.NET Framework開発用
ソースコード保護ツール

Dotfuscator Professional Edition Ver1.2

本田 真規 HONDA, Masaki

Software Information

OS	Windows 2000/XP
動作環境	.NET Framework 1.1
価格	304,500円

問合先

株式会社エージーテック
TEL ：03-3293-5283
FAX ：03-3293-5270
URL ：http://www.agtech.co.jp/products/preemptive/dotfuscator/
MAIL：info@agtech.co.jp

はじめに

.NET Frameworkのポータブル実行可能（PE）ファイルはCPUに依存しない中間言語MSIL（Microsoft Intermediate Language）とメタデータによって構成されています。

メタデータには、依存している他のアセンブリを利用するのに必要なすべての情報やコードに定義されているすべての型およびメンバなどが言語に中立的な形で記述されています。このメタデータによって言語間、コンポーネント間の相互運用の簡略化、バージョン管理や配置の手間の軽減などの利点を得られるようになりました。

また、MSILはほとんどのCPUの機械語よりも格段に高水準で配列の直接操作、例外処理、オブジェクトの型の判別などができます。高水準であるがゆえにMSILは非常に読みやすく理解しやすいコードです。

事実、.NET Framework SDKに含まれるILDASM.exe（MSIL逆アセンブラ）や逆コンパイラを使用すると簡単に判読可能なレベルにリバースエンジニアリング、つまり簡単にアセンブリ内のアルゴリズムを盗用したり、セキュリティの欠陥をつくことができてしまいます。これは.NETの欠陥ではなくコンパイル済みの中間言語を使用するプラットフォーム、たとえばJavaで記述されたアプリケーションにも共通して見られる問題です。

しかし、これらの知的財産の盗難という脅威は.NETアプリケーションを開発して配布する際に問題となるため、十分に対策を検討する必要があります。

この問題に対して、“アセンブリの隠蔽”という形で対処するツールが今回ご紹介する「Dotfuscator Professional Edition」（以下Dotfuscator Pro）です。

Technology Tools

- ◯ Visual Basic
- ✓ Visual C#
- ◯ Visual C++
- ◯ SQL Server
- ◯ Oracle
- ◯ Access
- ◯ ASP.NET
- ◯ Other:

アセンブリの隠蔽

アセンブリの隠蔽とは、さまざまな手法を使用してロジックの意味を保ちながら理解不能にすることです。アセンブリの隠蔽を正しく行なうことで、アプリケーションの機能を保ちながら人間が見ても解釈できず、逆コンパイラを失敗させることができます。

たとえば、隠蔽によってロジックを保ちながら逆コンパイルの可能性が複数存在するように変更できます。この状態でプログラミング言語にリバースエンジニアリングしようとすると、逆コンパイラにはどれが正しい意味に再コンパイルされるか認識できないために変換があいまいになり失敗します。

通常、手作業で解析して逆コンパイルすることが大変なために、逆コンパイラを記述して自動化します。その逆コンパイラを混乱させるほどですから、逆コンパイラよりも能力が劣る人間に対しても効果があると言えます。

隠蔽という作業の結果、ロジックは

容易に解読できなくなりますが、これは暗号化とは異なります。隠蔽することで（暗号化でも同様ですが）完璧な安全性を確保できるというわけではありません。現実的な時間内に可能かどうかは別として、時間をかければ元のソースコードにたどり着くことが可能です。それなら暗号化のほうが安全と思われるかもしれませんが、これはロジックの保護という問題を鍵の保護という問題に置き換えただけです。しかも鍵が漏れてしまうと完全に逆コンパイルできてしまうことを考慮しなければなりません。

このことから考えると、逆コンパイラも人間も混乱させることができる隠蔽のほうがアセンブリの保護という観点からは適切であるといえるでしょう。

Dotfuscator Professional Editionの機能

Dotfuscator Proは次のような機能を提供しています。Visual Studio .NET 2003にバンドルされている機能限定版のCommunity Editionでは①～③のみ使用することができます。

①複数アセンブリにまたがる難読化
②不使用メタデータの除去
③名前の変更
④拡張オーバーロード誘導
⑤文字列の暗号化
⑥制御フローの難読化
⑦ILDASMの打破
⑧リネームプレフィックス機能
⑨圧縮／不要コード除去
⑩.NET Compact Frameworkに対する包括的なサポート
⑪増分難読化（インクリメンタルな難読化）
⑫サテライトDLLに対するシームレスな難読化
⑬Managed C++アセンブリのサポート

ページ数の都合上すべての機能を紹介することはできませんが、Dotfuscator Proの代表的な機能を見ていきましょう。

名前の変更

名前の変更とは、クラス、メソッド、フィールドなど識別子の名前を変更します。このとき、できるだけ短い識別子を使用することで逆コンパイル時の出力をわかりにくくすると共に、実行可能ファイルのサイズを小さくする効果があります。Community Editionでは英小文字のみ使用できますが、Professional Editionでは英小文字に加えて、英大文字、数字、印刷不能文字を使用することができるため、さらに出力結果がわかりにくくなります。

開発を行なう場合は通常、コーディング規約などで識別子には「できる限りわかりやすい名前を付ける」ことが求められます。これは保守する場合などに名前から意味を判断できるようにする意図があるわけですが、逆コンパイル時にも機能の内容が推測できてしまうという問題があります。Dotfuscator Proではこのような保守性のためのコーディング規約の逆を実行することでコードの意味を容易に判断できないようにします。

例を挙げると、ログイン時のパスワードチェック機能を提供するメソッドが次のように利用できるとします。

```
Login.CheckPassword();
```

このコードに対してDotfuscator Proで名前の変更を実施すると、

```
a.a();
```

のように変換されるため、変換後の名前を見て機能を推測することができなくなります。

それでは実際にDotfuscator Proを使

リスト1：名前の変更機能テスト用サンプル（sample1.cs）

```
using System;

namespace DotfuscatorSample
{
  public class Sample1
  {
    public bool ToBoolean(string value)
    {
      return Convert.ToBoolean(value);
    }

    public int ToInt32(string value)
    {
      return Convert.ToInt32(value);
    }

    public double ToDouble(string value)
    {
      return Convert.ToDouble(value);
    }

    public static void Main()
    {
      Sample1 s = new Sample1();
      Console.WriteLine(s.ToBoolean("False"));
      Console.WriteLine(s.ToInt32("123"));
      Console.WriteLine(s.ToDouble("1.234"));
    }
  }
}
```