

Forum:[Hack Forums](#)

Thread: [{Idea} Protect Visual Basic/.NET Applications with a native C++ app](#)

unKnown_H4CK3r - December 13, 2010, 11:36 am

Hi!

I had this idea a while ago. Let me explain myself.

You all know that we can decompile (even if the application is packed/obfuscated) and get the source of any Visual Basic or .NET managed applications. More or less easily, depends on which packer/obfuscator/technic you used.

The problem for VB/.NET developers is: protect the application to prevent it from a decompilation.

Also, maybe you know that the native C++ (or C or any other native language) is very harder (almost impossible?) to decompile than VB/.NET.

Ok.. Do you see where I want to drive you?

You create a native ([not .NET](#)) C++ project

You make a .NET program which will run the protected program (I'll explain below), and put it as resource in the C++ program. WE CALL THIS PROGRAM 'A'

Still in a resource file, put the encrypted program (the one you want to protect) (like `Convert.ToBase64String(IO.File.ReadAllBytes("MyFileToProtect.exe"))`). WE CALL THIS PROGRAM 'B'

The C++ exe will drop the program 'A' and run it with passing in arguments the 'B' program (encrypted program, the one you want to protect)

The 'A' program will decrypt the 'B' program (like `Convert.FromBase64String(Environment.GetCommandLineArgs(1))`)

Now, the 'A' program will use reflection to run the program you want to protect (like Spoiler [\(Click to View\)](#)`Code Private Sub ManagedPE(ByVal Bytes As Byte())`

`Dim myAssembly As Assembly = myAssembly.Load(Bytes)`

`myAssembly.EntryPoint.Invoke(RuntimeHelpers.GetObjectValue(RuntimeHelpers.GetObjectValue(myAssembly.CreateInstance(myAssembly.EntryPoint.Name))), New Object() {New String() {"1"}})`
`End Sub`

`Dim myThread As New Threading.Thread(AddressOf ManagedPE)`
`myThread.Start(DecryptedBytes)`

What do you think about it?

I think I'll try to code it during the holidays (in France: at the end of the week).

BravoTwoZero - December 13, 2010, 11:42 am

Wow/Jeez you're creative but yeah would be an good idea, native resource loader.. Good idea :D

µnKn0wn_H4CK3r - December 13, 2010, 11:51 am

(Dec 13, 2010 12:42 PM).Warlock. Wrote: [\[CITE\]](#) Wow/Jeez you're creative but yeah would be an good idea, native resource loader.. Good idea :DThanks.

Yeah, and instead of always encrypt the file and manually create a rc file, we could use UpdateResource API for example (and make a program to automatically protect a program when you click a beautiful button 'Protect !' ?).

Entrepreneur - December 13, 2010, 11:54 am

Sounds like a great idea, i hope you get this coded.

µnKn0wn_H4CK3r - December 13, 2010, 11:58 am

(Dec 13, 2010 12:54 PM)JMK940 Wrote: [\[CITE\]](#) Sounds like a great idea, i hope you get this coded. I'm sure it's possible. I'm not a C++ programmer but [this section](#) exists ? !

Also, if you want to help me with this little project.. Reply here or [PM me](#).

imi-tat0r - December 13, 2010, 12:13 pm

So if the c++ file is dropped, you just gotta unpack the ressources and bam here you got

µnKn0wn_H4CK3r - December 13, 2010, 12:14 pm

(Dec 13, 2010 01:13 PM)imi-tat0r Wrote: [\[CITE\]](#) So if the c++ file is dropped, you just gotta unpack the ressources and bam here you gotAnd BAM read again my post.

The protected file is never dropped.

BravoTwoZero - December 13, 2010, 12:18 pm

It is like running the compiled form but covered in a native compiled box. Simple explanation = right?

µnKn0wn_H4CK3r - December 13, 2010, 12:21 pm

(Dec 13, 2010 01:18 PM).Warlock. Wrote: [\[CITE\]](#) It is like running the compiled form but covered in a native compiled box. Simple explanation = right?Not quite. The native compiled drop a .NET managed that will use reflection to show the .NET managed you want to protect (because the native exe can't do that).

BravoTwoZero - December 13, 2010, 12:24 pm

But the protected file will be running in the memory and not dropped right?

µnKn0wn_H4CK3r - December 13, 2010, 12:25 pm

(Dec 13, 2010 01:24 PM).Warlock. Wrote: [\[CITE\]](#) But the protected file will be running in the memory and not dropped right?Yep, you're right ?

BravoTwoZero - December 13, 2010, 12:31 pm

Phew.... Finally I got it.. It is very complicated at first thought!! :S

Thanks for teaching me !!

µnKn0wn_H4CK3r - December 13, 2010, 12:34 pm

(Dec 13, 2010 01:31 PM).Warlock. Wrote: [\[CITE\]](#) Phew.... Finally I got it.. It is very complicated at first thought!! :S

Thanks for teaching me !!You're welcome ?

More feedbacks ? ?

BravoTwoZero - December 13, 2010, 12:36 pm

(Dec 13, 2010 01:34 PM)BabyHack Wrote: [\[CITE\]](#) You're welcome ?

More feedbacks ? ?From me none!!

To Other : C'mon Comment... let's see

H3x0S - December 13, 2010, 1:02 pm

You will still have to store your .NET file somewhere, and it could be decrypted/extracted from there. This won't work.

xSilent - December 13, 2010, 1:21 pm

I used a loader method with just .NET for world-net that read the bot from resources then decrypted and used reflection, but 98% of coders here won't know what to do with a C compiler.

Nice POC, baby. ? (no homo, I'm saying baby since its in your name)

imi-tat0r - December 13, 2010, 11:53 pm

But the protected file is a ressource of your C++ "Loader" and the Loader is dropped and there you go. Get the ressource of your C++ Loader and decompile

DragonHunter - December 14, 2010, 12:25 am

if you would dump the file somewhere to the hdd the protection is just useless because it's somewhere at the hdd

and if its using runPE you can just attach a debugger to it and dumping the file and protection gone

probably the best way to protect it just like writing ur program in C#/VB whatever and only making prototypes like..

public void lol() { } and thats it... and at the C++ side ull just hook the function 'lol' and putting in there some code for example

```
MessageBoxA(NULL, "this came from C++", "", MB_OK);
```

so when people even would be able to decompile ur .net program those functions will be empty ?

µnKn0wn_H4CK3r - December 14, 2010, 8:42 am

H3x0S Wrote:You will still have to store your .NET file somewhere, and it could be decrypted/extracted from there. This won't work.I'll store the encrypted .NET file in a resource file which will be compiled (C++). It can be decompiled, for sure, but not with a drag & drop..

xSilent Wrote:I used a loader method with just .NET for world-net that read the bot from resources then decrypted and used reflection, but 98% of coders here won't know what to do with a C compiler.

Nice POC, baby. ? (no homo, I'm saying baby since its in your name)Hm, ok. Thanks ? .

imi-tat0r Wrote:But the protected file is a ressource of your C++ "Loader" and the Loader is dropped and there you go. Get the ressource of your C++ Loader and decompileHarder than drag & drop a .NET file in Reflector, right?

DragonHunter Wrote:if you would dump the file somewhere to the hdd the protection is just useless because it's somewhere at the hdd and if its using runPE you can just attach a debugger to it and dumping the file and protection gone

probably the best way to protect it just like writing ur program in C#/VB whatever and only making prototypes like..

public void lol() { } and thats it... and at the C++ side ull just hook the function 'lol' and putting in there some code for example

```
MessageBoxA(NULL, "this came from C++", "", MB_OK);
```

so when people even would be able to decompile ur .net program those functions will be empty ?I won't dump the protected file anywhere. For the 'RunPE', I'm ok.

Then, if you only write the prototypes in C# and you implement the functions in C++, just write the entire program in C++..

This Girl Is On Fire - December 14, 2010, 2:58 pm

If the file is opened in memory, it can easily be dumped from memory. As it would have already been decrypted, then you will have a completely unsafe .net EXE.

µnKn0wn_H4CK3r - December 17, 2010, 10:53 am

I added a pool. Please answer ? !

xCocaine - January 5, 2011, 4:53 am

Zevilone is right, if you run it in memory can be dumped and decrypted in 2 seconds..

µnKn0wn_H4CK3r - January 5, 2011, 5:12 am

Ok, anyway, it's an old thread.

Closed.