

tKC

.NET reversing Tips

3



Basic Patching

Attention

Published for educational Purposes only

Welcome:-

I hope you have read the 2 previous tutorials and liked them too!
This is a quick tutor and it's to give you an idea about patching in .NET applications.

This tutor depends on the Opcodes table in Chapter I, so please make sure you have the Opcodes table since we will use it later.

Boolean-Returning Functions

A Boolean-returning functions is very common, you will face it in many programs, It's a function that returns a Boolean [True, False] data type, sometimes it's used on startup to make sure that the current user has registered or not, or to validate a license file or whatever, and believe me you will see lots of these.

```
IF IsLicensed(Key)=True then
    Application.run(New MainForm)

Else
    MsgBox "Please Register"      → Nag screen
    Application.run(New MainForm)

End IF
```

This is not a real piece of code, I just wrote it this way to explain how these functions can be used to change the program behavior, the **IsLicensed** function returns a value which could be true or false.

The **Boolean** is a 16-bit data type that can only represent two values: True and False, when a numeric value is converted to Boolean, any nonzero value is converted to True, and zero is converted to False. In the other direction, False is converted to zero, True is converted to 1.

The point here is to change the actual bytes of this function to make its return always evaluate to **True** or **False** according to the desired return value, what we have to do here is to find the method in ILdasm and then analyze its code and decide what we must change.

Now we will see the code of a simple **IsLicensed** function here.

Function IsLicensed(ByVal Name As String) As Boolean

If Name = "tkc" Then -----> [1]

Return True -----> [2]

Else

Return False -----> [3]

End If

End Function

And Now in IL

```
-----
.method public instance bool IsLicensed(string Name) cil managed
// SIG: 20 01 02 0E
{
    // Method begins at RVA 0x38b4
    // Code size      30 (0x1e)
    .maxstack 3
    .locals init ([0] bool IsLicensed)
    IL_0000: /* 00 |          */ nop
    IL_0001: /* 03 |          */ ldarg.1
    IL_0002: /* 72 | (70)0000C1 */ ldstr      "tkc"
    IL_0007: /* 16 |          */ ldc.i4.0
    IL_0008: /* 28 | (0A)000023 */ call      int32
    [Microsoft.VisualBasic]Microsoft.VisualBasic.CompilerServices.StringTy
    pe::StrCmp(string,
string,
bool)
    IL_000d: /* 16 |          */ ldc.i4.0
    IL_000e: /* 33 | 06          */ bne.un.s IL_0016
    IL_0010: /* 17 |          */ ldc.i4.1
    IL_0011: /* 0A |          */ stloc.0
    IL_0012: /* 2B | 08          */ br.s      IL_001c
    IL_0014: /* 2B | 05          */ br.s      IL_001b
    IL_0016: /* 00 |          */ nop
    IL_0017: /* 16 |          */ ldc.i4.0
    IL_0018: /* 0A |          */ stloc.0
    IL_0019: /* 2B | 01          */ br.s      IL_001c
    IL_001b: /* 00 |          */ nop
    IL_001c: /* 06 |          */ ldloc.0
    IL_001d: /* 2A |          */ ret
} // end of method Form1::IsLicensed} // end of method
Form1::IsLicensed
-----
```

About IL code :

```
IL_0002      :      Loads the string "tkc" from metadata
IL_0008      :      Calls the string compare function to test name
IL_000d      :      Pushes an integer of value 0 [ Assume it's A ]
IL_000e     :      Compares A with return value of String compare function
                  And jumps to IL_0016 [Bad boy] if string compare
                  function return is not zero, When 2 strings are equal
                  then the return of Strcmp function is zero
                  This is where we will patch later
IL_0010      :      If the strings are equal then the jump to IL_0016 is
                  Ignored and the function will return True, so an
                  Integer of value 1 is pushed on the stack but if the
                  jump is executed to line IL_0016 then an integer of
                  value 0 is pushed to return False
```

I can't say anything more, now we will invert the jump in line **IL_000e** which contains this code

```
IL_000e:  /* 33      | 06                      */ bne.un.s    IL_0016
```

The **bne.un.s** instruction can be found in the Opcodes table and it Transfers control to a target instruction (short form) when two unsigned integer values or unordered float values are not equal.

We will replace it with a **Beq_S** instruction which Transfers control to a target instruction (short form) if two values are equal.

The actual bytes of **bne.un.s** is **33**

The actual bytes of **Beq_s** is **2E**

So we will search for this function file offset and then replace the bytes, you can find the offset for this function in many ways, I explained 2 methods in Chapter 2 and I won't repeat myself here.

There are many ways to crack this type of function, one of them is to change the 2 first bytes of the function to **17 2A** instead of **00 03** here to make the function always return **True** and not execute any more code, but this method doesn't work all the time, the **17** pushes an integer of value 1 and **2A** returns to the caller of the function with a True Boolean.

This can be used for any jump that you want to invert and it's not only for functions that return True or false, so whenever you want to invert some jump make sure to find the right instruction to replace.

THE END

tKC

Monday, July 03, 2006

www.reteam.org