

Forum:[Hack Forums](#)

Thread: [C++ Crypter launching .NET exe](#)

Mr CYSM - December 19, 2019, 2:14 pm

Good afternoon together,

I'm currently trying to code my own Crypter in C++. I got it working with running native exe's from RAM. The next step is to launch a .NET exe from RAM. When I tried it with the same approach of running the native exe I got an error message that this application could not be started. I traced all calls to the WinAPI and figured out that mscorEE.dll opens the encrypted file from the hard disk instead of executing the decrypted file in ram. Then I rewrote my stub that it is now hooking all calls to CreateFileW, CreateFileMappingW, MapViewOfFile, ReadFile for all loaded modules. When .NET tries to open the encrypted .exe a pointer to the decrypted payload in the RAM is returned. When I now start my stub I do not get a new error message box with just random strings in title and body. Seems like there are errors with points when called MessageBoxW.

Can anyone tell me what steps are needed to PE load a .NET exe? Or how to debug the CLR internals (problems occur in clr.dll)?

Thank you,

Mr. CYSM

Emp3hack - December 20, 2019, 7:06 am

If you are coding a native c++ application and you want to use .NET then you are looking for CLR hosting. You should find many resources by searching this way.

Ofc if you are already using .net, the CLR will be loaded by default by the windows loader itself when your software is run.

BTW if you want to host CLR in a native crypter, just be aware that it can be considered as a strange behaviour by AVs.

As a side note, I would suggest you to avoid hooking unless strictly necessary for the same obvious reason.

Mr CYSM - December 21, 2019, 12:22 pm

(Dec 20, 2019 08:06 AM)Emp3hack Wrote: [\[CITE\]](#) If you are coding a native c++ application and you want to use .NET then you are looking for CLR hosting. You should find many resources by searching this way.

Ofc if you are already using .net, the CLR will be loaded by default by the windows loader itself when your software is run.

BTW if you want to host CLR in a native crypter, just be aware that it can be considered as a strange behavior by AVs.

As a side note, I would suggest you to avoid hooking unless strictly necessary for the same obvious reason.

Thank you. This is a keyword I could use for further research. If hosting CLR is unusual behavior of a program would it be smarter to code the crypter in .NET? Obviously C++ has the advantage you can not really decompile the code. Are there any more advantages for a native crypter? Or is using .NET the better option even for encrypting native payloads?

Emp3hack - December 22, 2019, 1:39 am

(Dec 21, 2019 01:22 PM)Mr CYSM Wrote: [\[CITE\]](#) Thank you. This is a keyword I could use for further research. If hosting CLR is unusual behavior of a program would it be smarter to code the crypter in .NET? Obviously C++ has the advantage you can not really decompile the code. Are there any more advantages for a native crypter? Or is using .NET the better option even for encrypting native payloads?

Yep, you got it right about the CLR.

The main idea is:

Why a native software should load manually the CLR when it could be written in .NET and let the loader load all the necessary dependencies and routine to start it? This can be flagged as "not typical behaviour" by protection software (and can trigger a detection if other suspicious flag are arisen)

Ofc i'm talking in general, there are many different scenarios when this is needed for a specific reason. There are also many different way to achieve a goal (like CLR hosting) so keep always that in mind when dealing with this sort of software.

Now let's talk about the .NET vs native.

[Biased opinion]

I used both .NET e C++, more exactly I started by using .NET but went with C++ after some time. Ofc both can be used to obtain almost the same result, the main difference here is that native is at lower level of abstraction.

In a similar context (where you need to manipulate process, thread, memory, ..) this can be used for your advantage.

Long story short:

.NET

Improved efficiency and ease of development. Some APIs could be missing and should be imported using PInvoke.

You can easily run your .NET payload without worrying about CLR and so on.

Win32 API in C++:

All APIs are available. Sometimes they are much annoying to use because the underlying complexity is not hidden behind .NET classes.

You will have more tools to get your job done because of the lower abstraction level.

Harder learning curve.

[/Biased opinion]

If I were you I'll just start with the one you are more confident in (or whatever you like more at first glance).

The funny (and useful) part here is not the language choice, but understanding how the mechanism involved in memory, process, thread and kernel management works.

Have fun!