КОМБИНИРОВАННЫЙ МЕТОД ЗАЩИТЫ ПРОГРАММНОГО КОДА

Кузнецов Л.К.
Финансовый университет при Правительстве Российской Федерации
longin_kuz@mail.ru

Аннотация. *Рассматриваются вопросы защиты программного кода в.NET приложениях. С помощью предлагаемого метода усложняется процесс обфускации программных продуктов.*
Ключевые слова: *защита, код, обфускация, приложение, программа.*

COMBINED METHOD OF SOFTWARE PROTECTION CODE

Kuznetsov L.K.
Financial University under the Government of the Russian Federation

Abstract. *Address the protection of the code in.NET applications. With the help of this method is complicated by the obfuscation process of software products.*
Keywords: *protection, security, code, obfuscation, software, program.*

Технологии программирования прогрессируют очень быстро. Идя в ногу со временем компания Microsoft предложила новую технологию.NET [1]. Технология.NET отражает видение этой компанией технологии создания приложений в эпоху Internet. Платформа.NET решает многие проблемы, которые в прошлом омрачали процесс разработки Windows-приложений. Теперь существует одна для всех поддерживаемых платформой языков программирования парадигма разработки приложений. Платформа.NET позволяет разрабатывать мощные, независимые от языка программирования, настольные приложения и масштабируемые (расширяемые) Web-службы, построенные на базе новой мощной полнофункциональной библиотеки классов. NET Framework.

Защита, или безопасность — это одно из основных требований к приложениям и при разработке ее следует учитывать не в последнюю очередь. Не секрет, что защита.Net-программ представляет собой значительную трудность из-за их большой открытости. Приложения для.NET не представляют сложностей для декомпиляции [2]. Это не является недостатком технологии.NET, а исходит из реальности современных языков с компилируемым промежуточным кодом. В технологии.NET используется явный файловый синтаксис для доставки исполняемого кода, или промежуточного языка Microsoft (MSIL). Будучи языком намного более высокого уровня, чем двоичный машинный код, файлы с промежуточным кодом имеют достаточное количество явно видимых и понятных идентификаторов и алгоритмов. Кроме всего, очевидно трудно создавать что-либо гибкое, легкое для понимания и способное к расширению с одновременным скрытием достаточно важных сведений.

Стоить отметить, что обратный анализ программ, написанных на языках, которые компилируются в промежуточный интерпретируемый код (например, Java, C# и другие CLR-языки), на порядок проще программ, написанных на языках, компилирующихся в машинный код, поскольку в исполняемый код таких программ записывается информация про их семантическую структуру (об их классах, полях, методах и т.д.). В связи с возросшей популярностью таких языков программирования (и, в частности,.NET Framework) задача защиты программ, написанных с их использованием, становится все более актуальной.

Имея в распоряжении декомпилятор.NET, можно легко получить декомпилированный код. В этом

случает код лицензии программного обеспечения, механизмы защиты от копирования и фирменная бизнес-логика будут доступны для обозрения, независимо от того, законно это или нет. Любой человек может использовать полученную информацию в своих целях. Например, он может попытаться найти уязвимости в защите, украсть уникальные идеи, взломать приложение и т.д.

Основными инструментами реверс-инженера являются дизассемблер и отладчик (большинство современных реализаций объединяют эти инструменты в одном продукте). Дизассемблер позволяет по выполняемому коду восстановить исходный код программы в виде инструкций на языке ассемблера, а в некоторых случаях – и в виде программы на языке более высокого уровня (например, C#). Отладчик позволяет загрузить программу «внутрь себя» и контролировать ход ее выполнения (выполнять инструкции программы «по шагам», предоставлять доступ к ее адресному пространству, отслеживать обращения к разным участкам памяти).

Современные дизассемблеры снабжены широким диапазоном средств для полноценного изучения кода человеком в полуавтоматическом режиме. Наиболее ярким представителем инструмента такого рода является дизассемблер IDA [5].

В силу открытости программного кода.Net-приложений применение апробированных методов, используемых для защиты обычных программ (их еще называют Native-программами, программами на родном языке процессоров) для защиты.Net-программ невозможно в принципе.

Таким образом, все предыдущие технологии защиты исполняемых файлов, наработанные в предыдущие годы, оказались неприемлемыми для.Net и работы по защите.Net-программ приходиться заново [3].

Создатели программного обеспечения на платформе.Net в настоящий момент четко осознают необходимость разработки простых и эффективных средств защиты.Net-программ. Сделать защиту оказалось так сложно и трудоемко, что до сих пор нет ни одной защиты.Net-программ чтобы они не взламывались специалистами по взлому.

Однако положение оказалось не безнадежным. Существует решение, которое может воспрепятствовать декомпиляции. Метод защиты программного обеспечения декомпиляции получил название обфускации (другие термины – запутывание, затемнение). Запутывание представляет собой технику безболезненного переименования символов в сборках, а также прочие методы для обмана декомпиляторов. Должным образом выполненное запутывание позволяет во много раз повысить защищенность приложения от декомпиляции, при этом абсолютно не влияя на его работоспособность.

Целью запутывания является скрытие зависимостей и создание неопределенности. При росте неопределенности способность человека понимать многогранные интеллектуальные концепции снижается. Примите к сведению, что здесь не говорится об изменении прямой (выполняемой) логики, а только о представлении ее в форме, затрудненной для понимания. Когда для запутывания программных инструкций используется высококачественное средство, возможный побочный эффект состоит не только в том, чтобы вызвать непонимание у человека, но и сбить с толку декомпилятор. В то же время при сохранении прямой (выполняемой) логики обратная семантика лишается всякого смысла. В результате этого любые попытки человека, выполняющего декомпиляцию инструкций в программный диалект наподобие C# или VB, скорее всего, окончатся неудачей, поскольку трансляция кода будет двусмысленной. Глубокое запутывание создает большое количество вариантов для декомпиляции, некоторые из которых могут привести к потере логики. Декомпилятор, являясь вычислительной машиной, не имеет никакого понятия о том, какие варианты могут быть декомпилированы с правильной семантикой. Люди создают и используют декомпиляторы для автоматизации декомпиляции алгоритмов, которые представляют сложность для понимания. Можно сказать, что любое средство запутывания, которое используется для защиты от декомпиляторов, является еще более сильным сдерживающим фактором для менее производительных попыток человека выполнить подобную задачу.

Затемнение (другой термин – запутывание) программного кода является естественным следствием желания обеспечить сохранность чувствительного кода путем максимального сокрытия логики его работы. Единственным способом, который мог бы затруднить обратный анализ, является запутывание семантики, сокрытие логики работы программы с т ем, чтобы помешать атакующему понять особенности ее работы.

Выделяется несколько видов затемняющих преобразований исходя из способа их действия [4]. На рис. 1 приведены четыре основных класса обфускации:

1. Лексическая.
2. Структур данных.
3. Потока управления.
4. Превентивная.

Лексическая обфускация обеспечивает замену имен идентификаторов (имен переменных, массивов, структур, хешей, функций, процедур и т.д.) на произвольные длинные наборы символов, которые трудно воспринимать человеку, удаляет все комментарии в коде программы или изменяет их на дезинформирующие; удаляет различные пробелы, отступы, которые обычно используют для лучшего визуального восприятия кода программы. Основная задача лексической обфускации затруднить анализ логики программы. При замене принято использовать непечатный набор символов, что мешает так называемому статическому анализу. Ориентироваться в преобразованном коде все равно, что в городе, где вместо названий улиц и номеров домов - случайные цифры. Данная операция необратима, и код получается очень трудночитаемый. Лексическая обфускация обеспечивает наиболее простой способ запутывания злоумышленника, анализирующего код.

Обфускация структур данных заключается в изменение структуры данных, с которыми работает программа. Различают:

- обфускацию хранения, обеспечивающую трансформацию хранилищ данных и самих типов данных (создание и использование необычных типов данных, изменение представления существующих и т.д.);
- обфускацию переупорядочивания, изменяющую последовательности объявления переменных, внутреннее расположения хранилищ данных, а также переупорядочивающую методы, массивы (использующую нетривиальное представление многомерных массивов), определение полей в структурах и т.д.
- обфускацию соединения – усложняет представление используемых программой структур данных. Например, соединение независимых данных или разделение зависимых.

Запутывание потока управления. Данный процесс состоит из конструкций ветвления, условий и итераций, в результате которых строится прямая (выполняемая) логика, при декомпиляции которой получаются неопределенные семантические результаты. В результате запутывания потока управления получается разветвленная логика, затрудненная для анализа. Обфускация потока управления изменяет граф потока управления одной функции или программы в целом. Она может приводить к созданию в программе новых функций. Это самый обширный класс запутывающих преобразований. Приведем методы обфускации потока управления:

- создание детерминированного диспетчера, получающего управление после выполнения ряда операторов и продолжающего выполнение алгоритма по заданному условию, переходя на другие блоки выполнения. Диспетчер полностью контролирует ход алгоритма и преобразует граф потока управления;
- открытая вставка функций. Тело функции подставляется в точку вызова функции;
- вынос группы операторов. Данное преобразование является обратным к предыдущему и обычно дополняет его;
- переплетение функций. Запутывание обеспечивается за счет объединения двух или более функций в одну функцию.

Преобразование циклов. Тело цикла многократно размножается. Условие выхода из цикла и оператор приращения счетчика соответствующим образом модифицируются.

Рис. 1. Классификация запутывающих преобразований

Превентивная обфускация обеспечивает защиту от применения деобфускаторов, декомпиляторов и остальных программных средств деобфускации.

Шифрование строк. С целью обнаружения основной логики приложения взломщики часто ищут в нем определенные строки. Например, если выполняется попытка обойти процедуру регистрации и проверки подлинности, то выполняется поиск строки, которая отображается в приложении при запросе серийного номера. После того как строка найдена, взломщик анализирует инструкции, которые находятся рядом, и изменяет логику программы. Шифрование строк затрудняет данную процедуру, поскольку нужная строка не будет обнаружена. Оригинальная строка будет отсутствовать в коде. Вместо нее будет присутствовать ее зашифрованная форма.

В предлагаемом методе используются сочетания известных и новых алгоритмов, что позволяет выполнять защиту приложений.NET максимально эффективным способом.

Новизна работы заключается в создании собственного метода, с помощью которого можно разнообразить и усложнить процесс обфускации программных продуктов. Эти методики и алгоритмы должны обеспечивать стойкость по отношению к алгоритмам деобфускации.

Разработанная совокупность методов и алгоритмов позволяет решать задачи защиты программного обеспечения от несанкционированного анализа. Разработанные алгоритмы являются автоматическими, что позволяет применять их на практике пользователям не обладающими специальными навыками и знаниями в области защиты программного обеспечения.

Была разработана программа-обфускатор со следующими возможностями:

1. Переименование переменных.
2. Шифрование текстовых строк.
3. Защита от декомпилирования.
4. Переименование методов.
5. Переименование параметров.
6. Изменение потока программы.

Комбинированный метод защиты кода представляет собой инструмент, который, не вмешиваясь и не внося изменения в исходный код, позволяет защитить интеллектуальную собственность, содержащуюся в приложениях, разработанных с использованием технологий.NET.

Список литературы

1. Рихтер Дж. CLR via C#. Программирование на платформе Microsof.NETFramework 4.0 на языке C#. 3-е изд. – СПб.: Питер, 2012. - 928 с.: ил.
2. Джонсон Гленн, Нортроп Тони. Разработка клиентских веб-приложений на платформе Microsoft.Net Framework. Учебный курс Microsoft / Пер. с англ. — М.: Русская Редакция, СПб.: Питер, 2007. - 768 с.: ил.
3. Проскурин В. Г. Защита программ и данных. – М.: Издательский центр «Академия», 2012. - 208 с.
4. Защита Net, кто виноват и что делать. – http://www.aktiv-company.ru/news/company-editorial-06-09-2011.html.
5. Обфускация и защита программных продуктов. –http://www.citforum.ru/security/articles/obfus/.