Zugriffsmodifizierer erfolgreich umgehen

Verstecken sinnlos

Wollten Sie auch schon einmal eine nicht öffentliche Methode einer fremden Klasse aufrufen? Oder haben sich darüber geärgert, dass eine Klasse keinen öffentlichen Konstruktor zur Verfügung stellt? Gewusst wie! dotnetpro zeigt, wie Sie private Methoden ohne direkte Zugriffsberechtigung aufrufen können. Das funktioniert sogar in nicht öffentlichen Klassen ohne öffentlichen Konstruktor.

Wussten Sie eigentlich, dass die Zugriffsmodifizierer *public*, *private* und *internal* – beziehungsweise *Friend* in Visual Basic .NET – nur von den Sprach-Compilern für C# und VB.NET erzwungen werden, aber nicht auf IL-Ebene repräsentiert werden?

Dies bedeutet: Sie können jedes als *private* oder *internal* deklarierte Mitglied aus einer beliebigen Klasse einer beliebigen Assembly aufrufen, indem Sie Reflection verwenden. Grundlageninformationen zu Reflection finden Sie unter [1].

Nehmen wir an, Sie bekommen eine kompilierte Assembly *InvokePrivateMembers.dll* mit einer Klasse *Calculations*, welche eine statische Methode namens *CalcStatic* enthält, die Sie verwenden möchten. Der Autor der Assembly hat die Klasse mit *internal* nur für die Verwendung innerhalb seiner Assembly freigegeben und die Methode *CalcStatic* als privat deklariert. Aus diesem Grund taucht die Klasse nicht im Objekt-Browser auf und kann somit auch nicht mittels *new*-Schlüsselwort erzeugt werden. Listing 1 zeigt den Code.

Listing 1

Nicht öffentliche Klasse mit privater statischer Methode.

```
// Die Klasse steht nur innerhalb dieser
// Assembly zur Verfügung.
internal class Calculations
{
  // Diese Methode kann nicht von außen
  // aufgerufen werden.
  private static int CalcStatic(int a, int b)
  {
    return a + b;
  }
}
```

Aus irgendeinem Grund benötigen Sie aber genau diese Funktionalität. Ein Lösungsansatz wäre, mithilfe von Tools wie .NET Reflector [2] die Methode zu dekompilieren. Das funktioniert aber nicht immer, und wenn sich die Implementierung der Methode ändert, weil eine neue Version der Assembly vorliegt, müssen Sie wieder manuell dekompilieren. Diesen Aufwand können Sie vermeiden, indem Sie die private Methode direkt aufrufen. Da die Sprach-Compiler dies nun aus nachvollziehbaren Gründen verbieten, muss ein Weg gefunden werden, die Methode aufzurufen, ohne dass der Compiler merkt, dass er hintergangen wird. Also: Reflection. Reflection ermöglicht es, beliebige Mitglieder auszuführen, solange der Name und gegebenenfalls Parameter und Rückgabewert bekannt sind.

Listing 2 löst das beschriebene Problem an dem genannten Beispiel.

Die ersten zwei Zeilen ermitteln den Typ der Klasse, von der die private statische Methode aufgerufen werden soll. Dieser Typ wird in einem *System.Type*-Objekt repräsentiert. Solange sich die Assembly im gleichen Ordner wie die Anwendung befindet, reicht hier die Angabe des Namens ohne die Endung „.dll“ aus. Von dem gewonnenen *Assembly*-Objekt wird die Methode *GetType* aufgerufen. *GetType* erhält als Parameter den vollständigen Namen der Klasse, also Namespace und Klassennamen getrennt durch einen Punkt. *GetType* gibt das gewünschte *Type*-Objekt zurück.

Die Methode *CalcStatic*, die aufgerufen werden soll, erfordert zwei Parameter. Dazu wird ein Objekt-Array vorbereitet, in dem die zwei Parameter *a* und *b* enthalten sind. Zum Schluss wird vom *Type*-Objekt die Methode *InvokeMember* aufgerufen.

Listing 2

Die private Methode unter Umgehung des Zugriffsschutzes aufrufen.

```
Assembly asm = Assembly.Load("InvokePrivateMembers");
Type t = asm.GetType("InvokePrivateMembers.Calculations");

object[] args = new object[] { a, b };
int ergebnis = (int)(t.InvokeMember("CalcStatic", BindingFlags.DeclaredOnly |
    BindingFlags.Static | BindingFlags.Public | BindingFlags.NonPublic |
    BindingFlags.InvokeMethod, null, null, args));
```

Auf einen Blick

Autor

Neno Loje ist Microsoft Student Partner (MSP) an der Universität Hamburg und Programmierer bei der KEEP IT SIMPLE GmbH in Hamburg. Bei Fragen erreichen Sie ihn über seine Website www.dotnet-online.de.

dotnetpro.code
A0407PrivateMethod

Sprachen C#, VB.NET

Technik Reflection

Voraussetzungen .NET SDK 1.1

Den vollständigen Artikel lesen Sie in:

dotnetpro 7/2004 auf Seite 90

dotnetpro-Abonnenten können diesen über das Online-Archiv herunterladen:

http://www.dotnetpro.de/articles/onlinearticle1373.aspx