

Forum:[Hack Forums](#)

Thread: [Licensing a Library ?](#)

Vexallus - May 4, 2019, 10:44 am

So I've been working over 3 weeks on an old project which i would like to call as "Codex Library", its something quiet new and i feel like it will get a decent amount of sales if released , The Library does the following [its meant for new users mainly but it still does the job really well for experienced users too] :

Creating a C++ Like Overlay D3D Menu with full options [you can design a full menu in VB or C# by only calling 2 Lines of codes]

Creating a strong licensing system if used within the same app [it has a driver hook which is a lil bit messy & a C++ DLL attached to it , communicated back and fourth within a server or registry depending on the user's preference]

Supports the user with Plenty of Functions for text management , Memory management , obtaining unique info & much more that i am too lazy to write about since this wasn't the reason i made this topic...

my question is...how can i license such a library so that it won't get leaked out for at least until i come up with a nice outcome..., any ideas ?? ? , i thought of using the C++ DLL file which is automatically called once the Library is loaded in order to do just more than .NET stuff, but this can easily be removed if included within the library code itself (I.E reversing the DLL loading func)

Aristocles - May 4, 2019, 6:23 pm

Remote execution. So you need to pass all arguments to an online API and execute the correct method with according parameters.

Today that's the only way; more than ever games are using such anti protect logic, most of them just take advantages of that to do games like Anthem so you don't have offline possibilities. You just need couple of servers to handle the needs of your customers.

Every single other solution is just not gonna provide what you are asking. This is the only solution because the sensitive code will not be on the customer's computer, there is no way to crack something that is not here. Obfuscators, packers, virtualizers, they all keep the code at a comfy place for hackers to use their one click tools.

Vyxn - May 5, 2019, 11:08 pm

You may check CodeElements.net. It has remote code/method execution.

Washi - May 7, 2019, 9:20 am

You have to remember that a licensing system does not equal an anti reverse engineering system.

A licensing system simply provides you with the means to assign a license key to a product that unlocks its functionality. Licenses can then be distributed to people that are privileged to use your software (e.g. by buying a license). This does not mean the licensing system protects your code from intellectual property theft. This is what anti-reverse engineer techniques are for. While they

often go along with each other, they are not equivalent. The question then becomes, what is exactly your goal? Clarifying what exactly is meant by a "leakage" that you want to prevent, will get you a long way.

Is it your intention to hide your code?

Then either an obfuscator, or moving lots of the logic to a remote server, is probably your best bet. The second option provides more security as it is virtually impossible to get your code back out of the server without actually breaking in the server. The obvious downside of course is that you need an active internet connection, and, more importantly, some pieces of code simply cannot be moved to another device (think of code that actually interacts with files or other processes on the user's computer). An obfuscator might then be a better option for you. While this does not provide security (obscurity is definitely not security), it might slow down the adversary enough that it is not worth for them to continue.

Is it your intention to limit the amount of users that can use your software legitimately?

Then you will need a licensing system, that either activates your product offline using some kind of key verification algorithm, or online, where the user authenticates itself and the server authorizes usage or not. The benefit of an online service is that you as the provider have more control on which users are allowed to use it and which are not. The downside again is the obvious requirement of an active internet connection. On the flip side, the weakness of an offline verification is of course that someone could derive the key verification function and generate their own license keys, or even circumvent it entirely by reverse engineering and patching the binaries.

The question then becomes, is it worth spending time on a robust license system in the first place? If you expect to be successful and make a lot of sales anyway, then a few leaked licenses might not be worth all the effort. There is a reason why companies like Adobe do not care all that much if one single user cracks one of their products. They make sales anyway, they will get their profit. Spending too much time on an unbeatable licensing system, especially if it inflicts the usability of the product, is simply not worth the time and money for them.

Is it both?

Then a combination might be the best solution.

Is it none of those?

Then you need to further clarify.