**yudi's Step 2 (CrackMe)**

Now that VBReFormer is a well advanced decompiler for Visual Basic application, I was searching for some unsolved crackmes in order to made sample of decompiling for learning purpose.

The website Crackmes.de contains an impressive number of crackmes applications, a perfect source of samples.

For the first sample of CrackMe solving with VBReFormer Professional I decided to take “Step 2” from yudi [more informations](#).

I will show you, step by step, how it’s simple to solve the yudi’s Step 2 using VBReFormer Professional.

- Running the application:

  ![Step 2](#)

  We can see that a serial is generated using the name of the user.

  How the serial is generated? See the following step.

- Now we just open the “Step 2.exe” file with VBReFormer Professional and getting the following result:
We will now take a look to the first method loaded on Visual Basic application.

```vbnet
Private Sub Form_Load
    var_num6 = -52 - 12 - 20
    Set var_pv23 = Me.Label4()
    var_pv23.Visible() = False
    Set var_pv23 = Me.Timer1()
    var_pv23.Interval() = 1000
End Sub
```

We can see on this capture that the “Label4” visibility is set to False (not visible) at the beginning of the application.

Take a look to that control in the resource editor of VBReFormer and you will agree that it’s the control that show the message “Registered user!”

![Visual Basic Code](image)

<table>
<thead>
<tr>
<th>Label - VB.Label</th>
<th>Object Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption</td>
<td>Registered user!</td>
</tr>
<tr>
<td>ForeColor</td>
<td>#800000016A</td>
</tr>
<tr>
<td>(Left; Top; Width; Height)</td>
<td>360; 4080; 5295; 375</td>
</tr>
<tr>
<td>TabIndex</td>
<td>7</td>
</tr>
<tr>
<td>Alignment</td>
<td>2</td>
</tr>
<tr>
<td>BackStyle</td>
<td>0</td>
</tr>
</tbody>
</table>

We now need to know where the “Label4” control visibility is set to true, and what does the “Timer1” control.
• The analysis of the Timer1 control is interesting but not very useful for the following of this tutorial.

```vbnet
Private Sub Timer1_Timer
    'var_num6 = -52 - 12 - 20
    ' *** API Reference to 'IsDebuggerPresent' From '
    Call FatalExit (0) '{Sub
    var_pv23 = ("OLLYDBG")
    var_pv23 = ("$32770")
    var_pv23 = ("18467-41")
    var_pv23 = ("28387380")
    var_pv23 = ("Import REConstructor v1.6 FINAL (C"
    Call SendMessageA (-4035, 16, 0, -52 - 12) '{Fun
    var_pv23 = ("Resource Hacker")
    Call SendMessageA (-52 - 12, var_pv23) '{Function
    Call SendMessageA (-4036, 16, 0, -52 - 12) '{Fun
    var_pv23 = ("VBExplorer")
    Call SendMessageA (-52 - 12, var_pv23) '{Function
    Call SendMessageA (-4037, 16, 0, -52 - 12) '{Fun
    var_pv23 = ("PVDasm v1.06d Program Disassembler"
    Call SendMessageA (-52 - 12, var_pv23) '{Function
    Call SendMessageA (-4100, 16, 0, -52 - 12) '{Fun
    var_pv23 = ("VBRezQ")
    Call SendMessageA (-52 - 12, var_pv23) '{Function
    Call SendMessageA (-4034, 16, 0, -52 - 12) '{Fun
    Call SendMessageA (-52 - 12, var_pv23) '{Function
    Call SendMessageA (-4108, 16, 0, -52 - 12) '{Fun
End Sub
```

We can see here that the “Timer1_Timer” function is called every second by “Timer1” control in order to check that no debuggers, and if one is running, to close it.

We can note that it also close any MessageBox windows.

• Now we are looking for the code under the “Try” button which check if the key match with the name.

That “Try” button is the “Command1” button in VBReFormer:

```
<table>
<thead>
<tr>
<th>Command1 - VB_CommandButton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption</td>
</tr>
<tr>
<td>Left: Top; Width: Height</td>
</tr>
<tr>
<td>TabIndex</td>
</tr>
</tbody>
</table>
```

Then just look to the Command1_Click() function in order to see the algorithm of key checking:
Private Sub Command1_Click
    Set var_pv2 = Me.Label14
    var_pv2.Visible() = False
    Set var_pv2 = Me.Text1
    var_pv3 = var_pv2.Text
    Set var_pv4 = Me.Text1
    var_pv5 = var_pv4.Text
    var_num8 = ((var_pv3) = ("")) Or ((var_pv5) = (""))
    'var_num6 = -52 - 24 + 12
    'var_num6 = 'var_num6 + 12
    If (var_num8) Then
        var_pv6 = ("Hey")
        var_pv7 = ("need something")
    End If
    var_pv0 = MsgBox(var_pv7, 4160, var_pv6)
    'var_num6 = 'var_num6 + 20
    Set var_pv2 = Me.Text2
    var_pv3 = var_pv2.Text
    Set var_pv4 = Me.Text2
    var_pv5 = var_pv4.Text
    var_num8 = ((var_pv3) = ("")) Or ((var_pv5) = (""))
    'var_num6 = 'var_num6 + 12
    'var_num6 = 'var_num6 + 12
    If (var_num8) Then
        var_pv6 = ("Hey")
        var_pv7 = ("need something")
    End If
    var_pv9 = MsgBox(var_pv7, 4160, var_pv6)
    'var_num6 = 'var_num6 + 20
    Set var_pv2 = Me.Text1
    var_pv3 = var_pv2.Text
    var_pv10 = (var_pv3)
    var_pv11 = (Date$) & ("")
    var_pv12 = (var_pv11) & (Time$)
    var_pv13 = (var_pv12)
    'var_num6 = 'var_num6 + 16
    var_pv12 = Len(var_pv13)
The algorithm seems a little complicated for newbie, but complete and without any syntax and source code error from VBReFormer.

That’s a great thing for us; we will be able to test the application into the Visual Basic IDE later (to make a key generator for example).
By analyzing the code we can see the following:

```
Set var_pv2 = Me.Text1()
var_pv3 = var_pv2.Text()
var_pv10 = (var_pv3)
var_pv11 = (Date$) & (" ")
var_pv12 = (var_pv11) & (Time$)
var_pv13 = (var_pv12)
```

This part of code is showing us that the key is generated from the Name, but also with the Date and the Time!

That’s meaning it’s almost impossible to generate a key that does not expire the following second.

- In order to made the Key Generator, save the project with VBReFormer, and open it with Visual Basic 6. When it’s opened into the Visual Basic IDE, remove the debugger watching functions and just keep the following:
  - Command1_Click
  - Command2_Click

Now remove the following block conditions from Command1_Click function:
```
If (var_num8) Then
var_pv6 = ("Hey")
var_pv7 = ("need something")
var_pv8 = MsgBox(var_pv7, 4160, var_pv6)
End If

If (var_num8) Then
var_pv6 = ("Hey")
var_pv7 = ("need something")
var_pv9 = MsgBox(var_pv7, 4160, var_pv6)
End If
```

These block are showing an alert when the “Name” field and when the “Key” field are empty, but it’s not usefull for a keygen.

At the end of the Command1_Click function we can see the serial check condition:
```
Set var_pv2 = Me.Text2()
var_pv3 = var_pv2.Text()
var_pv21 = (var_pv3)
var_pv22 = (var_pv19 Like var_pv21)
If (((var_pv22) = (True))) Then

Set var_pv2 = Me.Label4()
var_pv2.Visible() = True
End If
```

That code is checking that the serial (stored in var_pv19 variable) generated from the name with the algorithm is the same than the one entered in the “Serial” field (Text2.Text).
To show the generated serial, we just need to replace that condition block by the following line of code:

```vbnet
For var_pv14 = 1 To 24 Step 4
    var_pv19 = ((var_pv19 & Mid$(var_pv18, 'var_num6 = 'var_num6 + 16
Next var_pv14
var_pv20 = (Len(var_pv19) - 1)
var_pv19 = (Mid$(var_pv19, 1, var_pv20))
Text2.Text = var_pv19
```

You must also remove the following line of code which remove the content of the both fields:

```vbnet
Set var_pv2 = Me.Text1()
var_pv2.Text() = ""
Set var_pv2 = Me.Text2()
var_pv2.Text() = ""
Set var_pv2 = Me.Text1()
```

After all change and simplifications, we have the following keygen code:

```vbnet
Private Sub Command1_Click()
    var_pv10 = Text1.Text
    var_pv13 = Date$ & " " & Time$
    For var_pv14 = 1 To Len(var_pv13) Step 1
        If IsNumeric(Mid$(var_pv13, CLng(var_pv14), 1)) Then
            var_pv15 = Asc(Mid$(var_pv13, CLng(var_pv14), 1))
            If var_pv14 <= Len(var_pv10) Then
                var_pv16 = Str(Asc(Mid$(var_pv10, CLng(var_pv14), 1)))
                var_pv16 = Right$(var_pv16, 1)
                var_pv16 = Val(var_pv16)
            End If
            var_pv18 = var_pv18 & Chr$(CLng(var_pv15 + 17 + var_pv16))
            var_pv18 = var_pv18 & Chr$(CLng(var_pv15 + 17 + var_pv16 * 2))
        End If
    Next var_pv14
    For var_pv14 = 1 To 24 Step 4
        var_pv19 = var_pv19 & Mid$(var_pv18, CLng(var_pv14), 4) & "-
Next var_pv14
    var_pv20 = Len(var_pv19) - 1
    var_pv19 = Mid$(var_pv19, 1, var_pv20)
    Text2.Text = var_pv19
End Sub
```

- We now have to test our keygen:
The first window is the windows of our KeyGen created from the original crackme, and the second window is the one of the original Crackme, with the key from the KeyGen.

The result is that our keygen work perfectly!
Just note that the use of date and time make your key valid for only 1 minute after having generated it.

Is it possible to bypass that limitation?
Yes it is ! In fact, to get the “Registered user!” message you even don’t need a key generator. By reading the code you can see that the operator used to perform a comparison between the both string key is the “Like” operator.
The “like” operator allows to compare a string and a pattern...
Then you just can set “*” into the serial field and you will have a key which will be valid at anytime, with any name:

Source code of the key generator can be downloaded here:
http://www.decompiler-vb.net/documentation/crackmes/step_2.zip

Enjoy it!

Sylvain Bruyere
http://www.decompiler-vb.net