УДК 621.396

И.В. Чумаченко[1], Е.Е. Малафеев[2], Е.Л. Шевцов[1]

[1]*Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков*
[2]*Научно-исследовательский институт радиоизмерений, Харьков*

# ОБФУСКАЦИЯ АЛГОРИТМИЧЕСКИХ СТРУКТУР

*Рассмотрены методы обфускации (запутывания) программ. Для повышения защищенности программ предлагается обфускацию проводить на ранних этапах создания программного продукта – на этапе разработки алгоритмического обеспечения. В основе метода лежит использование динамических алгоритмических преобразователей.*

***метод обфускации, защищенность программ, алгоритмические структуры***

## Введение

**Постановка проблемы.** Информационные технологии прочно вошли в бизнес, образование, производство, открыв человеку небывалые возможности в достижении высоких скоростей получения и обработки информации, автоматизации производственных, управленческих и иных процессов. Глобальное проникновение информационных технологий в нашу жизнь, постепенный переход к электронным способам ведения бизнеса ставят перед участниками рынка новые задачи по обеспечению информационной безопасности. Всеобщая информатизация сопровождается ростом числа компьютерных преступлений и, как следствие, материальных потерь. Поэтому информационная безопасность стала обязательным условием ведения современного бизнеса.

Решение задач, связанных с предотвращением воздействия непосредственно на информацию, осуществляется в рамках комплексной проблемы обеспечения безопасности информации и имеет недостаточно развитую научно-методическую базу. В последнее время появились новые проблемы обеспечения безопасности, связанные с информационными технологиями, которые, по мнению ряда зарубежных и отечественных экспертов в области их создания и применения, в значительной степени определяют эффективность создаваемых компьютерных систем. На первый план выдвигается безопасность технологий создания программного обеспечения компьютерных систем [1, 2].

**Анализ исследований и публикаций.** Используемые в настоящее время системы защиты информации основаны на использовании программно-аппаратных технологий обеспечивающих аутентификацию пользователя путем сравнения кода пользователя с эталонным кодом. Существующие способы, затрудняющие использование средств трассировки основаны на контроле последовательности исполнения команд в процессоре, обращений к отладочным прерываниям, времени исполнения эталонного участка программы. В настоящее время широко распространены языки программирования, такие как Java, в которых "исполняемой" формой программы является не машинный код для некоторого типа процессоров, а машинно-нейтральное представление. Задача декомпиляции программы из такого представления обратно в программу на языке Java значительно проще, чем декомпиляция из машинного кода. В качестве одного из способов борьбы с этим используется запутывание программ [3, 4].

Запутанной называется программа, которая получается в результате применения к исходной незапутанной программе запутывающих преобразований и на всех допустимых для исходной программы входных данных выдаёт тот же самый результат, что и оригинальная программа. Запутанная программа более трудна для анализа, понимания и модификации, а её обратная инженерия становится экономически невыгодной. Известные подходы к запутыванию относятся, в основном, к программам на языках Си и Java и неэффективны для языков низкого уровня. В настоящее время нет методов защиты программ для этих языков [3].

Запутывание программ – достаточно молодое направление исследований. Анализ запутывающих преобразований опубликован в работе [5] группы, возглавляемой К. Колбергом и К. Томборсоном. Результаты исследований конкретных алгоритмов запутывания графа потока управления и данных программы, а также приложения запутывания программ к смежным областям, таким как обеспечение устойчивости программы к несанкционированной модификации (tamper-resistance) или внесение в программу "водяных знаков" (watermarking) приведены в работах [6]. В работе [7], предложен новый подход к запутыванию графа потока управления программы, который заключается в преобразовании графа в "плоскую" форму. Доказано, что статический анализ запутанной программы с целью восстановления порядка следования базовых блоков является NP-трудной задачей.

Запутывающие преобразования можно разделить на несколько групп в зависимости от того, на

трансформацию какой из компонент программы они нацелены. Преобразования форматирования, которые изменяют только внешний вид программы (удаление комментариев, отступов в тексте программы и др.). Преобразования потока управления изменяют граф потока управления одной функции. Они могут приводить к созданию в программе новых функций. Существующие методы запутывания и инструменты для запутывания программ используют не единственное запутывающее преобразование, а некоторую их комбинацию [8, 9].

Анализ известных подходов к обфускации программ показывает, что методы, используемые в них, как правило, подобраны эмпирически и слабо обоснованы теоретически.

**Целью статьи** является разработка нового подхода к обфускации программ, позволяющего повысить их защищенность.

## Изложение основного материала исследования

Анализ методов обфускации программ показал, что в большинстве из них запутывание выполняется для уже разработанной программы. Структура таких преобразований приведена на рис. 1.

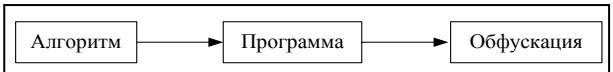


Рис. 1. Структура преобразований программ

Для повышения защищенности программ предлагается обфускацию проводить на ранних этапах создания программного продукта – на этапе разработки алгоритмического обеспечения [10-12]. Варианты структур преобразований для этого случая приведены на рис. 2.

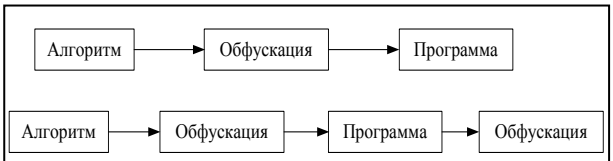


Рис. 2. Варианты структур преобразования программ с обфускацией алгоритма

Для обфускации алгоритма предлагается использовать динамические алгоритмические преобразователи (ДАП) – программно-аппаратные средства, реализующие заданный алгоритм при различных преобразованиях операндов. Поскольку входные операнды постоянно преобразуются, то каждый раз будут активизированы различные «ветви» алгоритма, что затруднит анализ алгоритма и программы.

Обобщенная структура обфускатора алгоритмов приведена на рис. 3.

Преобразователь операндов в соответствии с выбранной группой преобразований и варианта преобразования S производит преобразование входных значений операндов X в соответствующие значения Y, поступающие на входы ДАП. Вариант преобразования формируется на основании управляющих сигналов и (в ряде случаев) специальных сигналов, формируемых ДАП. Выбор структуры ДАП зависит от вида реализуемых алгоритмов.

В общем случае ДАП имеет n условных операторов $Y_1,\ldots,Y_n$. В процессе преобразования им ставятся в соответствие значения условных операторов из множества $X_1,\ldots,X_k$, причем $n > k$, а также константы «0» или «1». В результате преобразований алгоритм, реализуемый ДАП, преобразуется в некоторый алгоритм от меньшего количества переменных.

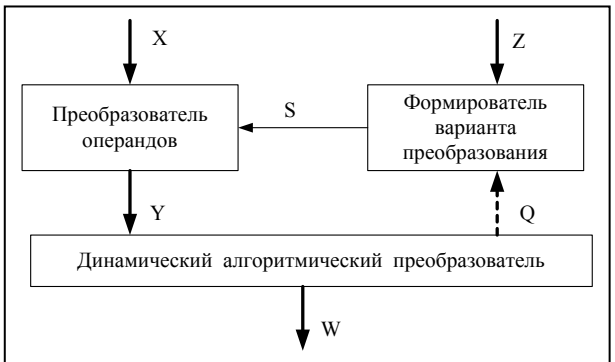


Рис. 3. Обобщенная структура обфускатора алгоритмов

На рис. 4 приведен пример алгоритма A1, реализующего с помощью преобразований {Y1 = 0, Y2 = X1, Y3 = X2}, {Y1 = 1, Y2 = X1, Y3 = $\overline{X2}$}, {Y1 = X2, Y2 = X1, Y3 = X1} алгоритм A2 (рис. 5).

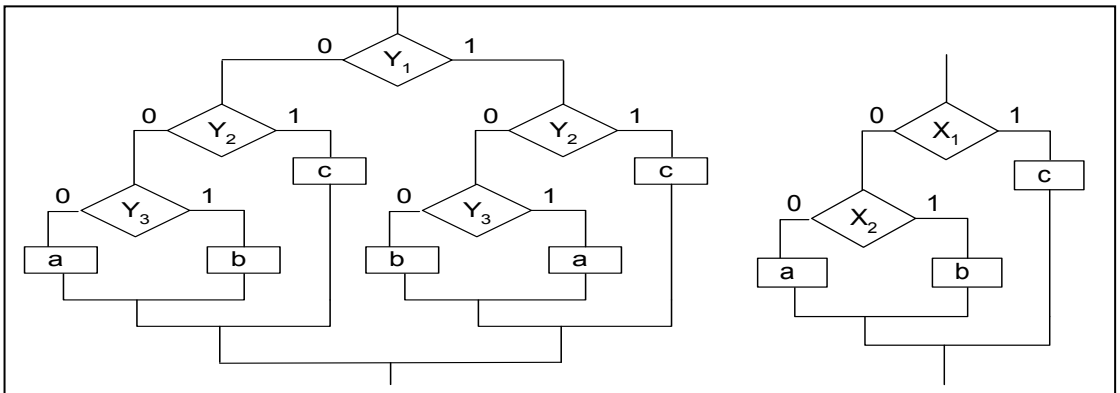


Рис. 4. Алгоритм A1




Рис. 5. Алгоритм A2

Для решения задачи обфускации алгоритмов необходимо решить следующие поэтапные задачи метода (рис. 6):


Рис. 6. Поэтапные задачи метода

− разработать метод построения формализованного описания ДАП, для этого необходимо выбрать или разработать соответствующий математический аппарат;

− разработать метод синтеза структуры ДАП, которую также сделать запутаной;

− для оценки характеристик обфускационных алгоритмов необходимо ввести метрики, характеризующие степень обфускации и критерии сложности.

## Выводы

Рассмотренный подход к обфускации программ позволяет повысить сложность обратной трансляции и анализа программы, т.к. запутывание программы является многоуровневым и начинается на алгоритмическом уровне. Для реализации метода необходимо разработать методы решения поэтапных задач.

## Список литературы

*1. Collberg, Thomborson C., Low D. A Taxonomy of Obfuscating Transformations. Department of Computer Science, The University of Auckland, 1997. – 344 p.*

*2. Казарин О.В. Безопасность программного обеспечения компьютерных систем. – М.: МГУЛ, 2003. – 212 с.*

*3. Чернов А.В. Интегрированная среда для исследования "обфускации" программ. Доклад на конференции, посвящённой 90-летию со дня рождения А.А. Ляпунова. – Россия, Новосибирск, 8-11 октября 2001 года.*

*4. Чернов А.В. Анализ запутывающих преобразований программ // Труды Института Системного программирования РАН. – 2002. – С. 87-96.*

*5. Chow, Gu Y., Johnson H., Zakharov V. An approach to the obfuscation of control-flow of sequential computer programs. – LNCS. – 2001. – P. 144-155.*

*6. Barak B., Goldreich O., Impagliazzo R., Rudich S., Sahai A., Vadhan S., Yang K. On the (Im)possibility of Obfuscating Programs. – LNCS. – 2001. – P. 1-18.*

*7. Cifuentes C., Gough K.J. Decompilation of Binary Programs. Technical report FIT-TR-1994-03. Queensland University of Technology, 1994.*

*8. Collberg C., Thomborson C., Low D. Breaking Abstractions and Unstructuring Data Structures // IEEE Int. Conf. on Computer Languages, ICCL'98. – Chicago, IL, May 1998.*

*9. Collberg C., Thomborson C. Watermarking, Tamper-Proofing, and Obfuscation. – Tools for Software Protection. Technical Report 2000-03. Department of Computer Science, University of Arizona, 2000.*

*10. Чумаченко И.В., Малафеев Е.Е., Шевцов Е.Л. Обфускация алгоритмов. // Матеріали міжнар. наук.-практ. конф. "Розвиток наукових досліджень 2005". – Полтава: ИнтерГрафіка. – 2005. – Т. 8. – С. 134-136.*

*11. Патент України по заявці № 200511063, G06F17/00. Аналізатор обфускаційних алгоритмів. / І.В. Чумаченко, В.А. Дергачов, Є.Є. Малафєєв, Є.Л. Шевцов. – № 200511063; Заявл. 22.11.2005; рішення про видачу деклараційного патенту на корисну модель 21.02.2006.*

*12. Комп'ютерна програма "Програма побудови обфускаційних алгоритмів"/ І.В. Чумаченко, В.А. Дергачов, Є.Є. Малафєєв, Є.Л. Шевцов: Свід. Держ. реєстр. прав автора на твір № 15337. – Зареєстр. в Держ. департ. інтелектуальної власності Мін. освіти і науки України 16.01.2006 р.*

*Поступила в редакцию 24.07.2006*

**Рецензент:** д-р техн. наук, проф. А.Ю. Соколов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.