

Forum:[Hack Forums](#)

Thread: [\[VB.NET\] How to Make A Builder and Stub the BETTER Way \[Source + Explanations\]](#)

Laser™ - March 27, 2014, 8:34 pm

Updated Builder And Stub Tutorial!

The better AND easier way. It's noob proof.

Hey everyone, welcome to my first tutorial on HackForums! In this post I'll be showing you how to properly make a stub builder and the stub itself in VB.NET! If you don't know what those things are, I'll explain them for you:

The builder is the program that accepts information from the user such as an email address, password, and settings, and writes the stub plus those variables to a new file.

The stub is a template in which every file made by the builder will inherit. The builder reads the information in this file, copies it, and writes it to a new file which also has the predefined variables.

An example: Pretty much every RAT and keylogger on the market utilizes a builder/stub system. For a keylogger, you input information such as your email address and password into the builder, and the builder copies this plus the data from the stub to a new file - which is what you would send to your victim.

Note: This is going to be a long tutorial. I'm going to try to explain everything the best I can, so bare with me.

Please note that this tutorial is more directed to beginners that still use the FileOpen blahblah method as shown below, and I'm aware of the other methods available.

Ok, now that that's out of the way, let's get started! If you encounter any errors or have a question, feel free to post it below!

RequirementsMinor experience in VB.NET programming

[Visual Basic Express](#) or [Visual Studio](#)

Before we start any coding, I want to make sure you know how this method is different, and much better, than the typical outdated method. The other method usually looks something like this:

```
Code FileOpen(1, Application.StartupPath & "\sub.exe", OpenMode.Binary, OpenAccess.Read)
FileGet(1, "string to put file into")
FileClose(1)
```

First of all, I don't think this method really even works anymore. I tried using it a couple days ago, which is what caused me to write this tutorial, and I simply couldn't get it to work. Also, this method is slower, more CPU intensive, and more detectable than my method. Yeah that's right, it's more detectable because you typically end up doing something called 'file dropping', which gets picked up by AV's very easily. With my method we're never opening files directly, we're getting the bytes

stored in them instead. You'll see, it's great.

The builder

Let's start off by designing our builder form. Just add a couple textboxes, a checkbox, and a button as shown below:

First off, we have to add the following code so we can use `Encoding.*` without having to type the full path all the time:

Code Imports System.Text

' This is where Public Class Form1 should be

Alright, now let's start working in the `Button1_Click` event. Add the following code and I'll explain what it does after:

Code Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

Dim filePath As String

Dim filesaver As New SaveFileDialog

If filesaver.ShowDialog = Windows.Forms.DialogResult.OK Then

filePath = filesaver.FileName

Dim stubBytes as Byte()

Dim email As String = TextBox1.Text

Dim password As String = TextBox2.Text

Dim useSecure As String = CheckBox1.Checked.ToString

Const FileSplitter = "----FILESPLIT----

End If

End Sub

What we're doing in this code is creating a new `SaveFileDialog` (the browser that pops up) called `filesaver`, checking to make sure the path is valid, and then setting the string `'filePath'` as the chosen path.

Then we have to declare our variables. We get these from the textbox's and the checkbox. The only thing that may be a little weird to you there is that `'.Checked.ToString'`. The reason for this is that we can only convert strings into bytes, which is how we are going to write to the stub. That little function returns the string `'true'` if it's checked, and `'false'` if it isn't. Oh, and the constant value `'----FILESPLIT----` is how we are going to separate the stub code from the variables. You'll understand it when we get to it. All you need to know now is that it's important, so copy it somewhere.

Simple right? Ok, let's move on. Add the following right below the `Const FileSplitter` line:

Code Dim fileSystem = My.Computer.FileSystem

stubBytes = fileSystem.ReadAllBytes(Application.StartupPath & "\Resources\stub.exe")

Don't worry, I'm going to explain all of that right now! We made a new instance of `My.Computer.FileSystem` so we can easily read and writes bytes to and from files. We then get all the bytes from our stub file by using the `ReadAllBytes` method. I'm assuming that you'll have your builder inside a folder, and your stub inside that folder, inside another folder called `'Resources'`. You can change this to match your needs of course. You should already see a major difference between my method and the typical method.

Now comes the meat of the builder; writing the stub PLUS the variables to our new file. First we'll copy the stub's bytes to the file using the following code, right below where we left off:

```
Code fileSystem.WriteAllBytes(filePath, stubBytes, False)
```

```
    fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(FileSplitter), True)
```

Ok, this will probably be the hardest thing for you to understand if you're new, and to be honest it's really not that complicated. This is what the WriteAllBytes method is composed of:

```
Code WriteAllBytes(PathToFile, BytesToWrite, BooleanOverwrite)
```

Using that format, we set the path to the path the user browsed to, the bytes to write is the stub's bytes, and since this is the first thing we're writing to the file we don't want to overwrite. The second line is what I call a variable splitter. We need to put this in-between every piece of data we write and it's how we'll retrieve the written data when we're working in the stub. Of course, we set the boolean to true because we don't want to overwrite the stub bytes.

Hang in there, we're almost done with the builder! Let's write the variables now using the same method:

```
Code fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(email), True)
```

```
    fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(FileSplitter), True)
```

```
    fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(password), True)
```

```
    fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(FileSplitter), True)
```

```
    fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(useSecure), True)
```

Hey, you did it! Now all your data plus the stub data has been written to the new file. Hats off to you, sir! ? Let's just add a MessageBox for confirmation. Here's what your whole class should look like:

```
Code Imports System.Text
```

```
Public Class Form1
```

```
    Dim stubBytes As Byte()
```

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

```
        Dim filePath As String
```

```
        Dim filesaver As New SaveFileDialog
```

```
        If filesaver.ShowDialog = Windows.Forms.DialogResult.OK Then
```

```
            filePath = filesaver.FileName
```

```
            Dim email As String = TextBox1.Text
```

```
            Dim password As String = TextBox2.Text
```

```
            Dim useSecure As String = CheckBox1.Checked.ToString
```

```
            Dim fileSystem = My.Computer.FileSystem
```

```
            Const FileSplitter = "----FILESPLIT----"
```

```
            stubBytes = fileSystem.ReadAllBytes(Application.StartupPath & "\Resources\stub.exe")
```

```
            fileSystem.WriteAllBytes(filePath, stubBytes, False)
```

```
            fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(FileSplitter), True)
```

```
            fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(email), True)
```

```
            fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(FileSplitter), True)
```

```
            fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(password), True)
```

```
            fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(FileSplitter), True)
```

```
            fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(useSecure), True)
```

```
            MessageBox.Show("Stub has been created!")
```

```
        Else
```

```
            MessageBox.Show("That isn't a valid file path!")
```

```
        End If
```

```
    End Sub
```

```
End Class
```

Once again, good job! You're more than half way done now. Great work.

The Stub

Here comes the easy part. Create a new project and setup a couple textbox's like so:

View the code for Form1 and Import System.Text just like last time:

```
Code Imports System.Text
```

```
' This is where Public Class Form1 should be
```

```
And now we will declare all the variables we need:
```

```
Code Dim Settings(), selfText, email, password, useSecure As String
```

```
Dim self As Byte()
```

```
' Private Sub Form1_Load will be here
```

Seem familiar? I bet you see where this is going, don't you? But wait, what's that self as byte declaration? This is another major difference between my method and the common method, our stub doesn't have to open itself per say, it just needs to read itself and store its own bytes in the variable 'self'. Lets get at it, put the following code inside the Form1_Load sub:

```
Code self = My.Computer.FileSystem.ReadAllBytes(Application.ExecutablePath)
```

```
selfText = Encoding.Default.GetString(self)
```

What we're doing here is we're using the same FileSystem as in the builder to get the bytes of the stub, this program, and then we encode the bytes into a readable string. Now to get the variables, or 'settings', that we wrote to the stub with the builder. We will do this with the following code:

```
Code Settings = Split(selfText, "----FILESPLIT----")
```

Recognize that FileSplit? That MUST be the same as the FileSplit you used in your builder program. Check and the check again. What we're doing here is giving the array 'settings' a new value every time the 'variable splitter' shows up in the code. This is why we inserted a FileSplit after every variable we injected into the stub; this is how we're going to read that information.

Good. Now comes the fun part...getting the data! Right below the Settings = blah blah line, write this down:

```
Code email = Settings(1)
```

```
password = Settings(2)
```

```
useSecure = Settings(3)
```

It's very important that you understand how this works. If you're a beginner this might be news to you, but the first piece of data in an array has a ID of 0, NOT 1. So you might be wondering why the email shouldn't be Settings(0), since it's the first piece of information we wrote to the file. Right? Well, not exactly. If you look at the code from the builder program, we actually wrote the stub data first, then the email, and so on. Take a look:

```
Code fileSystem.WriteAllBytes(filePath, stubBytes, False)
```

```
fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(FileSplitter), True)
```

```
fileSystem.WriteAllBytes(filePath, Encoding.Default.GetBytes(email), True)
```

See? Good, it's important that you understand how this works.

The whole class should now look like this:

```
Code Imports System.Text
```

```
Public Class Form1
```

```
    Dim Settings(), selfText, email, password, useSecure As String
```

```
    Dim self As Byte()
```

```
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
        self = My.Computer.FileSystem.ReadAllBytes(Application.ExecutablePath)
```

```
        selfText = Encoding.Default.GetString(self)
```

```
        Settings = Split(selfText, "----FILESPLIT----")
```

```
        email = Settings(1)
```

```
        password = Settings(2)
```

```
        useSecure = Settings(3)
```

```
    End Sub
```

```
End Class
```

Guess what? That's it! You've successfully built a builder and stub the proper way. It's even easier than the other method, isn't it?

Using the Data

Now I'm going to quickly show you how you can use the data you collected in this application. This will be a super simple example, we're basically going to mimic the builder form, but it'll help you understand everything a bit more:

```
Code Me.TextBox1.Text = email
```

```
    Me.TextBox2.Text = password
```

```
    If useSecure.Equals("True") Then
```

```
        Me.CheckBox1.Checked = True
```

```
    Else
```

```
        Me.CheckBox1.Checked = False
```

```
    End If
```

All that code does is sets all the textbox's to show the data that the builder injected into the stub. Also, if the user checked the box in the builder, the box will be checked in the stub!

Build Format

Unless you changed the code in the places I told you you could, this is how you should have your built files set up; Folder -> TutorialBuilder.exe + Another Folder -> Inside that folder is the stub.exe.

Note: Unless you named your project just 'stub', your stub will be called something else when it's compiled. You must rename it to 'stub.exe'.

Now run your builder.exe and voila, it works! Check it out!

WOO, THAT'S IT! Thanks so much for reading my tutorial, I hope it helped you out! And remember, if you have any questions or comments feel free to post them below. That's a wrap. ?

[Download the source!](#)

Note: I spent over 2 hours writing this and making sure I covered every last detail. All I ask in return is for you to leave some feedback if this helped you out and give me credit where it's due. Thanks

and have a good one!

loopy90 - March 27, 2014, 11:30 pm

Nice Tut, you actually took the time to explain things instead of just telling kiddies to paste code into their projects. However, you should use CodeDOM instead of EOF.

Doge - March 28, 2014, 12:40 am

This is actually a very good tutorial, I learnt a few things from reading this. +rep for your high your high quality tutorial OP.

LethalNET - March 28, 2014, 12:56 am

Why influence people to use EOF? This is no where near the "Better" way of making a builder, When things like Codedom and mono.cecil and dnlib are on the table, Where as EOF is the oldest method for making builders in the book.

It was indeed nice of you to go through such lengths in writing this tutorial BUT its rather outdated and people will flame you for this.

Better putting that brain power into learning to use Codedom / cecil / dnlib.

derpin0n - March 28, 2014, 5:21 am

That is a very nice tutorial... But, as Lethal said, EOF is a very old method... You can use for example, CodeDOM, wich is alot easier and better...

Increment - March 28, 2014, 6:43 am

Quote:The stub is a template in which every file made by the builder will inherit.

You obviously don't understand inheritance.

Using FileSplit is really kiddie, when many new solutions came out like said before.

Code If useSecure.Equals("True")
What ?

Laser™ - March 28, 2014, 8:41 am

Thanks for all the feedback guys, really appreciate it.

(Mar 28, 2014 12:56 AM)LethalNET Wrote: [\[CITE\]](#) Why influence people to use EOF? This is no where near the "Better" way of making a builder, When things like Codedom and mono.cecil and dnlib are on the table, Where as EOF is the oldest method for making builders in the book.

It was indeed nice of you to go through such lengths in writing this tutorial BUT its rather outdated and people will flame you for this.

Better putting that brain power into learning to use Codedom / cecil / dnlib.

You're absolutely right, but the point of this tutorial was to redirect people from using the method I showed in the beginning to this, because I believe this is better and easier. It's more directed to beginners who still try to use those old tutorials from 2011. I'll definitely look into those methods though ?

(Mar 28, 2014 06:43 AM)Increment Wrote: [\[CITE\]](#) You obviously don't understand inheritance.

Using FileSplit is really kiddie, when many new solutions came out like said before.

Code If useSecure.Equals("True")
What ?

Actually I do, I was just trying to explain it how I think beginners will understand best. I could have

said copies or replicates, it was just my word choice.

Also, I explained that in the tutorial.

tweety - March 29, 2014, 9:27 am

You're using a method from the 2001. Use resources.

Six-Nine - March 30, 2014, 3:02 am

Very good tutorial.

You explain very good so its easy to understand!

LinkMoney - March 30, 2014, 4:46 am

Wow ! Im gonna Bookmark this :)

Nice Tutorial

Laser™ - March 30, 2014, 2:56 pm

Thanks guys! Glad it helped you out.

topLevel - March 30, 2014, 4:40 pm

Thanks for this man bookmarked for sure!

Madara-Uchiha - March 30, 2014, 4:57 pm

Very good tutorial. You definitely explained everything pretty good so also beginners can follow.

But in these days I would not recommend to use the EOF file method.

In my opinion Codedom or Mono.Cecil is the better way to do the same thing.

Also when creating for example a binary with codedom it will be much smaller compared to the EOF method.;

?????Thr????? - March 30, 2014, 7:43 pm

EOF is pretty insecure and easy to detect, and you seem to be writing everything in here as plaintext, whereas it'd be safer to at least include some type of encryption of some sort. This is a pretty old method.

Laser™ - March 30, 2014, 8:35 pm

(Mar 30, 2014 07:43 PM)?????Thr????? Wrote: [\[CITE\]](#) EOF is pretty insecure and easy to detect, and you seem to be writing everything in here as plaintext, whereas it'd be safer to at least include some type of encryption of some sort. This is a pretty old method.

This has been mentioned a few times before as you probably saw and I responded by saying I'm aware of the better methods but their are already tutorials on them - this is directed towards people who still gravedig The FileOpen EOF tutorials, which is a worse method than this one.

Also, of course if the data is sensitive encrypt it, but that's kind of a different topic and not required for people who are just learning how to generate files.

Sc?rf?c3_ - March 31, 2014, 4:25 am

I want to say you made a neat thread and all explained, good work on that....

(not considering is not the "better" way as claimed for the reasons we already know)

but im appreciating the effort. Good luck!

ubbelol - March 31, 2014, 4:36 am

(Mar 29, 2014 09:27 AM)Tweety. Wrote: [\[CITE\]](#) You're using a method from the 2001. Use resources.

You're using a method from the 2010. Use Cecil/Dnlib.

[Eprouvez] - March 31, 2014, 5:35 am

Nice tutorial but to be honest I threw up a little when I saw FileOpen, forgot it existed. lol Anyway nice work, look into using CodeDom or even AppendFile I think its called.

Laser™ - April 1, 2014, 12:06 pm

(Mar 31, 2014 05:35 AM)[Eprouvez] Wrote: [\[CITE\]](#) Nice tutorial but to be honest I threw up a little when I saw FileOpen, forgot it existed. lol Anyway nice work, look into using CodeDom or even AppendFile I think its called.

Thanks man, but I never used FileOpen, I was showing how bad that method is haha not telling you to use it. Cheers!

Cook John - November 18, 2014, 11:48 pm

thanks for this coach ? I was learning VB.Net (file handling) this will bring my coding to the next level :)

Alxnder - November 19, 2014, 3:17 am

This is actually a very good tutorial thanks

OffCode - November 19, 2014, 5:06 am

Ohh thanks very much this help me so much learning how builder works! Thank you!

Zettabit - November 19, 2014, 12:56 pm

This is still EOF. I'd rather use CodeDom, but at least you're teaching people.

frawg - January 3, 2015, 7:50 am

Well detailed tutorial, thanks for this!

mechic123 - January 5, 2015, 3:43 am

Really nice tut, thanks for posting this amazing tut!

Nyan Kat - January 5, 2015, 5:14 am

That's one nice tutorial, Bookmarked keep up the good tutorials ;)

Adderall - January 9, 2015, 12:40 am

Great tutorial for building stubs, it really helped me with things.
-Timsel10

xVipX - January 9, 2015, 12:12 pm

Nice Tutorial

but the stub able to cryepte ?

Volty - January 9, 2015, 3:31 pm

Cool, thanks for this easy and simple explanation, I have never done a builder and stub in Vb.net, time to try! :D

SIM-3933056 - November 7, 2018, 11:35 am

error....

stubBytes = fileSystem.ReadAllBytes(Application.StartupPath & "\\Resources\\TutorialStub.exe")
can't find it.. even i added it on my resources..

StarZ - November 7, 2018, 1:07 pm

(Nov 7, 2018 12:35 PM)Mondron Wrote: [\[CITE\]](#) error....

stubBytes = fileSystem.ReadAllBytes(Application.StartupPath & "\\Resources\\TutorialStub.exe")

can't find it.. even i added it on my resources..
This thread is almost 4 years old.

Also, This method is almost (if not) always detected by Avira as a TR/Dropper.gen variant. To bypass this you generally need to use resources instead of EOF (End of File) storage.

xHeaven - November 7, 2018, 2:15 pm

(Nov 7, 2018 12:35 PM)Mondron Wrote: [CITE](#) error....

```
stubBytes = fileSystem.ReadAllBytes(Application.StartupPath & "\\Resources\\TutorialStub.exe")
```

can't find it.. even i added it on my resources..

How did you manage to bring back an almost 5 years old thread?

The "TutorialStub.exe" has to be in a folder called "Resources", and the "Resources" folder has to be in the same folder as your main application. [Example](#).

Edit: the dude above me is right btw, this shit is heavily detected by basically everything.