УДК 681.3.06, 681.324.06

СТАТИСТИЧЕСКИЙ ПОДХОД ДЛЯ ВНЕДРЕНИЯ ВОДЯНЫХ ЗНАКОВ В ИСПОЛНЯЕМЫЙ КОД

С.С. ПОРТЯНКО, В.Н. ЯРМОЛИК

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 18 февраля 2005

В данной статье затронута проблема обеспечения защиты интеллектуальных прав разработчиков программного обеспечения. В частности, рассмотрен вопрос о противодействии несанкционированному повторному использованию программных компонент. В качестве одного из решений данной проблемы рассмотрены технологии водяных знаков и отпечатков пальцев. Произведен сравнительный анализ подходов к внедрению статических водяных знаков и выделены преимущества статистического подхода. На основе данного подхода предложен метод размещения в исполняемом коде программы признака авторства, использующий статистические свойства случайных последовательностей. Приведены результаты моделирования предложенного метода, а также возможное направление дальнейших исследований.

Ключевые слова: интеллектуальные права, программное обеспечение, водяной знак.

Введение

В настоящее время выпускается большое количество компонентов программного обеспечения, выполненных в виде библиотечных модулей и рассчитанных на повторное использование. Реализованные в таких модулях процедуры могут быть вызваны из другой программы посредством включения кода библиотеки в тело основной программы. Для компонент, реализованных на языке C++, при условии отсутствия их исходных кодов угроза обратного проектирования с целью понимания работы запрограммированных алгоритмов является несущественной, поскольку однозначное преобразование ассемблерного кода в исходный на языке C++ представляет собой крайне трудоемкую задачу. Причиной этому, как правило, является отсутствие информации о версии использованного компилятора, параметрах оптимизации кода и библиотеках сторонних производителей, включенных в тело модуля, без которой восстановление исходного кода на C++ кода является практически невозможным. Однако достаточно ощутимой является иная проблема, которая связана с несанкционированном использованием уже скомпилированной библиотеки, исходный код на языке высокого уровня которой не доступен, но интерфейс известен. Так, легально приобретенная разработчиком программного обеспечения библиотека может по ряду причин оказаться общедоступной, в результате чего не исключено ее несанкционированное использование третьими лицами. Такие действия могут повлечь за собой потерю прибыли как для легального пользователя библиотеки, так и для ее разработчика. Для того чтобы последний мог осуществлять контроль над распространением и использованием своих библиотек другими лицами, необходимо наличие средства поиска фрагментов библиотеки в теле программного модуля и идентификации ее авторов и легальных пользователей. Возможным решением данной проблемы является использование "водяных знаков" (ВЗ) и "отпечатков пальцев" — скрытых информационных сообщений,

размещаемых в программном обеспечении и служащих для защиты интеллектуальных прав его разработчиков.

Краткий обзор существующих подходов к внедрению ВЗ

Водяные знаки, внедряемые в программное обеспечение при помощи существующих на данный момент методов, можно разделить на так называемые динамические и статические [1]. Особенностью динамических водяных знаков является необходимость запуска программы на выполнение для того, чтобы определить их наличие. Статические водяные знаки, в отличие от динамических, для своего извлечения не требуют выполнения программы и могут быть обнаружены посредством анализа ее содержимого (кода или данных). Кроме того, по объекту внедрения водяные знаки разделяют на водяные знаки, размещаемые в структурах данных программы ("data watermarking"), и на ВЗ, внедряемые в исполняемый код ("code watermarking").

По способу интерпретации и информационной емкости водяные знаки могут быть разделены на водяные знаки, дающие ответ на вопрос "является ли данный разработчик автором программы?", чаще всего представляющие собой некоторый специфический признак, и на ВЗ, отвечающие на вопросы "кто именно является автором данной программы?" и/или "кто ее легальный пользователь?", часто называемые "отпечатками пальцев". Последние в самом простом случае могут представлять собой закодированную в коде программы последовательность символов, например, зашифрованную "Copyright" строку. Что касается признака авторства, им может служить некая специфическая характеристика, внедренная в программу, например, нестандартные частоты встречаемости определенных наборов инструкций [2]. Такой тип ВЗ обладает сравнительно низкой информационной емкостью, однако является более устойчивыми к искажениям.

Преимущества методов внедрения ВЗ, основанных на статистических характеристиках

Методы, использующие статистические свойства программы, обеспечивают присутствие в ней некоторой уникальной статистической характеристики, которое интерпретируется как наличие водяного знака. Самым простым примером алгоритма внедрения, основанного на статистических свойствах исполняемого кода, является изменение частот использования определенных инструкций. Даже такой примитивный метод является достаточно устойчивым к случайным искажениям отдельных команд программы [3]. В случае же использования методов внедрения ВЗ, основанных на его кодировании в участках кода, необходимо обеспечение жесткой привязки команд, хранящих внедренную информацию, к логике работы программы для того, что бы исключить возможность ее преднамеренного искажения, которое оставило бы без изменения логику работы программы. Обеспечение такого требования является весьма сложной задачей, поскольку практически к любой программе могут быть применены эквивалентные преобразования кода, не влияющие на ее логику, однако способные исказить до неузнаваемости внедренную информацию об авторе.

Главным преимуществом методов размещения ВЗ, основанных на статистических характеристиках, является малое количество внедряемой информации при достаточно большом объеме преобразований объекта, что обеспечивает повышенную по сравнению с другими подходами устойчивость против атак, нацеленных на удаление или искажение. Кроме того, при данном подходе могут быть использованы более сложные схемы, основанные на свойствах случайной последовательности, обеспечивающие построение средств внедрения водяных знаков, оказывающих минимальное влияние на качество программы и позволяющих размещать в теле программы ВЗ, устойчивые к применению частотного анализа и обладающие повышенной защищенностью от случайного обнаружения. Один из таких подходов будет рассмотрен в данной статье.

Требования, предъявляемые к исполняемому коду как к объекту внедрения ВЗ

Для того чтобы разместить дополнительную информацию в некотором объекте, в нем необходимо выделить множество элементов, значения которых являются близкими к случайным и допускают модификацию, не влияющую на функционирование самого объекта. Наличие возможности модификации элементов объекта определяется степенью избыточности его представления. Касательно программного обеспечения – на самом высоком уровне избыточность

Касательно программного обеспечения – на самом высоком уровне избыточность проявляется в наличии возможности написания программы множеством различных способов. Точно так же может существовать множество вариантов реализации одного и того же алгоритма, и, наконец, на самом низком уровне некоторые примитивные операции, такие как, например, обмен содержимым регистров или увеличение значения переменной на единицу, может быть осуществлены более чем одним набором команд процессора. Последнее утверждение является справедливым, в частности, для языка ассемблер процессоров серии Intel x86.

В результате анализа исполняемого кода программных модулей было выяснено, что последний удовлетворяет требованиям к объекту внедрения скрытой информации, и в качестве элементов, образующих модифицируемое множество, хранящее водяной знак, могут выступать отдельные машинные инструкции, либо образованные ими группы. Для модификации данных элементов этого могут быть использованы механизмы замены инструкций на эквивалентные, а также механизмы перестановок независимых команд.

Внедрение водяного знака в цепочку равновероятных взаимозаменяемых инструкций

Как уже отмечалось выше, элементы, образующие множество элементов, способное хранить скрытую информацию, представляют собой примитивные операции. Для модификации данных элементов может быть использован один из двух подходов. Первый из них основывается на эквивалентных преобразованиях (подстановках) — заменах одного набора команд другим, выполняющим в точности такую же операцию [4]. При таком подходе, если рассматриваемая операция может быть выполнена при помощи, например, двух различных наборов команд, то соответствующий ей элемент может принимать два значения, например, ноль и единицу. Второй подход основан на перестановках независимых команд, образующих модифицируемый элемент.

Критерии отбора инструкций, кодирующих ВЗ

При первом подходе, использующем эквивалентные замены команд, не все примитивные операции могут выступать в роли модифицируемых элементов. Это объясняется тем, что только часть операций может быть выполнена двумя и более способами — при помощи разных наборов машинных инструкций, и только это множество операций может рассматриваться как элементы, множество которых способно хранить признак авторства. Еще одним требованием, предъявляемым к таким операциям, является близкая частота использования наборов инструкций соответствующих данной операции. Соблюдение этого требования необходимо для того, что бы в процессе внедрения водяного знака в результате модификации элементов не нарушались частотные свойства исполняемого кода программы. В противном случае, при ее частотном анализе может обнаружиться, что некоторая инструкция встречается гораздо чаще, чем следует, что выдаст присутствие водяного знака.

Поскольку число примитивных операций, которые могут быть выполнены более чем двумя способами, крайне мало, в качестве элементов предлагается использовать операции, имеющие два значения – допускающие их выполнение двумя различными наборами команд. Таким образом, каждый элемент может иметь значение ноль или единица, т.е. хранить один бит информации.

Тест на монотонность как средство обнаружения ВЗ

Все множество вхождений таких элементов в тело программы представляет собой последовательность бит. Предлагается внедрять скрытую информацию в статистические свойства данной последовательности. В качестве статистической характеристики предлагается использовать результат теста на монотонность ("Runs Up" test) [5]. Данный тест был разработан для проверки качества псевдослучайных последовательностей. Результат теста представляет собой распределение длин неубывающих участков. На рис. 1,*а* приведена гистограмма распределения, которое было получено при проведении данного теста на последовательности бит, образованной значениями элемента, которому соответствует операция обнуления регистра. Последовательность значений элементов может быть модифицирована таким образом, чтобы в ней полностью отсутствовали

неубывающие участки определенной длины, в результате чего будет получено распределение изображенное на рис. 1,*б*.

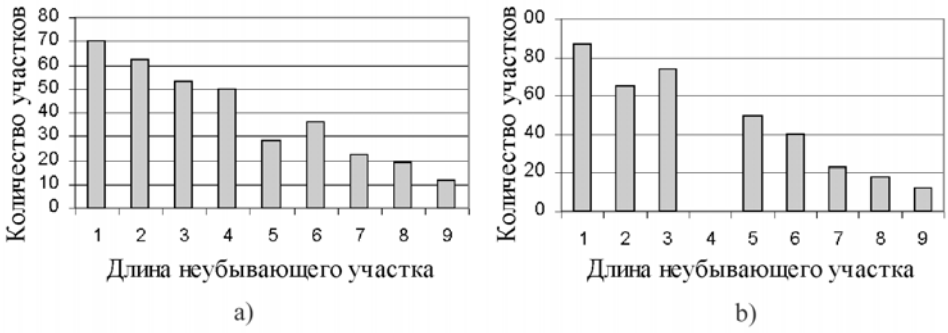

Рис. 1. Распределение, полученное тестом на монотонность для немодифицированной последовательности (*а*) и распределение, полученное для последовательности, в которой отсутствуют все неубывающие участки длиной четыре (*б*)

Последовательность бит, которой соответствует распределение, изображенное на рис. 1,*б*, несет дополнительную информацию, заключающуюся в отсутствии невозрастающих участков длиной четыре. Такая характеристика последовательности может быть достигнута изменением значений некоторых ее элементов с нуля на единицу и наоборот. Предлагается использовать метод, в соответствии с которым последовательность модифицируется таким образом, чтобы соотношение нулей и единиц после внесения признака осталось таким же, как и до модификации. Это будет означать, что соотношение частот использования наборов команд, представляющих значения элемента, в исходной программе и в ней же, но содержащей водяной знак, останется неизменным.

Параметризация алгоритма внедрения ВЗ

В описанном выше подходе к внедрению скрытой информации при проведении теста на монотонность значения элемента анализировались в такой очередности, в которой элемент встречается в теле программы, т.е. от младших адресов к старшим. Внедренная таким образом информация может быть обнаружена в случае, если известна операция, соответствующая кодирующему водяной знак элементу, и алгоритм получения статистической характеристики (в рассматриваемом случае — тест на монотонность). Это означает, что водяной знак, внедренный таким способом, не является скрытым. Для того чтобы избежать обнаружения водяного знака, алгоритм его внедрения должен быть параметризован. Предлагается сделать порядок анализа значений элементов зависимым от секретного ключа. При использовании такой параметризации во время внедрения скрытой информации существенные отличия в распределении, выдаваемом тестом на монотонность, от первоначального будут обнаружены только при условии наличия ключа, задающего очередность анализа значений элементов. При анализе значений элементов от младших адресов к старшим в получаемом распределении признак, характеризующий наличие скрытой информации (отсутствие невозрастающих участков последовательности заданной длины), будет отсутствовать. На рис. 2 показаны результаты моделирования такого подхода для алгоритма внедрения водяного знака, основанного на удалении всех неубывающих подпоследовательностей длины четыре.

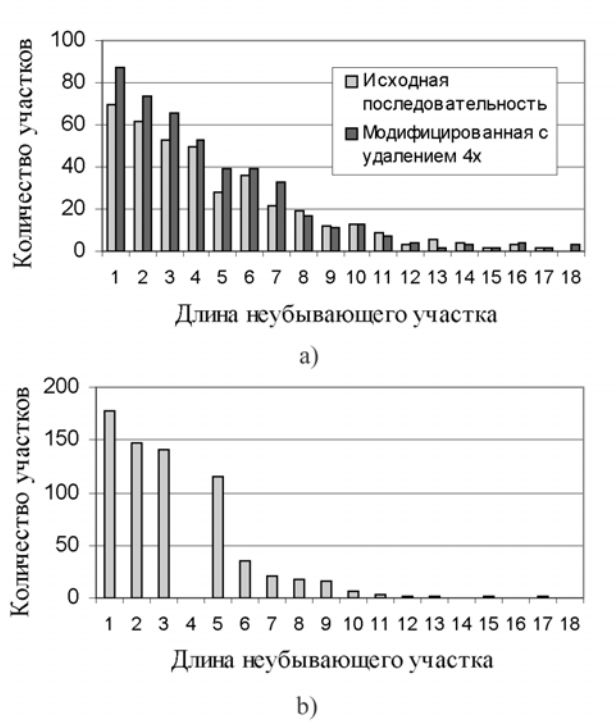

Рис. 2. Результат теста на монотонность исходной и модифицированной последовательности при анализе от младших к старшим адресам (*a*) и результат теста на монотонность модифицированной последовательности при последовательности анализа, задаваемой ключом *K* (*b*)

На рис. 2,*a* показано распределение, полученное для исходной последовательности при порядке анализа значений элементов "от младших адресов к старшим" и распределение, полученное в результате анализа значений элементов последовательности, содержащей скрытый признак, "от младших адресов к старшим". На рис. 2,*б* показано распределение, полученное при анализе значений элементов последовательности, содержащей скрытый признак, в порядке, определяемом ключом *K*.

Результаты моделирования процесса внедрения водяного знака

Было проведено моделирования внедрения водяного знака в соответствии с описанным выше методом. Алгоритм внедрения был параметризован 16-битным ключом. На рис. 3 приведены результаты оценки количества подстановок наборов инструкций, требуемых для внедрения водяного знака в тело программы, содержащее 3000 модифицируемых элементов. Алгоритм внедрения параметризован при помощи ключа *K*=12345.

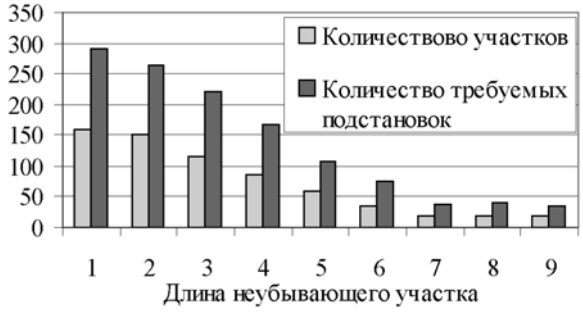

Рис. 3. Результаты оценки числа подстановок, требуемых для "удаления" неубывающих участков различной длины

Как показывают результаты моделирования процесса внедрения водяного знака, объем преобразований модифицируемой последовательности элементов, представляющих собой эквивалентные наборы инструкций, не превышает 10%. Объем преобразований относительно всей программы не превышает 0,2% от всех инструкций.

Работа выполнялась в рамках гранта BMBF BLR 02/006 "DSP-based control systems for safety-related applications"

Заключение

Предлагаемый метод основывается на статистических свойствах последовательности, образованной наборами команд ассемблера. В качестве конкретного алгоритма вычисления интересующей статистической характеристики предлагается использовать тест на монотонность, однако возможно применение и других статистических тестов, таких, например, как тест серий. Разработанный подход к внедрению скрытого признака, который является водяным знаком, параметризуется за счет использования секретного ключа, чем достигается скрытость внедряемой информации. Разрядность ключа, используемого для параметризации алгоритма, может быть увеличена путем увеличения степени порождающего полинома, используемого для задания очередности анализа значений модифицируемых элементов.

STATISTICAL APPROACH TO EXECUTABLE CODE WATERMARKING

S.S. PARTSIANKA, V.N. YARMOLIK

Abstract

In this paper the problem of software developer’s intellectual property protection is touched. In particular, the question of program components unauthorized reuse is considered. Software watermarking and fingerprinting technologies are examined as one of problem solutions. Comparative analysis of existing static watermarking approaches is provided and statistical approach advantages are considered. The method of authorship sign embedding in program’s executable, based on mentioned approach, which employs statistical properties of random sequences is proposed. The results of method simulation and possible direction of further studies are represented.

Литература

1. *Collberg C., Thomborson C.* // In Principles of Programming Languages. 1999. P. 311–324.
2. *Stern J., Hachez G., Koeune F., Quisquater J* // In Information Hiding Workshop '99. 1999. P. 368–378.
3. *Ярмолик В.Н., Портянко С.С.* // Вести Института современных знаний. 2004. N 1(18). С. 62–69.
4. *Портянко С.С.* // Докл. БГУИР. 2004. N 5. С. 29–30.
5. *Knuth D.* The Art of Computer Programming. Vol. II, Seminumerical Algorithms. Addison-Wesley, 1981. P. 65.