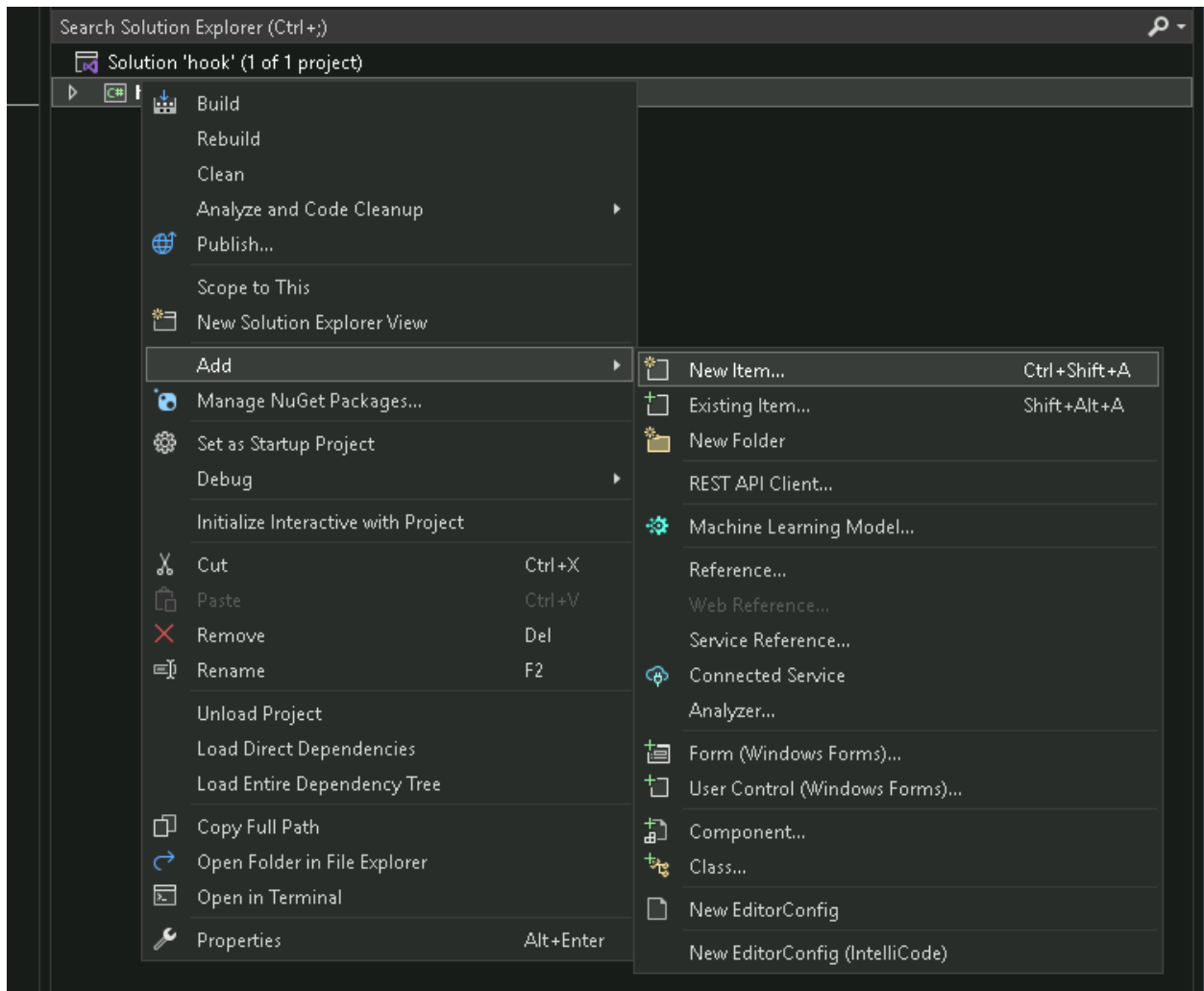


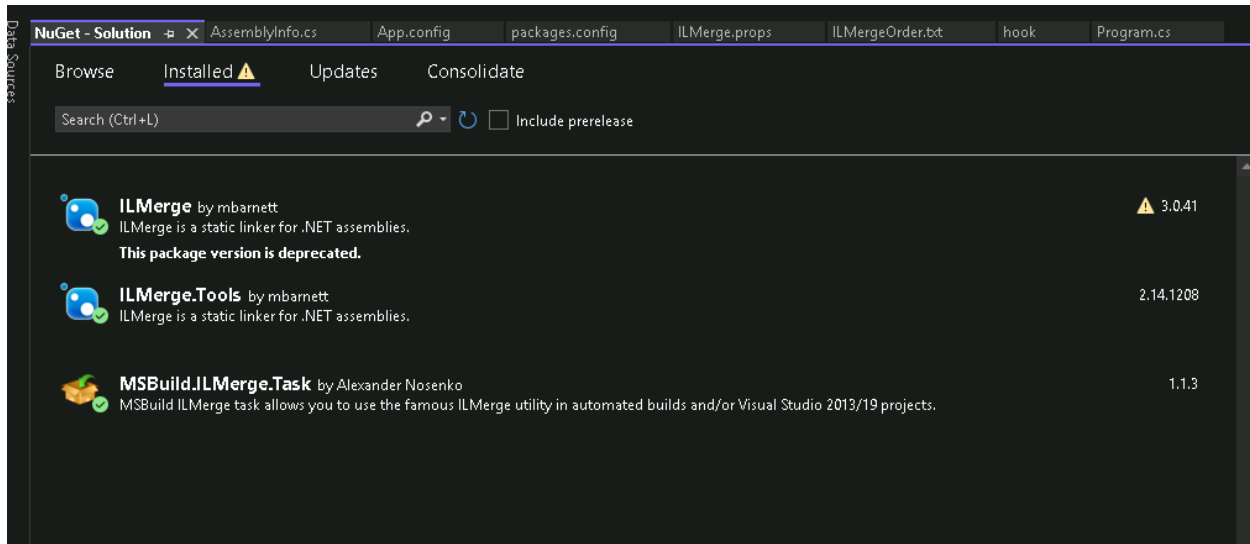
# .NET TradeCraft & D/Invoke

I will save you the time, and not go through D/Invoke too much as its been explained in many other blogs as well. D/Invoke is a tool from the Wover that lets you dynamically Invoke unmanaged code without using P/Invoke in an attempt to evade API hooking. Its been very popular in the MalDev Space lately. I, a noob, ran into a few issues that I saw many others running into as well. I wanted to put this guide together as an intro to getting D/Invoke operational. Due to its popularity, it is heavily sig'd and caught by most A.V engines. Another complaint I see others voice is that they have a loader, and the dll. Many would prefer a single, clean, binary. ILMerge can help!

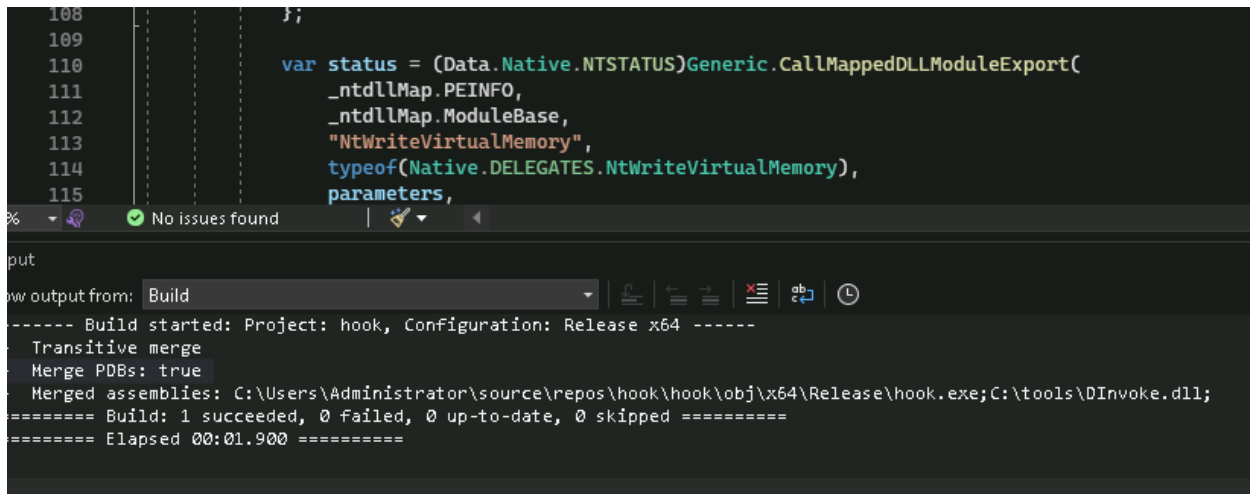
The first thing we want to do is add dinvoke.dll as a reference to our project. You can achieve this by rightclicking the project, going to add>Reference>browse and then choosing the dinvoke.dll file youve downloaded from github.



The next step we want to take is download the ILMerge Nuget Package.



Now we can go ahead and build our solution. You can check in the build output that our two assemblies have been successfully merged. (As a side-note, I started this as a console app .NET FRAMEWORK project).

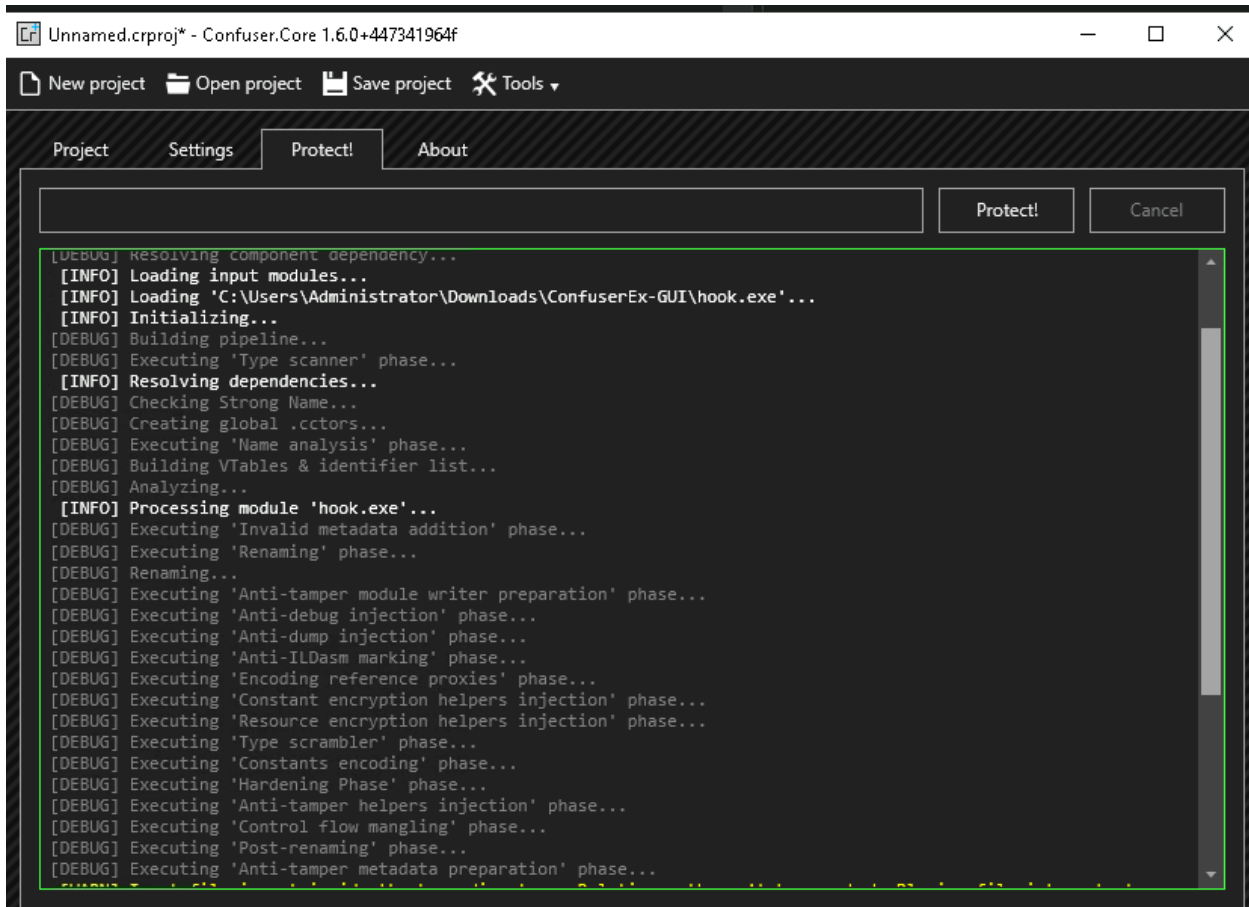


Voila:

```
Directory of C:\Users\Administrator\source\repos\hook\hook\bin\x64\Release
02/13/2023  08:46 PM    <DIR>          .
02/13/2023  08:46 PM    <DIR>          ..
02/13/2023  08:46 PM                83,968 hook.exe
02/13/2023  06:12 PM                189 hook.exe.config
02/13/2023  08:46 PM                54,784 hook.pdb
                3 File(s)      138,941 bytes
                2 Dir(s)  21,833,408,512 bytes free

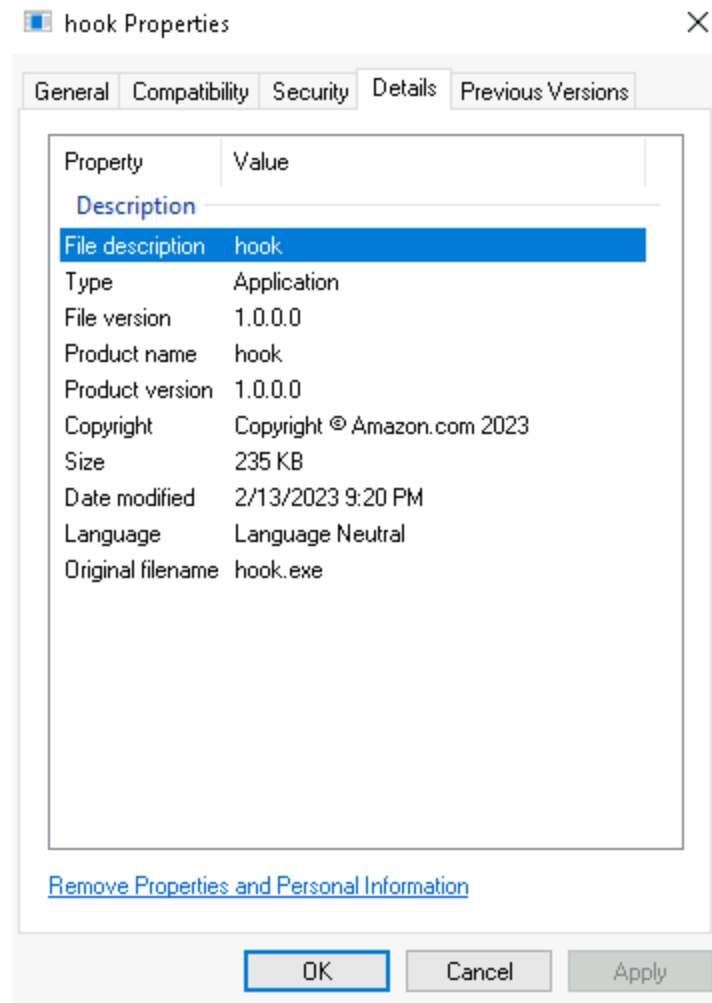
C:\>_
```

Generally when using .NET tools I will use a tool called confuse-ex. It is INCREDIBLE. It offers control flow obfuscation, watermarking, anti-dump etc etc.



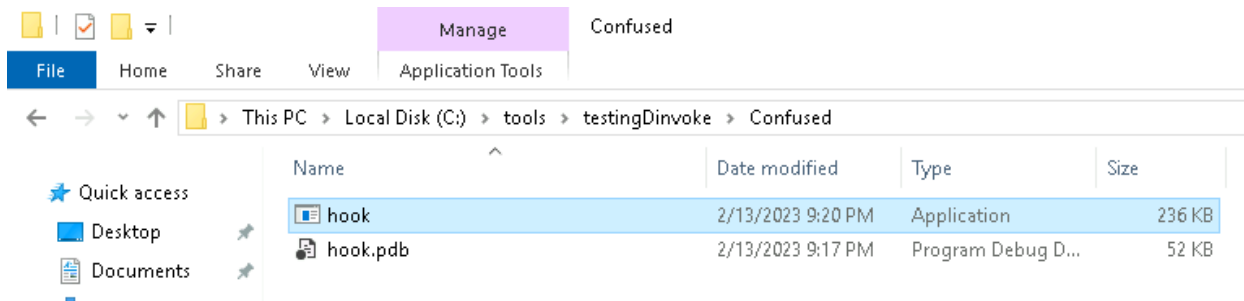
Alright, now we have a basic loader. We've utilized D/Invoke and ILmerge to make one clean binary we can send away.

Lets take a look at our loaders properties and see what it looks like now.

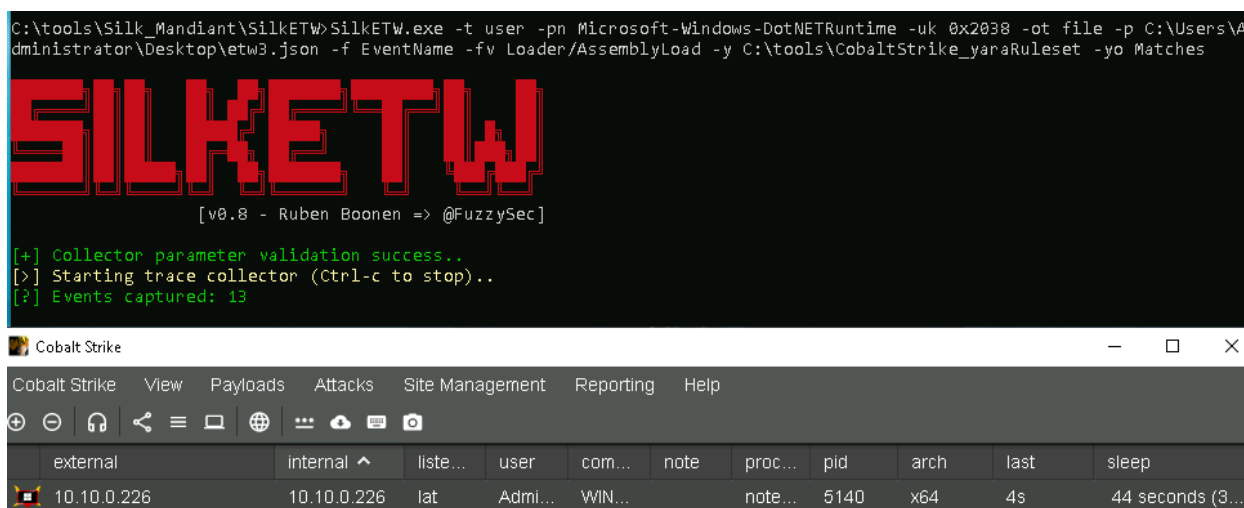


Nice! not bad! (If this was real life, yes, you'd want to put a bit more into the spoofing).

Lets run it



Nice!



Nice! We didnt get a match for image loads on any of our yara rules we have for cobalt strike beacons.

Sources:

<https://github.com/TheWover/DInvoke>

## ConfuserEx 2

ConfuserEx 2 is a free, open-source protector for .NET applications. It is the successor of Confuser project and the ConfuserEx project. Command-line interface Graphical interface

<https://mkaring.github.io/ConfuserEx/>

```
ConfuserEx v2.0.0.0
[INFO] ConfuserEx v2.0.0 Copyright (C) Ki 2014
[INFO] Running on Microsoft Windows NT 5.1.2600 Service Pack 3 .NET Framework
[DEBUG] Discovering plugins...
[INFO] Discovered 10 protections, 1 packers.
[DEBUG] Resolving component dependencies...
[INFO] Loading input modules...
[INFO] Loading 'ICSharpCode.AvalonEdit.dll'...
[INFO] Loading 'ICSharpCode.NRefactory.dll'...
[INFO] Loading 'ICSharpCode.NRefactory.CSharp.dll'...
[INFO] Loading 'ICSharpCode.NRefactory.UI.dll'...
[INFO] Loading 'ILSpy.BamlDecompiler.Plugin.dll'...
[INFO] Loading 'ILSpy.Sharpeye.Isp.dll'...
[INFO] Loading 'ILSpy.exe'...
[INFO] Loading 'ICSharpCode.Decompiler.dll'...
[INFO] Loading 'ICSharpCode.TreeView.dll'...
[INFO] Loading 'Mono.Cecil.dll'...
[INFO] Loading 'Mono.Cecil.Pdb.dll'...
[INFO] Initializing...
[DEBUG] Building pipeline...
[INFO] Resolving dependencies...
[DEBUG] Checking Strong Name...
```

<https://github.com/mandiant/SilkETW>