Информационные технологии

УДК 004.056.5

Пимкин Максим Евгеньевич
ООО «ОСМОКОД»
г. Москва, Россия
E-mail: wowmaks@bk.ru
Евдокимова Виктория Вадимовна
Амурский государственный университет
г. Благовещенск, Россия
E-mail: vika.evdokimova.0014@inbox.ru
Фомин Денис Васильевич
Амурский государственный университет
г. Благовещенск, Россия
E-mail: gefest-uni@yandex.ru
Pimkin Maxim Evgenievich
LLC OSMOCODE
Moscow, Russia
E-mail: wowmaks@bk.ru
Evdokimova Victoria Vadimovna
Amur State University
Blagoveschensk, Russia
E-mail: vika.evdokimova.0014@inbox.ru
Fomin Denis Vasilyevich
Amur State University
Blagoveschensk, Russia
E-mail: gefest-uni@yandex.ru

ЛЕКСИЧЕСКАЯ ОБФУСКАЦИЯ ЯЗЫКОВ .NET НА УРОВНЕ IL-КОДА

LEXICAL OBFUSCATION OF .NET-LANGUAGES AT THE IL-CODE LEVEL

Аннотация. Рассматривается лексическая обфускация кода программ, написанных на языках .NET на IL-уровне. Перечислены существующие направления и методы обфускации. Определены основные проблемы обфускации IL-кода. Обоснована эффективность обфускации IL-кода.

Abstract. Lexical obfuscation of the code of programs written in languages is considered.NET at the IL-level. The existing directions and methods of obfuscation are listed. The main problems of IL-code obfuscation are identified. The effectiveness of IL-code obfuscation is substantiated.

Ключевые слова: защита программного обеспечения, обфускация, лексическая обфускация, IL-код, .NET, C#, информационная безопасность.

Key words: software protection, obfuscation, lexical obfuscation, IL-code, .NET, C#, information security.

DOI: 10.22250/20730268_2023_101_158

Введение

Одной из основополагающих черт информатизации и цифровизации является широкое распространение информационных технологий, их проникновение во все сферы человеческой жизни и приобретение ключевой роли компьютерных устройств и программного обеспечения. К сожалению, программное (ПО) и аппаратное (АО) обеспечение часто становится целью для действий злоумышленников. Одним из направлений защиты ПО от реверс-инжиниринга, нелицензионного использования, а также модификаций с целью внедрения вредоносного кода или изменения поведения программ является обфускация.

Обфускацией называется приведение исходного текста или исполняемого кода программы, с сохранением ее функциональности, к виду, который затруднительно анализировать, понимать и производить его последующую модификацию [1]. Таким образом, обфускация предполагает затруднение действий злоумышленника путем повышения сложности анализа программного продукта какими-либо методами, минимально влияющими на показатели качества ПО (быстродействие, точность, отказоустойчивость и др.).

К основным видам обфускации относятся [1-3]:

1) лексическая обфускация, благодаря которой происходит изменение внешнего вида программного кода для усложнения его восприятия и анализа;

2) обфускация хранения, которая изменяет типы и виды хранения, их замену на пользовательские типы (поток данных в программе становится неочевидным);

3) обфускация потока управления, что изменяет и запутывает логические структуры, добавляет запутывающие блоки, не влияющие на результат выполнения программы.

Ряд языков высокого уровня (Visual Basic, F#, C# и др.), использующиех платформу .NET, предполагают трехступенчатое формирование исполняемого кода. В частности, готовый исходный код компьютерной программы сначала преобразуется в IL-код (Intermediate Language – промежуточный язык). После чего выполняется его оптимизация. При этом убираются в основном операции, не приводящие к какому-либо результату и не оказывающие влияния на работу программы: например, переменные, которые нигде не используются, и операции по их изменению. И затем уже оптимизированный IL-код преобразуется в финальный исполняемый код.

Такая организация процесса компиляции резко снижает эффективность обфускации исходного кода: большая часть внесенных «запутывающих» изменений будет нивелирована алгоритмами оптимизации компилятора. Таким образом, для языков платформы .NET обфускацию в общем случае целесообразно применять не на уровне исходного кода, а на уровне исполняемого и IL-кода.

При этом обфускация на уровне IL-кода представляется более простым и эффективным техническим решением, а лексическая обфускация является самым простым в реализации способом защиты программного кода, но при этом не менее эффективным, чем другие виды обфускации.

Лексическая обфускация

Данный вид обфускации включает следующие методы [2-5]:

1) удаление или изменение комментариев в коде программы;

2) удаление пробельных символов, горизонтальных и вертикальных отступов, используемых для повышения читаемости программного кода;

3) замена имен идентификаторов на произвольные наборы символов;

4) добавление «мусорных» (избыточных) операций;

5) изменение порядка следования операторов и целых блоков кода.

Рассмотрим метод замены имен идентификаторов на примере следующей программы на языке C#:

```
public static void Main()
{
   Console.WriteLine("Введите оклад");
   int salary = Convert.ToInt32(Console.ReadLine());
   Console.WriteLine("Введите ставку");
   double rate = Convert.ToDouble(Console.ReadLine());
   double result = salary * rate - salary * rate * 13 / 100;
   Console.WriteLine($"Вы получите на руки {result:f2}");
}
```

В этом примере приведена простейшая программа, рассчитывающая количество денег, которые сотрудник должен получить после вычета налога. Даже если убрать текстовые сообщения, то по смыслу можно достаточно просто догадаться, что выполняет данный фрагмент кода и для чего он нужен. Для усложнения понимания того, что видит перед собой злоумышленник, можно провести лексическую обфускацию идентификаторов переменных. Результат соответствующего преобразования исходного кода приведен ниже.

```
public static void Main()
{
   Console.WriteLine("Введите оклад");
   int E12GH6Z8 = Convert.ToInt32(Console.ReadLine());
   Console.WriteLine("Введите ставку");
   double H21JK82M = Convert.ToDouble(Console.ReadLine());
   double I4GHN913L16 = E12GH6Z8 * H21JK82M - E12GH6Z8 * H21JK82M * 13 / 100;
   Console.WriteLine($"Вы получите на руки {I4GHN913L16:f2}");
}
```

Как видно на представленном примере, сложно сразу ответить, какую функцию осуществляет та или иная переменная (если не учитывать текстовые сообщения). Это усложняет работу злоумышленника при анализе программы, внедрении вредоносного кода или попытке изъятия данных из необходимых переменных (например, токен подключения к серверу). Аналогичным образом в программном коде вместо конкретных числовых и символьных значений можно использовать константы. Математические операции, а также операции ввода-вывода в свою очередь можно заменить вызовами функций с обфусцированными именами, переменным количеством входных и выходных параметров и запутанным выполнением.

Обфускация IL-кода

Для рассмотрения лексической обфускации на уровне IL-кода используем ту же программу расчета выплаты денежных средств. Ниже приведен ее IL-код. Отметим, что к программе никакие методы обфускации еще не применялись.

```
.method public hidebysig static void
  Main() cil managed
{
  .entrypoint
  .maxstack 3
  .locals init (
    [0] int32 salary,
    [1] float64 rate,
    [2] float64 result
  )
  IL_0000: ldstr       "Введите оклад"
  IL_0005: call        void [mscorlib]System.Console::WriteLine(string)
  IL_000a: call        string [mscorlib]System.Console::ReadLine()
  IL_000f: call        int32 [mscorlib]System.Convert::ToInt32(string)
  IL_0014: stloc.0     // salary
  IL_0015: ldstr       "Введите ставку"
  IL_001a: call        void [mscorlib]System.Console::WriteLine(string)
  IL_001f: call        string [mscorlib]System.Console::ReadLine()
  IL_0024: call        float64 [mscorlib]System.Convert::ToDouble(string)
  IL_0029: stloc.1     // rate
  IL_002a: ldloc.0     // salary
  IL_002b: conv.r8
  IL_002c: ldloc.1     // rate
  IL_002d: mul
  IL_002e: ldloc.0     // salary
  IL_002f: conv.r8
  IL_0030: ldloc.1     // rate
  IL_0031: mul
  IL_0032: ldc.r8      13
  IL_003b: mul
  IL_003c: ldc.r8      100
  IL_0045: div
  IL_0046: sub
  IL_0047: stloc.2     // result
  IL_0048: ldstr       "Вы получите на руки {0:f2}"
  IL_004d: ldloc.2     // result
  IL_004e: box         [mscorlib]System.Double
  IL_0053: call        string [mscorlib]System.String::Format(string, object)
  IL_0058: call        void [mscorlib]System.Console::WriteLine(string)
  IL_005d: ret
} // end of method Program::Main
```

Как видно из листинга, IL-код воспринимается гораздо труднее, чем исходный код на языке высокого уровня. Однако IL-код сохраняет именование всех переменных. После применения простейшей лексической обфускации рассмотренный код примет следующий вид:

```
.method public hidebysig static void
 Main() cil managed
{
 .entrypoint
 .maxstack 3
 .locals init (
  [0] int32 E12GH6Z8,
  [1] float64 H21JK82M,
  [2] float64 I4GHN913L16
 )
 IL_0000: nop

 IL_0001: ldstr      "Введите оклад"
 IL_0006: call       void [mscorlib]System.Console::WriteLine(string)
 IL_000b: nop
 IL_000c: call       string [mscorlib]System.Console::ReadLine()
 IL_0011: call       int32 [mscorlib]System.Convert::ToInt32(string)
 IL_0016: stloc.0    // E12GH6Z8
 IL_0017: ldstr      "Введите ставку"
 IL_001c: call       void [mscorlib]System.Console::WriteLine(string)
 IL_0021: nop
 IL_0022: call       string [mscorlib]System.Console::ReadLine()
 IL_0027: call       float64 [mscorlib]System.Convert::ToDouble(string)
 IL_002c: stloc.1    // H21JK82M
 IL_002d: ldloc.0    // E12GH6Z8
 IL_002e: conv.r8
 IL_002f: ldloc.1    // H21JK82M
 IL_0030: mul
 IL_0031: ldloc.0    // E12GH6Z8
 IL_0032: conv.r8
 IL_0033: ldloc.1    // H21JK82M
 IL_0034: mul
 IL_0035: ldc.r8     13
 IL_003e: mul
 IL_003f: ldc.r8     100
 IL_0048: div
 IL_0049: sub
 IL_004a: stloc.2    // I4GHN913L16
 IL_004b: ldstr      "Вы получите на руки {0:f2}"
```

```
IL_0050: ldloc.2     // I4GHN913L16
IL_0051: box         [mscorlib]System.Double
IL_0056: call        string [mscorlib]System.String::Format(string, object)
IL_005b: call        void [mscorlib]System.Console::WriteLine(string)
IL_0060: nop
IL_0061: ret
} // end of method Program::Main
```

Очевидно, что теперь смысл выполняемых операций выдается только оставшимися конкретными числовыми и строковыми значениями, но не именами переменных.

Проблемы лексической обфускации IL-кода

Помимо вышеупомянутых, существуют и другие методы обфускации, но их реализация может сопровождаться следующими проблемами. Прежде всего, невозможность управления памятью в явном виде не дает реализовать *обфускацию адресов*, поскольку компилятор располагает данные так, как считает оптимальным [2-5].

Обфускация путем преобразования данных реализуема с нюансами. Например, при попытке изменить способ записи числовой константы из десятичного вида в двоичный после компиляции и последующей декомпиляции константная переменная будет представлена вновь в десятичном виде. При попытке хранения строковой информации как набора символов, вызываемых по их коду, после декомпиляции они вновь будет представлена в виде стандартной символьной строки [6].

Обфускация инструкций ассемблера становится задачей обфускации инструкций IL-кода, так как языки .NET транслируются сначала в IL-код, который имеет сходство с языком ассемблер [6,7].

Обфускация отладочной информации в рамках рассматриваемых языков прежде всего должна проводиться разработчиком. Не следует оставлять лишнюю информацию, которая раскрывает работу программы [2].

Надо также отметить, что обфускация любого кода может привести к результату, обратному оптимизации. Полезная работа будет выполняться дольше по причине увеличения количества выполняемых операций или усложнения кода даже для вычислительной техники. Несоизмеримо малые изменения привнесет смена названий переменных, так как вычислительная машина должна просто считать большее или меньшее количество символов на этапе компиляции, а во время исполнения программы эти данные будут уже не критичны.

При лексической обфускации IL-кода можно столкнуться со следующими проблемами:

невозможность сохранения чисел в недесятичном виде в стандартных переменных (int, float, double, decimal). Во время трансляции в IL-код все числа принимают десятичный вид, даже если перед этим они были написаны в двоичном или шестнадцатеричном видах;

невозможность хранения строк в виде набора кодов символов. При трансляции все коды символов становятся непосредственно символами;

невозможность записи текста программы в одну строку. При декомпиляции про-

граммы будет восстановлена «правильная» (условно правильная) структура с учетом отступов и вложенности в области видимости методов, переменных и т.д.

Заключение

Исходя из вышеперечисленного, можно утверждать, что в контексте защиты исполняемых файлов программного обеспечения, написанного на языках семейства .NET, от обратной разработки применение обфускации IL-кода будет более эффективным решением, чем обфускация их исходного кода.

Более того, программный инструмент – обфускатор IL-кода сможет обрабатывать код программ, написанных на любом из языков платформы .NET. Это достигается за счет однотипности генерируемого компиляторами IL-кода и использования обфускатором методов, не опирающихся на специфику конкретного языка программирования.

1. Величко, В.В. Защита .NET-кода методами обфускации // Информатизация образования. – 2006. – № 2 (43). – С. 71-82.

2. Ивченкова, Ю.С. Виды и способы обфускации / Ю.С. Ивченкова, М.К. Савкин // Электронный журнал: наука, техника и образование. – 2016. – № 2(6) – С. 60-66.

3. Драгунцева, И.С. Сравнительный анализ защиты информации методами криптографии, стеганографии и обфускации // Молодой ученый. – 2022. – №14 (409). – С. 9-11.

4. Бортничук, А.В. Обфускация кода приложений.net // Сбор. тез. докл. науч.-практ. конф. студ. КГУ., Курган, 22.03.2021. – Курган: Курганский государственный университет, 2021. – С. 281-282.

5. Тахаутдинов, А.Р. Методы запутывания при защите программного кода / А.Р. Тахаутдинов, Ч.М. Лыонг Ха, А.А. Привалова // Молодежный вестник УГАТУ. – 2021. – № 1(24). – С. 58-60.

6. Лебедева, Д.А. Сравнительный анализ программных средств обфускации // Инновации. Наука. Образование. – 2021. – № 48. – С.1699-1704.

7. Малявский, Р.Д. Обфускация кода программного обеспечения // Молодежная научная школа кафедры «Защищенные системы связи». – 2021. – №1 (3). – С. 20-23.