ЗАЩИТА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

Д.С. Акулич

Белорусский государственный университет информатики и радиоэлектроники,
кафедра программного обеспечения информационных технологий
Минск, Республика Беларусь
e-mail: dmitry.akulich@gmail.com

Проведён анализ методов защиты интеллектуальной собственности разработчиков программного обеспечения. Рассмотрены особенности программного обеспечения как объекта защиты авторских прав. Для защиты программного обеспечения от несанкционированного использования разработан метод внедрения водяных знаков на основе модификации потока управления программы. Разработано программное средство для внедрения и извлечения водяных знаков.

Ключевые слова – защита, интеллектуальная собственность, несанкционированное использование, программное обеспечение.

ВВЕДЕНИЕ

Программное обеспечение является интеллектуальной собственностью. В результате нарушений прав интеллектуальной собственности разработчики несут убытки. Потери разработчиков, связанные с нарушением прав интеллектуальной собственности, с каждым годом растут. Существует потребность в исследовании угроз и разработке методов защиты программного обеспечения.

МЕТОДЫ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Для программного обеспечения актуальными являются следующие угрозы: несанкционированное использование, обратное проектирование и несанкционированная модификация. Для ликвидации данных угроз существуют технические направления: внедрение водяных знаков, обфускация, шифрование программного кода и защита от модификации [2].

Обфускация ("obfuscation" - запутывание) – один из методов защиты программного кода, который позволяет усложнить процесс реверсивной инженерии кода защищаемого программного продукта. Суть процесса обфускации заключается в том, чтобы запутать программный код, то есть трансформировать его так, чтобы он был очень труден для изучения и модификации посторонними лицами (будь то взломщики, или программисты, которые собираются узнать уникальный алгоритм работы защищаемой программы).

Шифрование программного кода используется для того чтобы предотвратить вмешательство в программу, а также усложнить изучение взломщиком того, как устроена программа, как она работает, как в ней реализован метод защиты и т.д. Данный метод защиты предусматривает шифрование кода программы, после чего она в зашифрованном виде поставляется конечным пользователям. Когда пользователь запускает такую программу, вначале будет запущена процедура расшифровки программы, которой потребуется ключ, с помощью которого будет расшифрована запускаемая программа.

При защите от модификации в программу помещается процедура проверки целостности самой программы (tamper-proofing), что позволяет определить, была ли программа изменена (были ли внесены какие-то либо изменения в ее код). Если эта процедура обнаруживает, что в программу внесены изменения, она делает программу нефункциональной. Это позволяет защитить программный продукт от изменений со стороны злоумышленника.

Использование водяных знаков (software watermark) основывается на записи в код программы скрытой информации, которая позволяет автору программы доказать то, что эта программа является его интеллектуальной собственностью.

К водяным знакам предъявляются следующие требования:

– водяной знак должен быть надежно расположен в программе и впоследствии может быть извлечен без каких-либо изменений (повреждений);

– водяной знак не должен влиять на работу программы;

– водяной знак должен нести определенную информацию, которая позволит доказать, то что ее присутствие в программе неслучайно, то есть является результатом преднамеренных действий.

Отпечаток пальца (software fingerprint) – технология, которая кроме записи информации, позволяющей доказать право интеллектуальной собственности на программу, требует запись в каждую копию программы уникального идентификационного кода, присваиваемого каждому покупателю программы, что позволяет впоследствии быстро отследить нарушителя авторского права, который, например, будет нелегально перепродавать программу.

ПОТОК УПРАВЛЕНИЯ ПРОГРАММЫ

Программный модуль может быть представлен как набор функций. Каждая функция состоит из последовательности инструкций. В некоторых случаях управление передаётся последовательно от исходной инструкции к следующей, в других случаях задаются условные и безусловные переходы, которые определяют, куда перейдёт управление.

Основным способом представления потока управления программы является граф потока управления [7]. Вершинам графа потока управления соответствуют последовательности инструкций, в которых управление передаётся от предыдущей к следующей. Такие последовательности инструкций называются базовыми блоками. Дугам графа соответствуют передачи управления, которые происходят на основе условных или безусловных переходов. Дуги, соединяющие базовые блоки потока управления, называются границами потока. Граф потока управления изображён на рисунке 1.

Интерес представляет случай, когда в графе потока управления есть вершины с одинаковыми списками предшествующих или последующих вершин (например, на рисунке 1 такими вершинами являются 3 и 4). Такие вершины (и соответствующие базовые блоки) будем называть равными по границам.

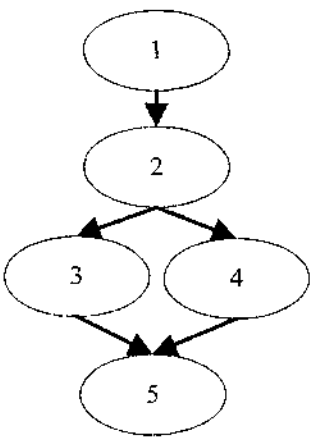

Рис. 1. Граф потока управления

ВНЕДРЕНИЕ ВОДЯНОГО ЗНАКА НА ОСНОВЕ МОДИФИКАЦИИ ПОТОКА УПРАВЛЕНИЯ

Внедрение водяных знаков в программы может осуществляться с помощью перестановки базовых блоков потока управления. Этот подход используется в методе Дэвидсона [5]. В соответствии с методом Дэвидсона, в защищаемой программе выбираются несколько произвольных базовых блоков потока управления. Эти блоки переставляются, чтобы скрыть дополнительную информацию; для сохранения функционирования программы вставляются дополнительные инструкции jump, с помощью которых восстанавливается начальная логика переходов между базовыми блоками. Метод Дэвидсона позволяет внедрять в программы водяные знаки и отпечатки пальцев большого объёма, но у этого метода есть следующие недостатки:

- перестановка произвольных блоков управления нарушает оптимизацию компилятора, что может снизить скорость выполнения защищаемой программы;
- вставка дополнительных инструкций jump вносит в защищаемую программу избыточность, что может быть использовано злоумышленниками для обнаружения водяных знаков.

Этих недостатков можно избежать, если переставлять только равные по границам базовые блоки. Такие перестановки не нарушают оптимизацию компилятора, так как равные по границам базовые блоки идентичны по входам и выходам. Кроме того, после их перестановки не требуется вставлять дополнительные инструкции jump.

В программе базовые блоки потока управления расположены в определённом порядке. Но равные по границам базовые блоки можно менять местами, что никак не отразится на функционировании модифицируемой программы. Последовательность равных по границам базовых блоков образует перестановку. При наличии N равных по границам базовых блоков существует N! вариантов перестановок базовых блоков. Применяя определённую перестановку, можно скрывать в программном модуле информацию.

Если переставлять только равные по границам базовые блоки, ёмкость программного модуля как контейнера водяных знаков будет существенно уменьшена по сравнению с оригинальным методом Дэвидсона. Для того чтобы определить, достаточно ли часто в потоках управления программных модулей встречаются равные по границам базовые блоки, были проведены измерения. Определялось количество ситуаций, когда в потоке управления присутствуют равные по границам базовые блоки, и количество таких базовых блоков для каждой ситуации. Если в потоке управления некоторой функции есть N базовых блоков, существует N! перестановок данных блоков и, следовательно, в данной функции можно скрыть $[\log_2(N!)]$ бит водяного знака. Таким образом, ёмкость программного модуля как контейнера водяных знаков выражается формулой:

$$C = \sum_{i=1}^{k} [\log_2(N_i!)] \qquad (1)$$

k – количество функций, содержащих равные по границам базовые блоки,

N_i – количество равных по границам базовых блоков функции i.

Измерения проводились для стандартных программных модулей платформы .NET. Эти модули не имеют принципиальных отличий по размеру и использованию инструкций от программных модулей, которые нуждаются в защите от несанкционированного использования. Результаты измерений приводятся в таблице 1.

Таблица 1

Частоты встречаемости равных по границам базовых блоков

Наименование программного модуля	Количество функций, содержащих равные базовые блоки	Ёмкость программного модуля, бит
System.dll	2194	2279
System.Configuration.dll	270	302
System.Deployment.dll	200	206
System.Design.dll	1673	1758
System.Drawing.dll	251	256
System.Management.dll	153	166
System.Messaging.dll	141	143
System.Security	173	187
System.Web.dll	3021	3155
System.XML.dll	1839	2189

Как правило, для внедрения водяного знака или отпечатка пальцев достаточно, чтобы контейнер имел ёмкость 128 бит. Можно сделать вывод, что равные по границам базовые блоки в потоках управления программных модулей встречаются достаточно часто, чтобы можно было использовать описанный метод для внедрения водяных знаков или отпечатков пальцев.

Для того чтобы внедрять и извлекать информацию в программы, должен быть способ определить исходные номера базовых блоков – это необходимо для определения, как нужно переставить блоки, чтобы они закодировали определённую перестановку, и какой перестановке соответствует определённое расположение базовых блоков. Так как базовые блоки представляют собой последовательность инструкций, для сравнения базовых блоков можно проводить последовательное сравнение их инструкций. Таким образом, можно отсортировать базовые блоки по возрастанию и пронумеровать их. На основании номеров базовых блоков они переставляются при внедрении водяных знаков. Также при извлечении водяного знака с помощью номеров базовых блоков определяется, какая перестановка применена в программе.

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ВНЕДРЕНИЯ ВОДЯНЫХ ЗНАКОВ

Было разработано программное средство, реализующее предложенный алгоритм внедрения водяных знаков. Программное средство разработано на основе платформы Microsoft Phoenix [7], предназначается для защиты .NET программ от несанкционированного использования. Окно программы изображено на рисунке 2.

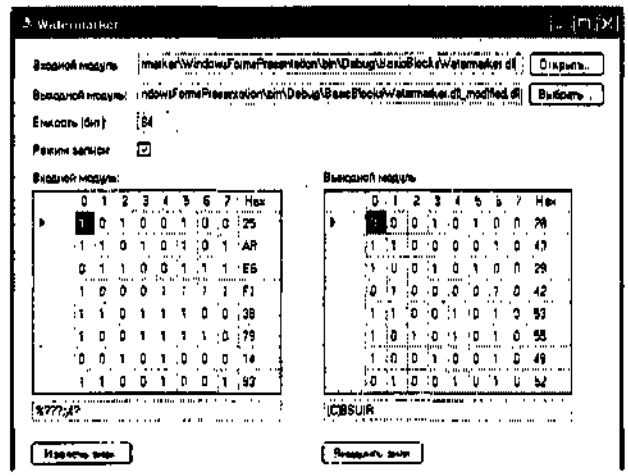

Рис. 2. Программное средство для работы с водяными знаками

Программное средство позволяет внедрять и извлекать водяные знаки как информацию в двоичном или текстовом представлении. Фактически, в качестве водяного знака можно использовать копирайт-строку или любую другую информацию, например зашифрованные данные о создателе программы или о клиенте, для которого она предназначается. Таким образом, можно применять программное средство для внедрения водяных знаков и отпечатков пальцев.

После запуска программного средства открывается окно, в котором доступны возможности открыть входной программный модуль и выбрать выходной модуль. Входным модулем может служить исполняемая программа или библиотека. После открытия файла входного модуля выбирается имя выходного модуля. Также после открытия файла входного модуля становится доступной возможность извлечь водяной знак. После извлечения водяного знака пользователь может видеть его представление в текстовом и двоичном виде. Также отображается ёмкость входного программного модуля – количество двоичных разрядов, которые можно внедрить как водяной знак. После извлечения водяного знака становится доступным выбор режима записи. В этом режиме программное средство позволяет внедрить водяной знак в программный модуль. При внедрении водяного знака будет сгенерирован модуль, функционально идентичный входному, но обладающий скрытым водяным знаком. Внедрённый водяной знак можно извлечь, воспользовавшись программным средством в режиме чтения.

Программы, работающие на платформе .NET, обладают особенностью: можно легко восстановить исходный код программы. Для демонстрации того, как работает алгоритм внедрения водяного знака, в таблице 2 приводится восстановленный исходный код программы до и после внедрения водяного знака.

Таблица 2

Исходный код защищаемой программы

До внедрения водяного знака	После внедрения водяного знака

if (operation)
{
num = a + b;
}
else
{
num = a – b;
}
	if (!operation)
	{
	num = a - b;
	}
	else
	{
	num = a + b;
	}

Видно, что функционирование приведённого участка программы не изменяется. При наличии обеих версий программы (до и после внедрения водяного знака) можно увидеть отличие модифицируемой программы. Но распространяться будет только программа с внедрённым водяным знаком, и при таких условиях анализ исходного кода не позволит обнаружить, что был внедрён водяной знак.

Таким образом, для внедрения водяного знака в распространяемую программу можно использовать модификацию потока управления. Для этого можно переставлять базовые блоки. Если переставлять не произвольные, а только равные по границам базовые блоки, будет снижена ёмкость программы как контейнера водяного знака. Но вместе с тем перестановка таких базовых блоков не вносит в модифицируемую программу ничего избыточного, что положительно сказывается на скрытности водяного знака. Также перестановка определённых базовых блоков потока управления не сказывается на характеристиках программы.

ЛИТЕРАТУРА

[1] Ахо, А. Компиляторы: принципы, технологии и инструменты / А. Ахо. — М. : Издательский дом "Вильямс", 2003. – 768 с.

[2] Обфускация и защита программных продуктов // CIT-Forum [Электронный ресурс]. - 2004. - Режим доступа : http://www.citforum.ru/security/articles/obfus/. Дата доступа : 20.09.2009.

[3] Collberg, C. Watermarking, Tamper-Proofing and Obfuscation -- Tools for Software Protection / C. Collberg, C. Thomborson // IEEE Transactons on Software Engineering. - 2002. – Vol. 28, № 8. – P. 735–746.

[4] Collberg, C. Software Watermarking / C. Collberg // University of Arizona [Electronic resource]. – 2008. – Mode of access : http://sandmark.cs.arizona.edu/pubs/wm04-handout.pdf . – Date of access : 20.09.2009.

[5] Hattanda, K. The Evaluation of Davidson's Digital Signature Scheme / K. Hattanda, S. Ichikawa. – Ieice, 2004. – 2 p.

[6] Holleyman, R. Piracy Study / R. Holleyman // Business Software Alliance [Electronic resource]. - 2008. – Mode of access : http://w3.bsa.org/globalstudy/. – Date of access : 20.09.2009.

[7] Phoenix Technical Overview // Phoenix [Electronic resource]. - 2007. – Mode of access: https://-connect.microsoft.com/Phoenix/content/content.-aspx?ContentID=4513. Date of access: 20.09.2009.