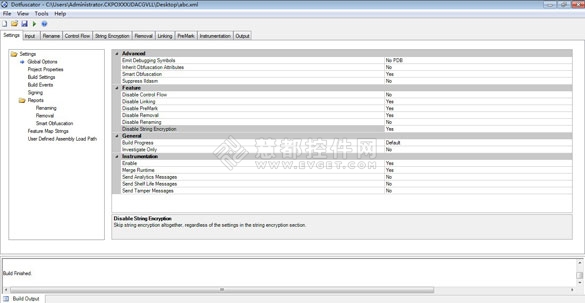
### 前言：

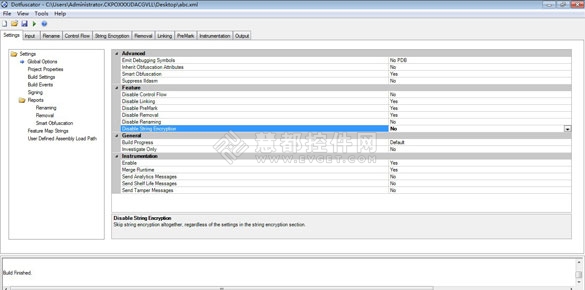
C#编写的代码如果不进行一定程度的混淆和加密，那么是非常容易被反编译进行破解的，特别是对于一些商业用途的C#软件来说，因为盯着的人多，更是极易被攻破。现今很多用C#编写的项目，做出的产品，都会遇到这种情况：C#的代码辛辛苦苦写出来之后，一个反射工具，就可以完全显露出来，这样会是个什么结果，不说被反射出来照搬功能，就是注册和加密的那一部分都会被轻易拿到，所以就必须使用代码混淆器。

C#的代码混淆器有很多了，比如Dotfuscator、xeoncode、foxit等等。这些混淆器大多是需要注意的，使用效果各不相同。普通的代码混淆器只是把私有字段和变量转换成随机码，如果是小程序，还可以猜一猜，如果是比较大的程序，猜起来可就不会那容易了。高级一点的是混淆成不可阅读的字符，这样的混淆后的代码干脆就比机器码还难懂，基本上是翻译不出来了。但这些代码混淆器都有一定的风险，即混淆后的代码实然无法正常调用了。

这里介绍的是[DotfuscatorPro\_4.9.7750](http://software.evget.com/Products/676/feature.html" \t "http://blog.csdn.net/l_serein/article/details/_blank)版本，安装 Dotfuscator 好 打开界面，有几个需要配置的地方： Options、Input、Rename、String Encryption、Build，其他的默认即可，下面一一讲解。 

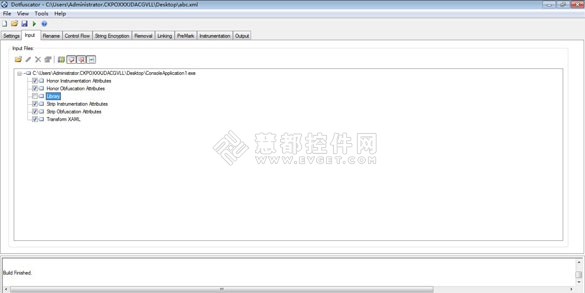
#### Settings->Global Options

这是全局配置选项,把 Disable String Encryption 设为 NO，即启用字符串加密；检查 Disable Control flow、Dsiable Renaming 也为 NO，即保证控制流和混淆功能打开，其他默认。如下图：



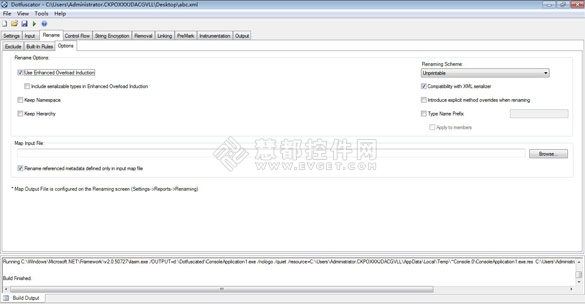
#### Input 选项

选择你需要混淆加密的 dll、exe文件，其中Library不要勾选，否则有些类、变量等等不会混淆。



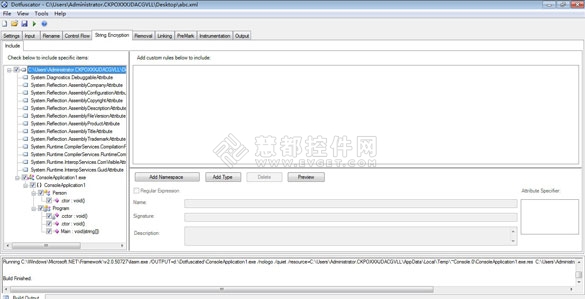
#### Rename 选项

勾选“use enhanced overload induction”（使用增强模式），注意：不要勾选下方的“ Do not suppress on serializable types”，即序列化类型的变量不加密，否则编译后可能异常； “Renaming Scheme” 选择“Unprintable”（不可打印字符，即乱码），也可以选择其他如小写字母、大写字符、数字的方式。



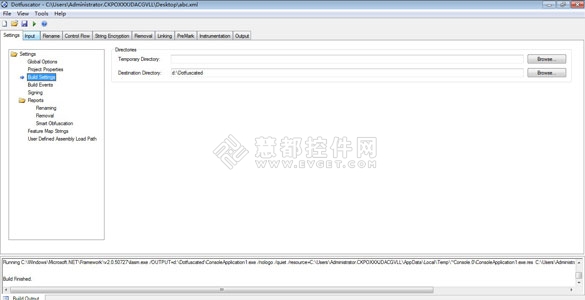
#### String Encryption 选项

include里勾选需要字符串加密的工程文件exe、dll等。



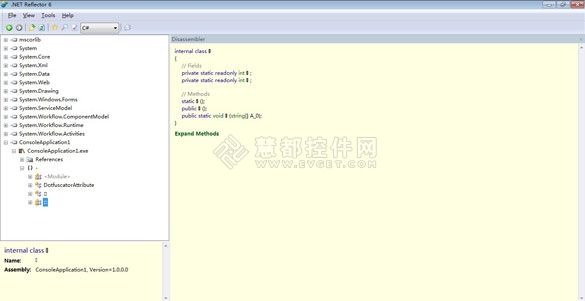
#### Settings->Build Settings 选项

设置输出目标文件夹，点击上方的运行按钮即可重新生成新的 exe、dll文件了。



接下来来看一下一段简单的C#代码加密混淆后的效果：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | **using** System;  **using** System.Collections.Generic;  **using** System.Linq;  **using** System.Text;  **namespace** ConsoleApplication1  {  **class** Program      {  **static** **readonly** **int**  A = B \* 10;  **static** **readonly** **int**  B = 10;          //const int A = B \* 10;          //const int B = 10;  **public** **static** **void**  Main(**string**[] args)          {              Person p = **new** Person();              Console.WriteLine("A is {0},B is {1} ", A, B);          }      }  **public** **class**  Person      {  **static** **int**  age;      }  } |

使用Dotfuscator进行代码混淆后 我们使用著名的****反编译软件 Reflector**** 来查看，可以发现函数体内的代码已经看不到了，字符串也变成了乱码，函数名称也不可见了，效果图如下： 

## 总结：

Dotfuscator作为一款代码混淆的软件，具有很强的反编译功能，但使用时应注意以下几点：

1. 如果你加密的dll、exe里有public类，需要被外部引用的话，那么引用这个public的相关dll、exe也要一起放到Dotfuscator Input 里，同时进行混淆加密，不然外部调用出错。
2. 如果有public类，但是你不想照着第1点的方式做的话，可以勾选 Library 模式（input 里 展开项目，见下图），或者不选择 rename 选项，仅仅用 control flow 混淆（该种方式下会看见类名、函数名，但是看不到函数体）。
3. 如果有public类，但是你也不想按照我的第一点、第二点的方法做的话，可以在rename选项里排除这些public的类名、方法名、属性名即可：使用 rename exclude 里左侧排除项目，或 右侧 正则排除方法。

当然如果你编写的软件不涉及商业用途的话，本着开源的思想，不推荐使用代码混淆器。使用代码混淆器存在一定的风险，在使用前最好做好相关代码备份。

在混淆的时候报错了，错误描述大致如下：

Could not find a compatible version of ildasm to run on assembly C:\xxx.dll This assembly was originally built with .NET Framework v4.0.30319

要混淆不同的.Net Framework版本需要在Settings，Project Properties，添加配置如下。

其中ILASM和ILDASM后面必需要加上对应版本的版本号，然后文件引用对应的版本文件即可。

ILASM\_v4.0.30319

C:\Windows\Microsoft.NET\Framework\v4.0.30319\ilasm.exe

ILDASM\_v4.0.30319

C:\Program Files\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools\ildasm.exe

如图：

