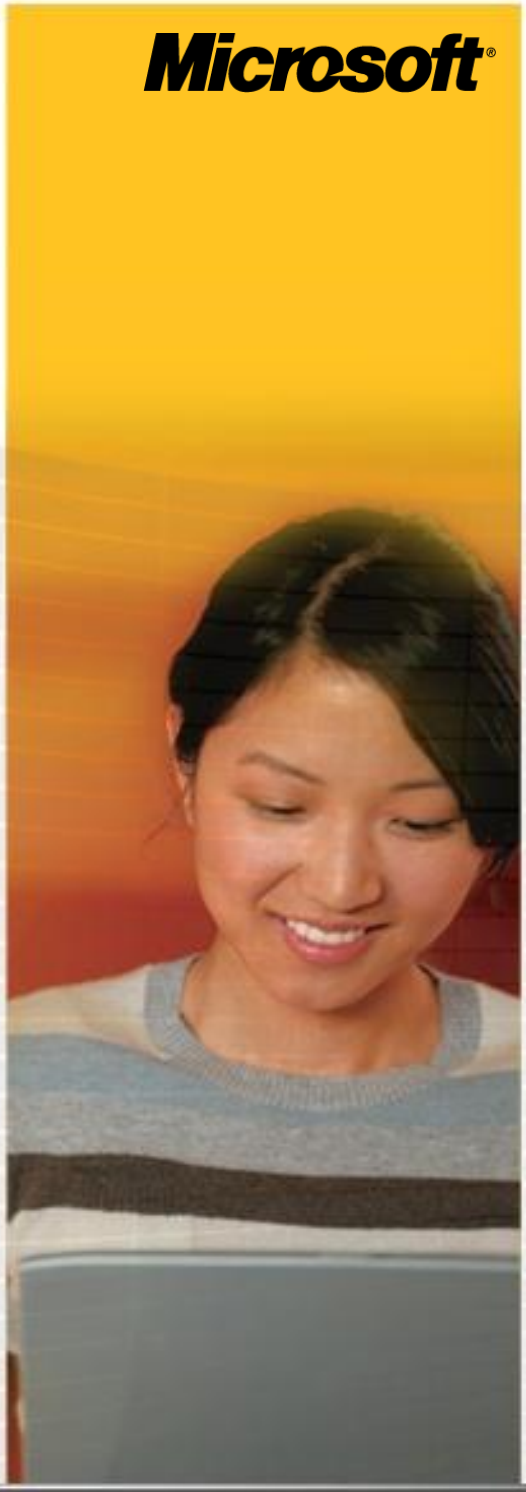
Published: October 2007
For the latest information, please see http://www.microsoft.com/slps

Microsoft®



Microsoft® Software Licensing and Protection (SLP) Services

Fight Back Against Piracy: Protect Your Code

White Paper

Contents

Protecting Your Investment

Your company's software and the source code behind it are at risk—and you may not even be aware of the extent of that risk.

For instance, according to research conducted by the Business Software Alliance (BSA), 35 percent of software installed on personal computers worldwide in 2006 was obtained illegally–at an estimated cost of $40 billion in lost revenue for software publishers.

In the United States alone, one in four software programs are unlicensed. Each year, Independent Software Vendors (ISVs) lose approximately half their realized revenue because of piracy and license noncompliance. *As a result of these risks, a software company that generates $10 million in revenue could experience losses of up to $5 million.*

Hackers and pirates are sophisticated in their attacks, and they are constantly looking for new ways to crack valuable intellectual property, and applications built on the Microsoft® .NET Framework run a particular risk of reverse engineering.

In addition to the potential for piracy, the reverse engineering of software can expose key intellectual property and trade secrets.

Corporate development (CorpDev) teams face similar but additional challenges. They also need to protect the intellectual property within the applications they develop, but they also risk the exposure of information about corporate infrastructure and security.

Hackers and pirates are sophisticated in their attacks, and they are constantly looking for new ways to crack valuable intellectual property, and applications built on the increasingly popular Microsoft® .NET Framework run a particular risk of reverse engineering.

Whether you are an ISV looking to recover revenue lost to piracy or a CorpDev team attempting to secure your intellectual property (IP) and infrastructure, you require easy-to-use, state-of-the-art code protection tools, but may have been disappointed by conventional code protection methods–each of which has weaknesses that can be exploited.

But, Microsoft now offers your company the protection you need from risks of reverse engineering and piracy. First, though, we will look at solutions traditionally used to protect against these threats.

Existing Solutions for Protecting .NET Applications

The Windows® development platform took a major step forward with the introduction of the .NET Framework. Software publishers using Microsoft .NET have benefited from faster development; reduced deployment issues; and a consistent platform across desktop, Web, and mobile applications–including a broad range of programming language choices.

However, the power of this consistent platform comes at a cost when it comes to reverse engineering. As opposed to traditional, 'native' development where high-level languages compile down to machine code, which executes against the CPU, .NET languages compile down to the Microsoft Intermediate Language (MSIL). Contained within the MSIL is all of the information necessary to convert it back into a high-level language, like C# or Microsoft® Visual Basic® .NET. Reverse engineering is like starting with a baked cake and working backward to the original ingredients and the exact recipe for mixing them. Without adequate protection, hackers can read, copy, and, even modify the compiled .NET code.

Reverse engineering is like starting with a baked cake and working backward to the original ingredients and the exact recipe for mixing them.

When your application is ready to go to market or deploy into the enterprise, you need a reliable, full-featured, and convenient way to help secure your software.

Traditionally, there are three methods to help protect your code: *obfuscation*, *encryption*, and *code splitting*.

Obfuscation

Recognizing that compiled .NET code can be easily rendered back into a high-level language, attempts were made to change the MSIL so that, upon reverse engineering, the resulting high-level code would be *obfuscated*–made more difficult to decipher.

Most obfuscation programs perform two tasks:

- Change the meaningful names of classes, methods, parameters, and variables into meaningless text.
- Change the flow of the code so that it has the same end result, but is much more difficult to read.

Again taking the analogy of baking of a cake from a recipe and applying it to obfuscation, the first task is like replacing the names of the ingredients with nonsense names. The second task is like mixing up the steps in the recipe so that the end result is the same, but the baking process is confusing and unnatural.

Obfuscation, though effective at slowing down hackers, possesses inherent weaknesses. First, obfuscation does not prevent decompiling and reverse engineering the code–it does not change the structure or the logic of the code, only makes the code more difficult to understand. Second, defects and other hard-to-track problems can be introduced along with the renaming and flow control changes.

Obfuscation is a good start, but it does not go far enough as a standalone solution to the problem of reverse engineering.

Encryption

If code obfuscation is analogous to changing a recipe to make it more difficult to understand, *encryption* is like putting the recipe into a secure lockbox until it is needed.

There are several tools available to encrypt Microsoft .NET assemblies, each one starts by using an encryption algorithm to encode the .NET modules, which are then decrypted at runtime. The problem here is that the key is delivered along with the lockbox. In order to execute the encrypted MSIL, it must be decrypted before the Common Language Runtime (CLR) can act on it. This leaves an opening for a hacker, who can use automated tools to recover the key and then use the decryption engine to decrypt the code.

Code Splitting

Code splitting is another approach to help protect application code from reverse engineering and typically works by breaking the application code into two pieces. The less-sensitive portion of the code is delivered in its usual form; the more sensitive piece of code is delivered on special external hardware, such as a smart card or a hardware dongle (that is, a security key).

This method has proven effective and offers a higher level of protection relative to other solutions on the market, but it also has several

Obfuscation is a good start, but it does not prevent reverse engineering, just makes it more difficult.

Encryption leaves an opening for hackers.

Code splitting is stronger still, but it can be costly and inefficient.

disadvantages. An external device carries additional costs and can be cumbersome for end users. In addition, the software publisher must manually convert code to the format of the specific device. Pushing out software patches and updates becomes complicated, and online distribution might not be possible. Finally, the code is physically deployed to the user, and though it would require greater skill, it could still be reverse engineered.

Although code splitting is a highly reliable solution, these limitations can make it costly and inefficient.

The SLP Services Solution

Microsoft® Software Licensing and Protection Services offers a family of products designed as a complete solution that addresses the weaknesses in earlier protection mechanisms. It begins with the tools that provide customer-and application-specific code transformations–SLP Code Protector and Permutations–to help protect you software, then goes further to provide a platform for license enforcement and management and product activation with SLP Server 2008 or the SLP Online Service combined with Activation Packs.

Code Protection through SVM Technology

At the heart of SLP Services is an innovative and unique approach to Microsoft .NET code protection: the Secure Virtual Machine (SVM).

The Code Protector application transforms MSIL–which is easy to reverse engineer–into a unique Secure Virtual Machine Language (SVML)–which is not.

As opposed to the method of protecting source code through encryption discussed above–where the encrypted code must be decrypted back into MSIL before it can be executed by the CLR–SLP Services use the Secure Virtual Machine which directly processes the protected code in the form of Secure Virtual Machine Language (SVML). Because the SVML is never converted back into the original MSIL, one significant gap in the protection of software has been closed.

Further, each instance of the SVM is a unique “virtual CPU” that resides inside your application. Because each SVM is unique, each version of the SVML must also be unique. This closes another security hole–if

the SVML for one company was somehow compromised, the security breach would be limited to just that company or application.

This combination of the unique SVM; the unique SVML, which runs on it; and the transformation process, which converts from MSIL into that unique SVML, is called a *Permutation*.

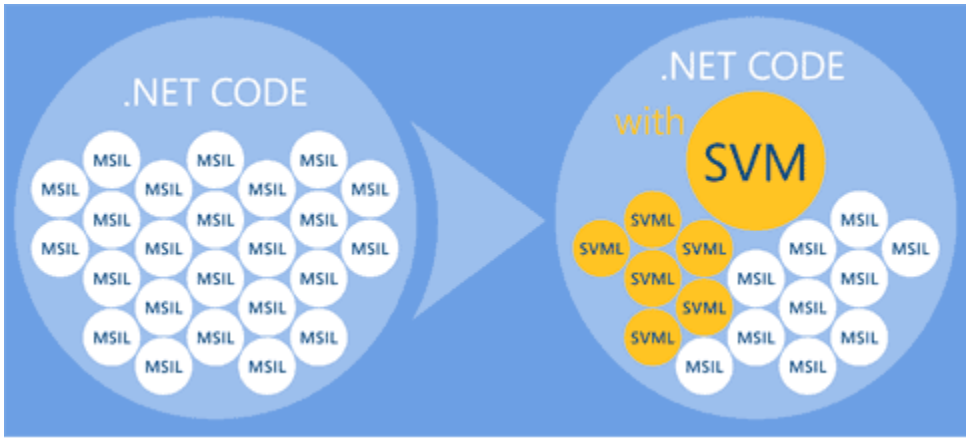
The SLP Code Protector application takes the Permutation and uses it to help protect the classes and methods you specify. In addition, the SVM is inserted directly into the application assemblies. There are no external libraries to be included which can be hacked, nor any embedded keys which can be discovered and used to reverse engineer the protected code.

This selective, one-way code transformation mechanism (Figure 1) provides a greater level of protection for highly sensitive intellectual property. Because transformed code is practically unreadable, there is minimal risk of in-memory code compromise on client machines.

Protecting just the code you need to–like trade secrets, security methods, and clues to your corporate infrastructure--allows you to balance protection with performance.

The SVM not only executes the transformed SVML, but it also acts as the gateway to the protected functionality, enforcing licensing rules, monitoring usage, and managing secure communication to the SLP Services servers and other network components. The SLP Code Protector Software Development Kit (SDK) allows even more precise control, enabling specialized licensing scenarios.

Figure 1. One-way code transformation with SVM



Protection vs. Performance

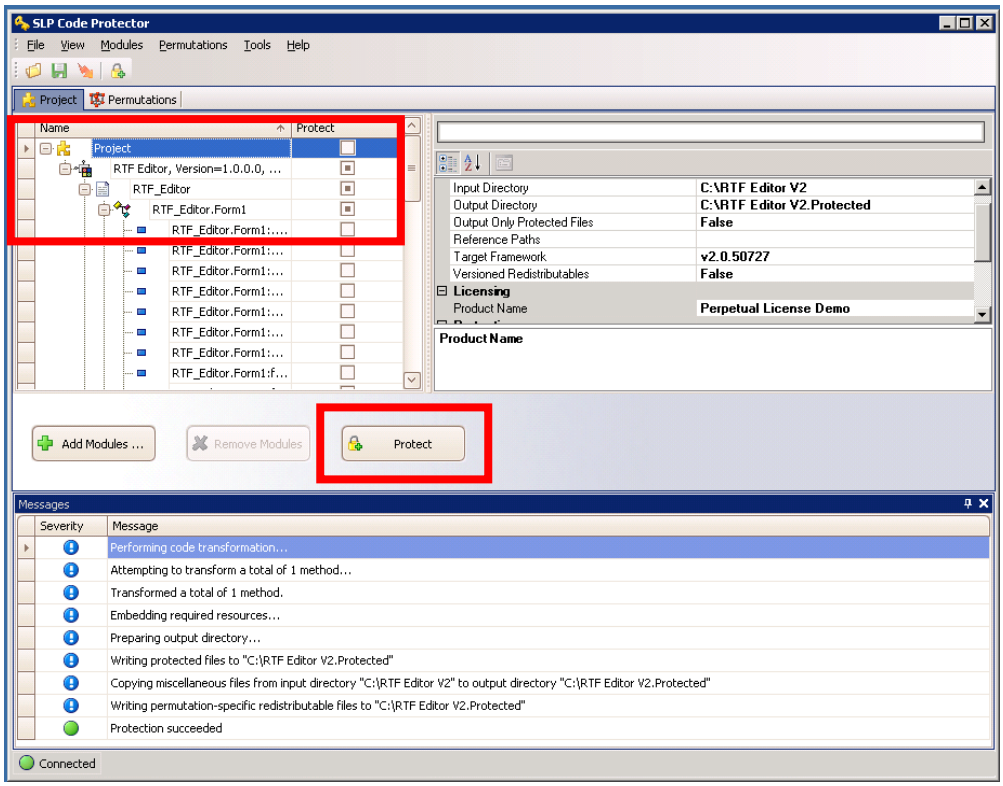
In the earlier analogy about baking a cake from a recipe, it was assumed that you had to protect the entire recipe. Of course, there is a lot of similarity between cake recipes, and it is unnecessary to protect the entire recipe, just those parts of it that make it unique. This would do little to reduce the security of the recipe, but makes it much faster to read–only those secret ingredients need to be decrypted.

Similarly, because the SVM needs to interpret the SVML code, and runs on top of the CLR, there is a performance element to the equation that needs to be addressed. You do not want to protect the entire code base, because it would slow the whole application down and add little to overall security. Instead, you want to protect only what is necessary: the secret ingredient.

Protecting just the code you need to–like trade secrets, security methods, and clues to your corporate infrastructure–allows you to balance protection with performance.

Figure 2 shows how easy it is to select the classes and methods to be transformed from the original .NET MSIL and into the Secure Virtual Machine Language.

Figure 2. Easily select which functions and features to protect



The ability to specify exactly what code to transform permits the balancing of protection against performance to reduce overhead in ways that no other code protection mechanism allows. When selecting code to protect, choose those classes and methods possessing a high intellectual property component, or that are particularly vulnerable to hacking or reverse engineering, including the following:

- Staging or initialization methods that permit access to functionality you want to control.
- Methods that enforce licensing.
- Code that implements algorithms unique to the product.
- Code that contains information about the infrastructure of the enterprise: database connections, passwords, etc.

SLP Services Code Protection Road Map

SLP Services is based on field-tested, proven technology. The initial release of SLP Services delivers significant functionality for the protection of source code and the enforcement of licenses with a scalable, robust infrastructure. Future improvements will focus on enabling integration with Enterprise Resource Planning and Customer Relationship Management systems, improving performance, and streamlining the workflow experience.

The SLP Code Protector SDK will be shipped as part of the Microsoft® Visual Studio® 2008 release, and future versions will feature tighter integration with the Visual Studio integrated development environment.

The Value of Protecting your Software

SLP Services can help ISVs and CorpDev teams protect valuable intellectual property against piracy, hacking, reverse engineering, and misuse. By helping to both increase the security of code and provide a robust licensing infrastructure, SLP Services provides greater peace of mind when distributing products to customers and end users.

With an estimated $40 billion in lost revenue worldwide, decreasing piracy is an effort that pays for itself.

For More Information

Please refer to the following resources for more information on the topics covered in this document and for related topics:

Microsoft Software Licensing and Protection Services Web site: www.microsoft.com/slps.

[We will include references to our other white papers as we finish them]

For specific questions, contact us at slpsinfo@microsoft.com.