Обфускация

Символьная обфускация

Символьная обфускация — наиболее простой способ запутать злоумышленника, анализирующего код. Она позволяет переименовать имена типов, событий, свойств, методов. При этом может использовать непечатный набор символов, что очень помешает т.н. статическому анализу. Данная операция необратима, и код получается очень трудночитаемый.

Результат работы символьный обфускации MSIL кода:

До защиты:

```
.method private hidebysig instance void  ExecuteUndo() cil managed {
  // Code size  81 (0x51)
  .maxstack  2
  .locals init (class Calculator.Calc/Stack V_0, string V_1)
  00: ldarg.0
  01: ldfld    class Calculator.Calc/Stack Calculator.Calc::stack
  06: stloc.0
  07: ldarg.0
  08: ldarg.0
  09: ldfld    class Calculator.Calc/Stack Calculator.Calc::undoStack
  0e: stfld    class Calculator.Calc/Stack Calculator.Calc::stack
  13: ldarg.0
  14: ldloc.0
  15: stfld    class Calculator.Calc/Stack  Calculator.Calc::undoStack
  1a: ldarg.0
  1b: ldfld    class [System.Windows.Forms]System.Windows.Forms.TextBox
               Calculator.Calc::inputTextBox
  20: callvirt instance string
               [System.Windows.Forms]System.Windows.Forms.Control::get_Text()
  25: stloc.1
  26: ldarg.0
  27: ldfld    class [System.Windows.Forms]System.Windows.Forms.TextBox
               Calculator.Calc::inputTextBox
  2c: ldarg.0
  2d: ldfld    string Calculator.Calc::undoInputString
  32: callvirt instance void
               [System.Windows.Forms]System.Windows.Forms.Control::set_Text(string)
  37: ldarg.0
  38: ldloc.1
  39: stfld    string Calculator.Calc::undoInputString
  3e: ldarg.0
  3f: call     instance void Calculator.Calc::UpdateStackTextBox()
  44: ldarg.0
  45: call    instance void Calculator.Calc::UpdateMemoryMenu()
  4a: ldarg.0
  4b: call    instance void Calculator.Calc::Restore()
  50: ret
} // end of method Calc::ExecuteUndo
```

После защиты:

```
.method private hidebysig instance void  'ı'() cil managed {
  // Code size 81 (0x51)
  .maxstack  3
  .locals init (class 'ı'/'I' V_0, string V_1)
  IL_0000: ldarg.0
  IL_0001: ldfld    class 'ı'/'I' 'ı'::'I'
  IL_0006: stloc.0
  IL_0007: ldarg.0
  IL_0008: ldarg.0
  IL_0009: ldfld    class 'ı'/'I' 'ı'::'I'
  IL_000e: stfld    class 'ı'/'I' 'ı'::'I'
  IL_0013: ldarg.0
  IL_0014: ldloc.0
  IL_0015: stfld    class 'ı'/'I' 'ı'::'I'
  IL_001a: ldarg.0
  IL_001b: ldfld    class
                    [System.Windows.Forms]System.Windows.Forms.TextBox 'ı'::'I'
  IL_0020: callvirt instance string
                    [System.Windows.Forms]System.Windows.Forms.Control::get_Text()
  IL_0025: stloc.1
  IL_0026: ldarg.0
  IL_0027: ldfld    class
                    [System.Windows.Forms]System.Windows.Forms.TextBox 'ı'::'I'
  IL_002c: ldarg.0
  IL_002d: ldfld    string 'ı'::'I'
  IL_0032: callvirt instance void
                    [System.Windows.Forms]System.Windows.Forms.Control::set_Text(string)
  IL_0037: ldarg.0
  IL_0038: ldloc.1
  IL_0039: stfld    string 'ı'::'I'
  IL_003e: ldarg.0
  IL_003f: call     instance void 'ı'::'I'()
  IL_0044: ldarg.0
  IL_0045: call     instance void 'ı'::'I'()
  IL_004a: ldarg.0
  IL_004b: call     instance void 'ı'::'U'()
  IL_0050: ret
} // end of method 'ı'::'ı'
```

Важно!
Будьте внимательны, если вы используете **Reflection API.** Типы и методы, доступ к которым происходит с его использованием, обфусцировать нельзя.

Шифрование строк

В процессе обфускации все текстовые строки в исходном тексте зашифровываются с использованием электронного ключа и помещаются в защищенный **native–контейнер.** Метод применяется для сокрытия важной информации, которая может заинтересовать злоумышленника или поможет ему найти нужный фрагмент кода. Используя **Reflection API** для вызова функций, данный механизм позволяет зашифровать имена методов и типов.

Результат работы режима шифрования строк:

До защиты:

```
// .Net App TestApp
void Test() {
  .......
  ldstr 'Str1'
  .......
  ldstr 'Str2'
  .......
}
```

После защиты:

```
// .Net App TestApp
void Test() {
  .......
  ldc,i4 0x35765f1
  call string
  T:LoadString(int32)
  .......
  ldc,i4 0x35765f1
  call string
  T:LoadString(int32)
  .......
}
```

Обфускация графа потока управления

Обфускация графа потока управления (начиная с версии ***Guardant SDK 6.0***) предоставляет возможность разбить код методов на несколько блоков, добавить фальшивые и скрыть от анализа последовательность их исполнения. Последовательность исполнения блоков в этом случае будет зашифрована с использованием электронного ключа Guardant. При этом в реальных функциях количество блоков легко превышает 10, делая невозможным статический анализ кода.

Результат работы режима обфускации графа потока управления:

До защиты:

```
// .Net App TestApp
void Test() {
  блок кода 1
  блок кода 2
  ret
}
```

После защиты:

```
// .Net App TestApp
void Test() {
    ldc i4 0x16576af
    stlock.2
    brl3
l1: блок кода 2
    ldc i4 0x14611f
    stlock.2
    brl3
l2: блок кода 1
    ldc i4 0x6771df
    stlock.2
    brl3
l3:
    ldlock.2
    call int32
    T::GetBlockID(int32)
    switch(l1,l2,l3,l4)
l4:
    ret
}
```