCast(a.a(a5), ArrayList), 1)))
        Dim a7 As A = a.b(a.a(a.A(a), 1))
        Do While (a.a(a7) <= a.a(a6))
            Dim a8 As a = a.d(DirectCast(a.a(DirectCast(a.a(a5), ArrayList), a.a(a7)), a))
            a.a(Me, a.a(a8), A)
            a7 = a.b(a.a(a.a(a7), 1))
        Loop
    Else
        Dim a9 As A = a.b(a.A(a.a(DirectCast(a.a(a5), ArrayList), 1)))
        Dim a10 As A = a.b(a.a(a.A(a), 1))
        Do While (a.a(a10) <= a.a(a9))
            Dim a11 As a = a.d(DirectCast(a.a(DirectCast(a.a(a5), ArrayList), a.a(a10)), a))
            a.a(Me, a.a(a11), A)
            a10 = a.b(a.a(a.a(a10), 1))
        Loop
    End If
End Sub
```

OBFUSCATED METHOD WITH:
`/RENAMING`
`/CALL`
`/FAKEVAR`
`/FAKEMATH`

GOLIATH .NET OBFUSCATOR 5.X (some features)

goliath assembly.ext /renaming /call /fakevar /fakemath /fakejump

The screenshot shows the Red Gate's .NET Reflector interface. The left pane displays a tree view of methods, with `ChangeNextNodeLocation(a, Int32) : Void` selected. The right pane shows the disassembled code for this method, which is heavily obfuscated. The code includes several variable declarations and assignments, such as `Dim a5 As a = a.d(a.a(a))`, and conditional logic like `If (a.a(a.a5)) Is Nothing Then`. The code is wrapped in a `<MethodImpl(MethodImplOptions.NoInlining), Obfuscation(Exclude:=True)>` attribute.

Handwritten notes on the right side of the disassembler pane list the obfuscation features used:

OBFUSCATED METHOD WITH:
/RENAMING
/CALL
/FAKEVAR
/FAKEMATH
/FAKEJMP

At the bottom of the interface, there is a yellow banner with the text: "Step through this assembly in the Visual Studio debugger using .NET Reflector Pro" and a button labeled "Install VS add-in".

GOLIATH .NET OBFUSCATOR 5.X (some features)

goliath assembly.ext /renaming /call /fakevar /fakemath /fakejump

In this document you can view some features of **Goliath .NET Obfuscator 5.x**. The protection techniques proposed can be used for any type of project (winforms, web, wp7, ect.):

- **/RENAMING:** symbols renaming with automatic overloads;
- **/CALL:** obfuscate all "call" to constructors & methods;
- **/FAKEVAR:** obfuscate all "variables" of the methods (using "alias");
- **/FAKEMATH:** obfuscate all "mathematical calculations" performed with primitive operators: "+", "-", "*", "/" and boolean function: "And", "Or", "Xor", "Not";
- **/FAKEJMP:** obfuscate all "conditional-jump": "<", "<=", "=>", ">");

NOTE: to allow verification of this protection techniques, was **not used**: string encryption, obfuscation of numerical values, controlflow obfuscation and encryption of the methods!!!

...choose the best obfuscator...choose: Goliath .NET Obfuscator!