

Forum:[Hack Forums](#)

Thread: [Secure Random Class - Updated](#)

debug - February 13, 2016, 10:04 pm

Hey all,

After Aeon berated me for using System.Random, I finally decided to stop using it. If you want a good read on why it's not secure, view his thread here: <http://coderrail.com/thread-2.html>

Now, for general purposes, System.Random isn't horrible to use, but RNGCryptoServiceProvider is much better. Again, for full explanation read aeon's thread above.

Anyway, here is the class.

VB.NET - <https://gist.github.com/debug-hf/a08553a0da1f5e1e51b4>

(Thanks to Eternal Darkness)

C# - <https://gist.github.com/debug-hf/1a932d5d3c7270bd57ac>

And thanks to Odus for correcting multiple things <3

EDIT: I have updated the class based on your suggestions. (Although I'm not certain on my implementation of SecureRandom. If it's incorrect let me know.)

Enjoy,

debug

Rhez - February 14, 2016, 3:40 am

Woah. This is new thing for me.

Thanks posting this. ?

Cecilia - February 14, 2016, 4:11 am

It is a nice little wrapper.

Thanks as always :)

B1N4RY - February 14, 2016, 6:18 am

I always knew entropy was a major concern when using a built in "Random" function for any language. Never realized how actually insecure it really was though, nice read. Thanks for sharing!

Ninja - February 14, 2016, 7:49 am

Don't declare anything as new outside of the functions.

Code Private ReadOnly rng As RandomNumberGenerator = New RNGCryptoServiceProvider() You should replace that with:

Code Private ReadOnly rng As RandomNumberGenerator And in the constructor assign the value to it:

Code RandomNumberGenerator = New RNGCryptoServiceProvider()

Temploit - February 14, 2016, 8:37 am

(Feb 14, 2016 08:49 AM)Tyranitar Wrote: [\[CITE\]](#) Don't declare anything as new outside of the functions.

Code Private ReadOnly rng As RandomNumberGenerator = New RNGCryptoServiceProvider() You should replace that with:

Code Private ReadOnly rng As RandomNumberGenerator And in the constructor assign the value to

it:

```
Code RandomNumberGenerator = New RNGCryptoServiceProvider()
```

It seems you have no idea what you're talking about. Directly initializing instance variables compiles down to them being assigned before anything else in the constructor(s).

+Haze - February 14, 2016, 9:20 am

Nice one debug, my nigga :D

Odus - February 14, 2016, 9:33 am

Looks good. Only thing I would say is pad out your code (especially the VB version). You also don't have to evaluate min and max, it will still work even if reversed. If you do, supply the offending param name with a message when throwing the range exception.

Cecilia - February 14, 2016, 2:04 pm

(Feb 14, 2016 08:49 AM)Tyranitar Wrote: [\[CITE\]](#) Don't declare anything as new outside of the functions.

Code Private ReadOnly rng As RandomNumberGenerator = New RNGCryptoServiceProvider() You should replace that with:

Code Private ReadOnly rng As RandomNumberGenerator And in the constructor assign the value to it:

```
Code RandomNumberGenerator = New RNGCryptoServiceProvider()
```

I just laughed more than I should have.

If variables are assigned in a class outside of the constructor, they are added there on compile time without the user knowing that it has been done.

Please, go and open up any project which you have in dnSpy that does a similar thing.

6e756c6c - February 14, 2016, 2:15 pm

Kind of off topic but Tyranitar is right. You should always initialize your default members inside of the constructor, if you let the compiler do it for you then you are forfeiting the ability to define the 'load' order. For instance, if one member relies on another for whatever reason; especially at construction but was automatically initialized first, then you are going to get a null reference exception (if your lucky? It may just silently ignore the error causing some bizarre issues). It's also much more readable (in my opinion) if members are initialized in the constructor as this saves you from searching all over the document. :P

Edit:

Looks awesome debug!

+Haze - February 14, 2016, 2:21 pm

Tbh I think the issue is to do with the way Tyranitar worded his reply as to what I think he intended to say. Rather than saying "Don't declare anything as new outside of the functions." he made it seem like a must, when infact it's sometimes down to preference and if you have overload constructor methods.

Personally, I'm with Tyranitar on the basis that I prefer to initialize my variables inside constructors. I find it more cleaner that way, that's just my preference. And also, when I have overload constructors, it helps to control load order like Aeon mentioned.

Cecilia - February 14, 2016, 2:48 pm

(Feb 14, 2016 03:15 PM)Aeonhack Wrote: [\[CITE\]](#) Kind of off topic but Tyranitar is right. You should always initialize your default members inside of the constructor, if you let the compiler do it for you then you are forfeiting the ability to define the 'load' order. For instance, if one member relies on another for whatever reason; especially at construction but was automatically initialized first, then

you are going to get a null reference exception (if your lucky? It may just silently ignore the error causing some bizarre issues). It's also much more readable (in my opinion) if members are initialized in the constructor as this saves you from searching all over the document. :P

Edit:

Looks awesome debug!

Yes, correct, but there is only 1 field in this class. There are of course issues in much larger projects.

6e756c6c - February 14, 2016, 3:18 pm

(Feb 14, 2016 03:48 PM)LaPanthere Wrote: [\[CITE\]](#) Yes, correct, but there is only 1 field in this class. There are of course issues in much larger projects.

In that case it's an issue of consistency which is just as important. o_o

Ninja - February 14, 2016, 3:43 pm

(Feb 14, 2016 03:04 PM)LaPanthere Wrote: [\[CITE\]](#) I just laughed more than I should have.

If variables are assigned in a class outside of the constructor, they are added there on compile time without the user knowing that it has been done.

Please, go and open up any project which you have in dnSpy that does a similar thing.

(Feb 14, 2016 03:21 PM)+Haze Wrote: [\[CITE\]](#) Tbh I think the issue is to do with the way Tyranitar worded his reply as to what I think he intended to say. Rather than saying "Don't declare anything as new outside of the functions." he made it seem like a must, when infact it's sometimes down to preference and if you have overload constructor methods.

Personally, I'm with Tyranitar on the basis that I prefer to initialize my variables inside constructors. I find it more cleaner that way, that's just my preference. And also, when I have overload constructors, it helps to control load order like Aeon mentioned.

Its just good practice. For this it surely works but its a bad habit if you keep on doing it like this especially in bigger projects or if you work with someone else on a project. I remember doing it the same way as OP instead of in a constructor which caused a shit-storm.

Temploit - February 14, 2016, 3:44 pm

(Feb 14, 2016 04:18 PM)Aeonhack Wrote: [\[CITE\]](#) In that case it's an issue of consistency which is just as important. o_o

I actually have to disagree on both your statements.

First, if you created a situation where load order becomes an issue, then you did it wrong. Knowing how the compiler does things, you should be able to control the load order without problems. If you really need 'recursive' instance variables (1st one uses 2nd one, 2nd one uses 1st one), then rethink your design.

Second, if your instance variable assignments don't change when you overload your constructor, it's simply bad design to put them in one in my opinion. If the value is always the same, it's just a burden to put them in the constructor, buried between other things, while you can just as easily put them after instance variables, making it clear what value it should have. A constructor should only do things with its arguments. If not, it fails single-responsibility. If overloading requires redundant duplication, it also fails 'as little code repeating' (don't know the actual name). It's actually inconsistent to put it in the constructor.

(It doesn't matter what happens after compilation, since most of c# is syntactic sugar anyways)

Ninja - February 14, 2016, 3:48 pm

Also, I noticed you dont declare your variables as a type, for instance:

Code Dim data = New Byte(4 - 1) {}and:

Code Dim randUInt = BitConverter.ToUInt32(data, 0)

EDIT: Just noticed I come off as a bit rude. I dont mean it to degrade you OP (or whatever the word is). Its a nice class and good job nonetheless. I'm just giving constructive criticism.

debug - February 14, 2016, 4:01 pm

(Feb 14, 2016 04:48 PM)Tyranitar Wrote: [\[CITE\]](#) Also, I noticed you dont declare your variables as a type, for instance:

Code Dim data = New Byte(4 - 1) {}and:

Code Dim randUInt = BitConverter.ToUInt32(data, 0)

EDIT: Just noticed I come off as a bit rude. I dont mean it to degrade you OP (or whatever the word is). Its a nice class and good job nonetheless. I'm just giving constructive criticism.No please, criticism is what I like. And if you're refering to the VB class, I just converted it from my C# code, so I'm sure there are mistakes.

+Haze - February 14, 2016, 4:30 pm

(Feb 14, 2016 04:48 PM)Tyranitar Wrote: [\[CITE\]](#) Also, I noticed you dont declare your variables as a type, for instance:

Code Dim data = New Byte(4 - 1) {}and:

Code Dim randUInt = BitConverter.ToUInt32(data, 0)

That's perfectly fine, it's because I assume debug coded this in C# first then converted it. There's nothing wrong with implicit declarations, it's something you see more in c# than vb.net, but I see no issues with it, if the assignment (right side) tells you what data type it is then it's optional if you want to specify the type in the declaration (left side).

Just by looking at the two variables I can tell the type data is a byte and an unsigned 32bit integer.

Eternal Darkness - February 14, 2016, 10:53 pm

(Feb 14, 2016 05:30 PM)+Haze Wrote: [\[CITE\]](#) That's perfectly fine, it's because I assume debug coded this in C# first then converted it. There's nothing wrong with implicit declarations, it's something you see more in c# than vb.net, but I see no issues with it, if the assignment (right side) tells you what data type it is then it's optional if you want to specify the type in the declaration (left side).

Just by looking at the two variables I can tell the type data is a byte and an unsigned 32bit integer. Don't convert C# to VB.Net if you do not understand the differences between them... VB.Net does not have implicit declarations (it allows them but does not pick the correct data type, by default). Expecting the user on the other end to turn on a Infer Option in this case, is quite ridiculous. You should be able to skim through the code... 40 lines of code which is automatically converted for you (by a converter) is too much work to skim/look through. Instead we should teach 50 leechers bad practices, because I need to save less than a literal minute...

(Feb 13, 2016 11:04 PM)debug Wrote: [\[CITE\]](#) Hey all,

After Aeon berated me for using System.Random, I finally decided to stop using it. If you want a good read on why it's not secure, view his thread here: <http://coderrail.com/thread-2.html>

Now, for general purposes, System.Random isn't horrible to use, but RNGCryptoServiceProvider is much better. Again, for full explanation read aeon's thread above.

Anyway, here is the class.

VB.NET - <http://pastebin.com/PLi1JsU>

C# - <http://pastebin.com/aqVshy99>

Enjoy,
debug

```
Code Public Class SecureRandom : Inherits Security.Cryptography.RandomNumberGenerator
    Private ReadOnly rng As New Security.Cryptography.RNGCryptoServiceProvider()
    Private Function GetRandomBytes() As Byte()
        Dim data As Byte() = New Byte(4 - 1) {}
        rng.GetBytes(data)
        Return data
    End Function
    Public Function [Next]() As Integer
        Return BitConverter.ToInt32(GetRandomBytes(), 0) And (Integer.MaxValue - 1)
    End Function
    Public Function [Next](maxValue As Integer) As Integer
        Return [Next](0, maxValue)
    End Function
    Public Function [Next](minValue As Integer, maxValue As Integer) As Integer
        If minValue > maxValue Then Throw New ArgumentOutOfRangeException()
        Return CInt(Math.Floor((minValue + (CDBl(maxValue) - minValue) * NextDouble()))))
    End Function
    Public Function NextDouble() As Double
        Return BitConverter.ToUInt32(GetRandomBytes(), 0) / (UInteger.MaxValue + 1.0)
    End Function
    Public Function NextDouble(minimum As Double, maximum As Double) As Double
        Return NextDouble() * (maximum - minimum) + minimum
    End Function
    Public Overrides Sub GetBytes(data As Byte())
        rng.GetBytes(data)
    End Sub
    Public Overrides Sub GetNonZeroBytes(data As Byte())
        rng.GetNonZeroBytes(data)
    End Sub
End Class
```

Here you go OP. Removed the unnecessary variables. Should work the same... Only this method should be faster and less "costly" in memory... Trade-off is a .0000001 second function call to get a randomized array (this method also removed repetitive code).

When you instantiate in VB.Net use a data type... "Dim data = New Byte(4 - 1) {}" should be "Dim data As Byte() = New Byte(4 - 1) {}". VB variables are not dynamic like C# variables. Assigning a data type in VB is important ([Fix](#)). Lets just say that you "knew this", code should not be shared with bad practices, such as that. Declare a datatype, its not hard. /rant

(Feb 14, 2016 09:37 AM)Temploit Wrote: [\[CITE\]](#) Directly initializing instance variables compiles down to them being assigned before anything else in the constructor(s).

Glad people have started to notice the difference. Although you might as well just use "Private ReadOnly rng As New Security.Cryptography.RNGCryptoServiceProvider()" in this case...

Cecilia - February 15, 2016, 12:04 am

(Feb 14, 2016 11:53 PM)Eternal Darkness Wrote: [\[CITE\]](#) When you instantiate in VB.Net use a data type... "Dim data = New Byte(4 - 1) {}" should be "Dim data As Byte() = New Byte(4 - 1) {}". VB variables are not dynamic like C# variables. Assigning a data type in VB is important ([Fix](#)). Lets just say that you "knew this", code should not be shared with bad practices, such as that. Declare a datatype, its not hard. /rant

Uh...

He's native to c#, he used an online converter to change from c# to vb, which will of course have issues at some point.

He simply provided the VB option so he could post it in this forum, aside from the dead c# section.

I did notice that he used 'var' though, not something I like (even though people say its great) outside of linq...

Eternal Darkness - February 15, 2016, 12:51 am

(Feb 15, 2016 01:04 AM)LaPanthere Wrote: [\[CITE\]](#) He's native to c#, he used an online converter to change from c# to vb, which will of course have issues at some point.

I'm aware. Which is why I said if you cannot detect simple differences in the two different languages, then don't convert it. Because his conversion of the C# code is not equivalent to the C# version.

(Feb 15, 2016 01:04 AM)LaPanthere Wrote: [\[CITE\]](#) He simply provided the VB option so he could post it in this forum, aside from the dead c# section.

The C# subform is not dead. I just corrected code there a few days ago. I was more dead than the C# subforum has been... If it is so inactive and he is native to it, why not post it in there and "revitalize the forum"? You are allowed to post a C# and its shorty conversion in the C# forum, the same way you can post it in the VB forum...

(Feb 15, 2016 01:04 AM)LaPanthere Wrote: [\[CITE\]](#) I did notice that he used 'var' though, not something I like (even though people say its great) outside of linq...

It is very common. People need to know that var and dim are not equal, and that you need to instantiate a data type in VB. If you are posting a "conversion" without it you are hurting the community by teaching people bad habits or giving people misinformation.

I am not just "complaining about it" or w/e. I went through optimized his code, making the changes I was talking about, and just gave him a simple overview of the fact that they were not equal.

Using var is "fine" in C# (handled at compile time, corrected to the right data type). But it absolutely should not end up giving you the inability to declare a data type... That is the point I was making...

Odus - February 15, 2016, 12:52 am

(Feb 15, 2016 01:04 AM)LaPanthere Wrote: [\[CITE\]](#) Uh...

He's native to c#, he used an online converter to change from c# to vb, which will of course have issues at some point.

He simply provided the VB option so he could post it in this forum, aside from the dead c# section.

I did notice that he used 'var' though, not something I like (even though people say its great) outside of linq...

As someone working in .NET, being able to change between them and recognize their differences isn't a bad thing.

Not with you on var, it's beautiful like LINQ. ?

Cecilia - February 15, 2016, 3:53 am

(Feb 15, 2016 01:52 AM)Odus Wrote: [\[CITE\]](#) As someone working in .NET, being able to change between them and recognize their differences isn't a bad thing.

Not with you on var, it's beautiful like LINQ. ?

Fuckin no thanks.

If I know the datatype on the right, I should type it, not use some shitty var.
If I don't, I should use a var or an object.

Odus - February 15, 2016, 4:38 am

(Feb 15, 2016 04:53 AM)LaPanthere Wrote: [\[CITE\]](#) Fuckin no thanks.

If I know the datatype on the right, I should type it, not use some shitty var.

If I don't, I should use a var or an object.

I get the logic. But when its type is immediately apparent on the right, there's no need to write it out.
E.g.

```
Code var secureRnd = new SecureRandom();
```

Reads as "I see a variable named 'secureRnd' and it's an instance of SecureRandom", just the same as:

```
Code SecureRandom secureRnd = new SecureRandom();
```

Equally obvious, but this one feels like I'm repeating myself.

Cecilia - February 15, 2016, 4:51 am

(Feb 15, 2016 05:38 AM)Odus Wrote: [\[CITE\]](#) I get the logic. But when its type is immediately apparent on the right, there's no need to write it out. E.g.

```
Code var secureRnd = new SecureRandom();
```

Reads as "I see a variable named 'secureRnd' and it's an instance of SecureRandom", just the same as:

```
Code SecureRandom secureRnd = new SecureRandom();
```

Equally obvious, but this one feels like I'm repeating myself.

Mhm, I realise its pretty stupid to not use var but I simply never started, and don't want to start :)

DisguiserNET - February 15, 2016, 5:17 am

The only time i use var is when im too lazy to write the declaration, especially for multilayerd dictionaries

+Haze - February 15, 2016, 5:22 am

(Feb 14, 2016 11:53 PM)Eternal Darkness Wrote: [\[CITE\]](#) Don't convert C# to VB.Net if you do not understand the differences between them...

Who are you to tell me I don't know the difference between VB.NET and C#?

Ninja - February 15, 2016, 7:28 am

(Feb 14, 2016 05:30 PM)+Haze Wrote: [\[CITE\]](#) That's perfectly fine, it's because I assume debug coded this in C# first then converted it. There's nothing wrong with implicit declarations, it's something you see more in c# than vb.net, but I see no issues with it, if the assignment (right side) tells you what data type it is then it's optional if you want to specify the type in the declaration (left side).

Just by looking at the two variables I can tell the type data is a byte and an unsigned 32bit integer. I know the compiler eventually makes it all right or something (dont know how to explain it in English) but its just good practice if you are working on a project with other people and they just see like a public property not defined as a type.

+Haze - February 15, 2016, 7:31 am

(Feb 15, 2016 08:28 AM)Tyranitar Wrote: [\[CITE\]](#) if you are working on a project with other people and they just see like a public property not defined as a type.

The whole point of the var keyword is that it can only be used on local variables and not class member variables. You can't use var on a public property, you'd have to specify the type explicitly. I guess in VB when do you not declare it as a type at class level, it's automatically inferred as an object.

Ninja - February 15, 2016, 7:51 am

(Feb 15, 2016 08:31 AM)+Haze Wrote: [\[CITE\]](#) The whole point of the var keyword is that it can only be used on local variables and not class member variables. You can't use var on a public property, you'd have to specify the type explicitly. I guess in VB when do you not declare it as a type at class level, it's automatically inferred as an object.

I know. bad example. My point is just Dim or just var works but it can make things confusing.

Odus - February 15, 2016, 8:06 am

(Feb 15, 2016 08:28 AM)Tyranitar Wrote: [\[CITE\]](#) I know the compiler eventually makes it all right or something (dont know how to explain it in English) but its just good practice if you are working on a project with other people and they just see like a public property not defined as a type.

Then it comes down to poor choice of naming. What a class member is/does should be plain, and by extension, so should the type. Even if not, hover over it and the Quick Info window will show you its location, type and any inline schemas attached.

Ninja - February 15, 2016, 8:07 am

(Feb 15, 2016 09:06 AM)Odus Wrote: [\[CITE\]](#) Then it comes down to poor choice of naming. What a class member is/does should be plain, and by extension, so should the type. Even if not, hover over it and the Quick Info window will show you its location, type and any inline schemas attached.

Oh didn't know that worked for Dim and var. Thanks.

debug - February 15, 2016, 8:07 am

Boy, who would have thought an online conversion would have started this.. Nonetheless, it's more interesting than the usual .NET section discussions.

+Haze - February 15, 2016, 8:10 am

(Feb 15, 2016 09:07 AM)debug Wrote: [\[CITE\]](#) Boy, who would have thought an online conversion would have started this.. Nonetheless, it's more interesting than the usual .NET section discussions. Indeed. Atleast there isn't any flaming, which is always good :)

Ninja - February 15, 2016, 8:13 am

(Feb 15, 2016 09:07 AM)debug Wrote: [\[CITE\]](#) Boy, who would have thought an online conversion would have started this.. Nonetheless, it's more interesting than the usual .NET section discussions. You mean conversation or are you talking about the code conversion? ?

Eternal Darkness - February 15, 2016, 10:03 am

(Feb 15, 2016 06:22 AM)+Haze Wrote: [\[CITE\]](#) Who are you to tell me I don't know the difference between VB.NET and C#?

I did not say that. Read it again. I said if you don't know the difference, and are unable to spend a minute skimming through, don't make the "conversion"... Anyone can paste it into a C# code converter. We don't need more users spreading shitty conversions and learning more bad practices.

It gets real tiring to see leechers improperly define variables, because thats what they see "good coders" are doing. When in reality OP used some inaccurate conversion software, and didn't take one second to look it over. Some shitty conversion is not beneficial to the community. At a minimum, if this is supposed to be cruddy converted code from C#, it should have been posted in the C# forum. You are very much so allowed to post a VB version there as well, although he should note that it is a shorty conversion (Copy and Pasted).

(Feb 15, 2016 08:31 AM)+Haze Wrote: [\[CITE\]](#) The whole point of the var keyword is that it can only be used on local variables and not class member variables. You can't use var on a public property, you'd have to specify the type explicitly. I guess in VB when do you not declare it as a type at class level, it's automatically inferred as an object.

Who are you to decide the point of var variables?

They can be used wherever you want in C#. Type is auto-generated at compile time in C#. ?

"I guess" implies you didn't know the difference. I will overlook that, as I have considered you to be a fairly good coder (No offense but your not DragonHunter/0xDEADDEAD). Why am I overlooking it, it contradicts with the weird statement you made above...

(Feb 15, 2016 09:07 AM)debug Wrote: [\[CITE\]](#) Boy, who would have thought an online conversion would have started this.. Nonetheless, it's more interesting than the usual .NET section discussions. So instead of owning up to not knowing the difference you are deflecting blame to us for telling you... Nice try kid-o...

Also, I like that your just trying to distract us, rather than just closing the thread. Which is a all around better idea for a thread which you deem "off-topic". Even though discussing crappy C# "conversions" is perfectly acceptable in the VB section...

+Haze - February 15, 2016, 10:19 am

(Feb 15, 2016 11:03 AM)Eternal Darkness Wrote: [\[CITE\]](#) I did not say that. Read it again. I said if you don't know the difference, and are unable to spend a minute skimming through, don't make the "conversion"...

Oh okay my bad I see :). I agree.

Quote:Who are you to decide the point of var variables?

They can be used wherever you want in C#.

Oops, I should reword, the point of var imho to save you the time from writing out the data type in the declaration. And no you can't use var everywhere in C#. You can't use var in a class level scope. Excluding ASP.NET(JS) You can only use var on local variables and not class level variables.

Odus - February 15, 2016, 10:21 am

(Feb 15, 2016 11:03 AM)Eternal Darkness Wrote: [\[CITE\]](#) Also, I like that your just trying to distract us, rather than just closing the thread. Which is a all around better idea for a thread which you deem "off-topic". Even though discussing crappy C# "conversions" is perfectly acceptable in the VB section...

The section covers all aspects of .NET, it isn't reserved for VB alone. The child forum is dedicated to C#, but that doesn't mean a project written originally in C# must be posted there with a VB version attached.

Believe me, I'm with you. I always rewrite in the other language when making a dual release.

Menalix - February 15, 2016, 10:44 am

Looks really good.

People should also take care of when taking use of this to generate a string, that the string will be interned in memory, and can easily be found, so remember to use the `SecureString` class.

Eternal Darkness - February 15, 2016, 11:11 am

@OP, Lifetip: If some user comes along and optimizes your code, and fixes your "conversion". Update the OP, remove your bad conversion/whatnot...

(Feb 15, 2016 11:19 AM)+Haze Wrote: [\[CITE\]](#) And no you can't use var everywhere in C#. You can't use var in a class level scope. Excluding ASP.NET(JS) You can only use var on local variables and not class level variables.

Shouldn't have used "everywhere". I meant at class scope (everywhere as in everywhere you can declare a variable, you simply cannot define a variable outside of a class). There is nothing that will happen differently if you use var at a global scope. Although I agree it is designed to be used locally. Just using your question against you to prove a point. Glad that you read my post and understand now though. :)

(Feb 15, 2016 11:21 AM)Odus Wrote: [\[CITE\]](#) The section covers all aspects of .NET, it isn't reserved for VB alone. The child forum is dedicated to C#, but that doesn't mean a project written originally in C# must be posted there with a VB version attached.

Believe me, I'm with you. I always rewrite in the other language when making a dual release. I appreciate you being on my side on the issue. Here is my explanation of the forum thing.

I never said that a conversion should be required. If it is a C# source, that is badly converted, the original should be posted in the C# forum. And maybe just skip the conversion if you aren't going to correct the errors in "conversion".

That is what I was trying to say. It doesn't need to be a dual release. If he wants to post it in C# only (hopefully what OP coded it in) there is nothing wrong with that. I am saying if you want to make a "dual release" or release of C# code in the VB forum, users should do their utmost to convert accurately. Its is as bad of a habit to accept shorty code releases as it is to make them yourself in my eyes. Because in total you are diminishing the average users knowledge about the difference between languages. As it is in the VB forum, the VB code should not be a worse version then the C# code right off the bat (why? because thats the link people will be clicking and learning from).

If something is wrong with the code conversion in the VB forum because they coded it in C# and used a converter (without thinking or giving it a two second looksee), then that should be brought to their attention so they can correct their mistake.

The only reason I brought up the C# forum thing was because LaPanthere said it was would be crazy to post C# in the C# forum these days. If it is C# code you wrote and you cannot be bothered to properly convert it to VB, then it does not belong in the VB forum. It belongs in the C# forum... Hardly gonna neg rep the guy, but I am just stating the fact.

debug - February 15, 2016, 11:31 am

Eternal darkness I don't really care to read your rambling so you can stop addressing me. Anyways, I plan to modify the post with some changes I've made later today.

Eternal Darkness - February 15, 2016, 11:35 am

(Feb 15, 2016 12:31 PM)debug Wrote: [\[CITE\]](#) Eternal darkness I don't really care to read your rambling so you can stop addressing me. Anyways, I plan to modify the post with some changes I've made later today.

Debug Debug Debug.

Not rambling. I am correcting you. ?

Glad to see you will learn from this mistake. That my "ramblings" have taught you something.

I am not forcing you to read my post. If you cannot handle free improvements to your code maybe you shouldn't post it online in the first place.

debug - February 15, 2016, 12:20 pm

(Feb 15, 2016 11:44 AM)Menalix Wrote: [\[CITE\]](#) Looks really good.

People should also take care of when taking use of this to generate a string, that the string will be interned in memory, and can easily be found, so remember to use the securestring class. Yes I am planning on implementing securestring when I update the post a bit later. Thanks for the suggestion. Will be useful for my captcha class as well.

Updated the class. Check OP.

jessman420 - February 19, 2016, 1:11 pm

This ways are very elaborate, most times you can use a simple function like this.

Note: the static declaration makes this object, maintain its last place, so it is like having it outside the function.

```
Code Function EZ_Random(ByVal Min As Integer, ByVal Max As Integer)
    Static RndObj As New Random
    Return RndObj.Next(Min, Max)
End Function
```

debug - February 19, 2016, 4:30 pm

(Feb 19, 2016 02:11 PM)jessman420 Wrote: [\[CITE\]](#) This ways are very elaborate, most times you can use a simple function like this.

Note: the static declaration makes this object, maintain its last place, so it is like having it outside the function.

```
Code Function EZ_Random(ByVal Min As Integer, ByVal Max As Integer)
    Static RndObj As New Random
    Return RndObj.Next(Min, Max)
End Function
```

I think you missed the whole 'secure' part of this class. Read <http://coderrail.com/thread-2.html>