

Protecting your IP on .NET

June 2007, Sheldon Lachambre, Camwood Development Manager

Introduction

Commercial organizations spent days, months and potentially years developing software solutions. The solutions all contain intellectual property (IP) of some sort unless the projects are open source or are intended to be developed for the public domain.

What constitutes IP?

Intellectual property can range from entire code assemblies in .Net right through to single SQL queries. Intellectual property is any form code information and knowledge that a company wants to protect. It is assumed that all IP is based on knowledge and experience that a company may want to keep away from competitors or malicious users of the system.

What is does .Net change the landscape?

Traditionally programming languages like C and C++ are compiled into native binary format, typically executables and dynamic link libraries. The binary files contains machine code which is specific to a given processor architecture and is not portable across to new machine architectures without a recompile and a potentially significant redevelopment. The holds true for all programming languages that compile into native code. There are advantages to this approach although the binary is not portable it is also very difficult to reverse engineer. The assembly code that is produced is very far removed from a high level language like C++ and although guesses can be made during reverse engineering the code produced is not even a reasonable approximation of what the programming code was before compilation.

Microsoft .Net, like Java, bridges this gap by "compiling" into a format very similar to their original source format. This allows portability across different architectures so one could run a .Net application developed on Windows on the Mono framework on Linux.

The truth of the matter is that the code is compiled into an intermediate language. On .Net this is called MSIL (Microsoft Intermediate Language) and this unlike native format is exceptionally easy to reverse engineer and the code produced from the reverse engineering is the same (or very nearly) as the original. Try it – download a tool called Reflector by Lutz Roeder (<http://www.aisto.com/roeder/dotnet/>) and load one of your .Net assemblies in. Select the language to decompile and presto reverse engineered code.

What is obfuscation?

Obfuscation is the process of scrambling the MSIL code into a format that makes is really difficult for a programmer to read, however the code still has to work so it cannot be modified too much.

There are a few basic approaches that most obfuscation providers take:

- Renaming Variables
- Encrypting Strings in the Code
- Changing the Program Flow to make it more difficult to read (think spaghetti code)

It is really important to note that obfuscation does not stop people from “reading” your code however the obfuscation as the name implies makes it really difficult to do so. In a commercial environment code protection with .Net is about stopping casual reverse engineering.

Obfuscation can present problems of its own, especially if you use runtime constructs and introspection like reflection in .Net. Depending on how you use reflection you may need to configure the relevant obfuscator in a different manner to create working code. It may, be in extreme circumstances, necessary to redo sections of code to not use dynamic reflection. These points should be covered in the user guide of any relevant obfuscator that you purchase.

What is code signing?

Code signing is the process of signing your assemblies and Windows installer packages with a digital certificate. The signing of code allows you to prove your identity to the consumer. Should anyone modify your Windows Installer package or any of the assemblies contained within then the signature will no longer be valid and the customer would be able to tell that someone had tampered with it.

What does copy protection do?

Copy protection is an entirely different part of protecting your intellectual property. Copy protection is not so much about protecting your code as it is about protecting one's product from casual piracy. Companies charge for their software solutions in many different ways and have differing licensing requirements. Companies that produce web products that they themselves host may simply restrict access themselves, however it is companies that have applications that are installed within a client environment (in the wild so to speak) that really need to be concerned about stopping unauthorized copies and installations of their software.

Things to look out for when choosing a licensing solution are:

- Ease of use in implementing the solution
- Flexibility of the solution
- Security
- Differing types of protection
- Restricting virtual environments
- Does the solution prevent unauthorized distribution
- Will you need a trial / demo mode?

Conclusion

No code can be fully protected and no licensing technology can provide a perfect and un-crackable system, however a combination of a strong licensing technology and good obfuscation can provide sufficient protection for most enterprises. The important thing to remember is that the more difficult you make it the longer it will take someone to reverse engineer or make unauthorized copies of your solution with diminishing returns.