УДК 004.021

О. В. Казарин, В. Ю. Скиба

ПРИМЕНЕНИЕ САМОКОРРЕКТИРУЮЩИХСЯ СРЕД ДЛЯ ОБЕСПЕЧЕНИЯ ПРОАКТИВНОЙ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

Для создания проактивно безопасных компьютерных систем предлагается применять алгоритмический инструментарий, использующий методологию самотестирующихся и самокорректирующихся программ. Данный подход наряду с использованием самокорректирующихся схем может применяться при создании проактивно безопасных компьютерных систем.

Ключевые слова: *проактивная безопасность, самотестирующаяся и самокорректирующаяся программы, вероятностная оракульная программа, свойство случайной самосводимости, устойчивость, тесты линейной и единичной состоятельности.*

Введение. Проактивная безопасность — это органичное структурное свойство компьютерной системы (КС), которое позволяет ей защищать свои информационные ресурсы и сохранять их функциональность как на этапе разработки, так и на этапе эксплуатации КС.

В настоящей работе предлагается применять для создания проактивно безопасных КС алгоритмический инструментарий, использующий методы самотестирования и самокоррекции программ [1—3], который наряду с самокорректирующимися схемами (см., например, определения из работ [4, 5]), может стать одним из базовых элементов для создания таких КС.

Основные положения методологии создания самотестирующихся и самокорректирующихся программ. Пусть необходимо написать программу P для вычисления функции f так, чтобы $P(x)=f(x)$ для всех значений x. Традиционные методы верификационного анализа и тестирования программ не позволяют убедиться с вероятностью, близкой к единице, в корректности результата выполнения программы, в частности, потому что тестовый набор входных данных, как правило, не перекрывает весь их возможный спектр. Один из методов решения данной проблемы заключается в создании так называемых самокорректирующихся и самотестирующихся программ [6—9].

Чтобы добиться корректного результата выполнения программы P, необходимо написать такую программу T_f, которая позволяла бы оценить вероятность того, что $P(x)\neq f(x)$ для любых x, т.е. *вероятность ошибки* выполнения программы P. При этом T_f может обращаться к P как к своей подпрограмме.

Обязательное условие функционирования программы T_f следующее: ее время выполнения, не учитывающее времени вызовов программы P, должно быть значительно меньше, чем время выполнения любой корректной программы для вычисления f. Самотестирование про-

граммы P должно лишь незначительно увеличивать время ее выполнения. Кроме того, желательно, чтобы длина кода программы T_f составляла константный мультипликативный фактор от длины кода программы [1].

Пусть π — некоторая вычислительная задача и(или) задача поиска решения. Для x, рассматриваемого в качестве входа задачи, пусть $\pi(x)$ обозначает результат решения задачи π. Пусть P — программа (предположительно предназначенная) для решения задачи π, которая останавливается (например, не имеет зацикливаний) на всех входах задачи π. Будем говорить, что P *имеет дефект*, если для некоторого входа x задачи π имеет место $P(x)\neq\pi(x)$.

Определим (*эффективный*) *программный чекер* C_π для задачи π; $C_\pi^P(I, k)$ является произвольной вероятностной машиной Тьюринга, удовлетворяющей следующим условиям. Для любой программы P (предположительно решающей задачу π), выполняемой на всех входах задачи π, для любого элемента I задачи π и для любого положительного k (параметра безопасности) имеет место:

— если программа P не имеет дефектов, т.е. $P(x)=\pi(x)$ для всех входов x, тогда $C_\pi^P(I,k)$ выдаст ответ „норма“ с вероятностью не менее $1–1/2^k$;

— если программа P имеет дефекты, т.е. $P(x)\neq\pi(x)$ для всех входов x, тогда $C_\pi^P(I,k)$ выдаст ответ „сбой“ с вероятностью не менее $1–1/2^k$.

Самокорректирующаяся программа — это вероятностная программа C_f, которая помогает программе P скорректировать саму себя, если только последняя выдает корректный результат с низкой вероятностью ошибки, т.е. для любого x программа C_f вызывает программу P для корректного вычисления $f(x)$, в то время как собственно сама P обладает низкой вероятностью ошибки.

Самотестирующейся / самокорректирующейся программной парой называется пара программ вида (T_f,C_f). Предположим, что пользователь может взять любую программу P, которая целенаправленно вычисляет f и тестирует саму себя при помощи программы T_f. Если P проходит такие тесты, тогда по любому x пользователь может вызвать программу C_f, которая, в свою очередь, вызывает P для корректного вычисления $f(x)$. Даже если программа P некорректна для некоторых входных значений, ее в данном случае все равно можно уверенно использовать для корректного вычисления $f(x)$ для любого значения x.

Кроме того, если удастся написать программу P' для вычисления f, тогда некоторая пара (T_f,C_f) может использоваться для самотестирования и самокоррекции P' без какой-либо ее модификации.

Вероятностная программа M является *вероятностной оракульной*, если она может вызывать другую программу, которая является исполнимой во время выполнения M (M^A означает, что M может делать вызовы программы A).

Пусть программа P предположительно вычисляет функцию f; I является объединением подмножеств I_n, где $n\in N$, $D^p=\{D_n|n\in N\}$ есть множество распределений вероятности D_n над I_n. Далее, пусть $err(P,f,D_n)$ — это вероятность того, что $P(x)\neq f(x)$, где x выбрано случайным образом в соответствии с распределением D_n из подмножества I_n. Пусть β есть некоторый параметр безопасности. Тогда $(\varepsilon_1,\varepsilon_2)$-самотестирующейся программой для функции f в отношении D^p с параметрами $0\leq\varepsilon_1<\varepsilon_2\leq1$ называется вероятностная оракульная программа T_f, которая для параметра безопасности β и любой программы P на входе n имеет следующие свойства:

— если $err(P,f,D_n)\leq\varepsilon_1$, тогда программа T_f^P выдаст на выходе ответ „норма“ с вероятностью не менее $1–\beta$;

— если $err(P,f,D_n)\geq\varepsilon_2$, тогда программа T_f^P выдаст на выходе „сбой“ с вероятностью не менее $1–\beta$.

Оракульная программа C_f с параметром $0\leq\varepsilon<1$ называется ε-самокорректирующейся программой для функции f в отношении множества распределений D^p, которая имеет

следующее свойство по входу n, $x \in I_n$ и β. Если $err(P,f,D_n) \le \varepsilon$, тогда $C_f^P = f(x)$ с вероятностью не менее $1-\beta$.

$(\varepsilon_1,\varepsilon_2,\varepsilon)$-самотестирующейся / самокорректирующейся программной парой для функции f называется пара вероятностных программ (T_f,C_f) такая, что существуют константы $0 \le \varepsilon_1 < \varepsilon_2 \le \varepsilon < 1$ и множество распределений D^p, при которых T_f есть $(\varepsilon_1,\varepsilon_2)$-самотестирующаяся программа для функции f в отношении D^p, а C_f есть ε-самокорректирующаяся программа для функции f в отношении распределения D^p.

Свойство случайной самосводимости. Пусть $x \in I_n$ и пусть $c>1$ есть целое число. Свойство случайной самосводимости заключается в том, что существует алгоритм A_1, работающий в течение времени, пропорционального $n^{O(1)}$ [1], посредством которого функция $f(x)$ может быть выражена через вычислимую функцию F от x, a_1, ..., a_c и $f(a_1)$, ..., $f(a_c)$ и алгоритм A_2, работающий в течение времени, пропорционального $n^{O(1)}$, посредством которого по данному значению x можно вычислить a_1, ..., a_c, где каждое a_i является случайно распределенным над I_n в соответствии с D^p.

Пусть свойство I может быть выражено уравнением $I(x_1, ..., x_k)=0$, где кортеж $\langle x_1, ..., x_k \rangle$ выбирается с распределением E из пространства D^k. Пара (I,E) характеризует семейство функций F, где $f \in F$ тогда и только тогда, когда для всех $(x_1,...,x_k)$ с ненулевой выборкой элементов кортежа из E, $I^f(x_1,...,x_k)=0$. Основным способом самотестирования является идентификация свойства устойчивости для семейства функций F. Неформально (D,D')-устойчивость пары (I,E) для семейства функций G означает следующее: если для программы $P \in G$ свойство $I^P(x_1, ..., x_k)=0$ удовлетворяется с высокой вероятностью, когда $\langle x_1, ..., x_k \rangle$ выбран с распределением E из D^k, существует функция $g \in F \cap G$, которая согласуется с P на большей части входов из D'.

Рассмотрим некоторое свойство линейности (I,E), где $I^f(x_1,x_2,x_3)$ тождественно $f(x_1)+f(x_2)=f(x_3)$ и свойство E означает $(x_1 \in_R Z_p, x_2 \in_R Z_p, x_1+x_2)$. Пара (I,E) характеризует $F=\{f(x)=cx \mid c \in Z_p\}$ — множество всех линейных функций над Z_p. В этом примере G — тривиальное множество всех функций и пара (I,E) устойчива для G.

Было установлено, что программа P удовлетворяет свойству устойчивости, далее можно переходить к тестированию программы на линейную и единичную состоятельность.

Прикладные результаты. *Метод верификации расчетных программ на основе ST-пары функций.* В качестве расчетной рассматривается любая программа, позволяющая получить значение некоторой вычислимой функции. Под верификацией расчетной программы понимается процесс доказательства того, что применение программы позволит получать на некотором входе истинные значения исследуемой функции.

В данном случае предлагается метод создания самотестирующихся программ для верификации расчетных программных модулей [1]. Предлагаемый метод можно использовать для программ, вычисляющих функции особого вида, а именно обладающих свойством случайной самосводимости.

Пусть для функции $Y = f(X)$ существует пара функций $(g_c, h_c)^Y$ таких, что:

$$Y = g_c(f(\alpha_1), ..., f(\alpha_c)),$$
$$X = h_c(\alpha_1, ..., \alpha_c).$$

Легко увидеть, что если значения α_i выбраны из I_n в соответствии с распределением D^p, тогда пара функций $(g_c, h_c)^Y$ обеспечивает выполнение для функции $Y = f(X)$ свойства случайной самосводимости. Пару функций $(g_c, h_c)^Y$ будем называть *ST-парой функций* для функции $Y = f(X)$.

Предположим, что на *ST*-пару функций можно наложить некоторую совокупность ограничений на сложность программной реализации и время выполнения. В этом случае пусть

длина кода программ, реализующих функции g_c и h_c, и время их выполнения составляет константный мультипликативный фактор от длины кода и времени выполнения программы P.

Предлагаемый метод верификации расчетной программы P на основе ST-пары функций для некоторого входного значения вектора X^* заключается в выполнении следующего алгоритма. (Всюду далее, если осуществляется случайный выбор значений, этот выбор выполняется в соответствии с распределением вероятностей D^p.)

1. Определить множество $\mathrm{A}^* = \left\{\alpha_1^*,...,\alpha_c^*\right\}$ такое, что $X^* = h_c\left(\alpha_1^*,...,\alpha_c^*\right)$, где $\alpha_1^*,...,\alpha_c^*$ выбраны случайным образом из входного подмножества I_n.

2. Вызвать программу P для вычисления значения $Y_0^* = f\left(X^*\right)$.

3. Вызвать c раз программу P для вычисления множества значений $\left\{f\left(\alpha_1^*\right),...,f\left(\alpha_c^*\right)\right\}$.

4. Определить значения $Y_1^* = g_c\left(f\left(\alpha_1^*\right),...,f\left(\alpha_c^*\right)\right)$.

5. Если $Y_0^* = Y_1^*$, то принимается решение, что программа P корректна на множестве значений входных параметров $\left\{X^*,\alpha_1^*,...,\alpha_c^*\right\}$, в противном случае данная программа является некорректной.

Таким образом, данный метод не требует вычисления эталонных значений и за одну итерацию позволяет верифицировать корректность программы P на (n+1) значении входных параметров. При этом время верификации можно оценить как

$$T = \sum_{i=1}^{c} t_i + t_x + t_g + t_{h^{-1}} < T_P\left(X\right)\left(1 + c + K_{gh}\left(X,c\right)\right),$$

где t_i и t_x — время выполнения программы P при входных значениях α_i, i = 1, …, c и X^* соответственно;t_g и $t_{h^{-1}}$ — время определения значения функции g_c и множества A^* соответственно; $T_P(X)$ — временная (не асимптотическая) сложность выполнения программы P; $K_{gh}(X, c)$ — коэффициент временной сложности программной реализации функции g_c и определения A^* по отношению ко временной сложности программы P (предположительно, он составляет константный мультипликативный фактор от $T_P(X)$, а его значение меньше единицы).

Для традиционного метода тестирования время выполнения и сравнения полученного результата с эталонным значением составляет:

$$T_0 = \sum_{i=1}^{c} t_i + t_x + \sum_{i=1}^{c} t_i^e + t_x^e > 2T_P\left(X\right)\left(1 + c\right),$$

где t_i^e и t_x^e — время определения эталонных значений функции $Y = f(X)$ при значениях α_i и X^* соответственно (в общем случае не может быть меньше времени выполнения программы).

Следовательно, относительный выигрыш в оперативности предложенного метода верификации (по отношению к методу тестирования программ на основе ее эталонных значений):

$$\Delta T = \frac{T}{T_0} = \frac{\sum_{i=1}^{c} t_i + t_x + t_g + t_{h^{-1}}}{\sum_{i=1}^{c} t_i + t_x + \sum_{i=1}^{c} t_i^e + t_x^e} < \frac{1 + c + K_{gh}}{2\left(1 + c\right)} = 1/2 + \frac{K_{gh}}{2\left(1 + c\right)}.$$

Так как $K_{gh} < 1$, а $c \geq 2$, то получаем относительный выигрыш в оперативности испытания расчетных программ указанного типа (обладающих свойством случайной самосводимости) более чем в 1,5 раза.

Исследования процесса верификации расчетных программ. В качестве примера работоспособности предложенного метода рассмотрим верификацию программы вычисления функции дискретного возведения в степень:

$$y = f_{AM}(x) = A^x \bmod M.$$

Для экспериментальных исследований была выбрана программа EXP из библиотеки базовых криптографических функций CRYPTOOLS [10], реализующая функцию дискретного возведения в степень. Исследования состояли из определения временных характеристик процесса верификации на основе использования *ST*-пары функций и определения возможности обнаружения преднамеренно внесенных программных ошибок.

Для этого были определены следующие *ST*-пары функций:

$$g_2(\alpha_1, \alpha_{23}) = \left[f_{AM}(a_1) f_{AM}(1) \right] (\bmod M) \text{ и } h_2(\alpha_1, \alpha_2) = \alpha_1 + 1;$$

$$g_3^1(\alpha_1, \alpha_2, \alpha_3) = \left[f_{AM}(\alpha_1) f_{AM}(\alpha_2) f_{AM}(\alpha_3) \right] (\bmod M) \text{ и } h_3^1(\alpha_1, \alpha_2, \alpha_3) = \sum_{i=1}^{3} \alpha_i ;$$

$$g_3^2(\alpha_1, \alpha_2, \alpha_3) = \left[f_{f_{AM}(\alpha_1)}(\alpha_2) f_{AM}(\alpha_3) \right] (\bmod M) \text{ и } h_3^2(\alpha_1, \alpha_2, \alpha_3) = \alpha_1 \alpha_2 + \alpha_3;$$

В процессе исследований изменялась используемая *ST*-пара функций и варьировалась размерность параметров A, *M* и аргумента *X*. Результаты экспериментов полностью подтвердили приведенные выше временные зависимости.

Исследование возможности обнаружения с помощью предложенного метода преднамеренно внесенных изменений заключалось в написании программы EXPZ. Спецификация для программ EXP и EXPZ одинакова, различие заключается в том, что программа EXPZ содержит программную закладку деструктивного характера.

Все входные значения, на которых произошел сбой программы, были обнаружены, что в дальнейшем подтвердилось с помощью проверочных тестов, основанных на использовании малой теоремы Ферма и теоремы Эйлера.

Таким образом, предложенный метод позволяет значительно сократить время испытания расчетных программ на предмет выявления непреднамеренных и преднамеренных программных дефектов. При этом по результатам испытаний можно получить экспериментальные оценки вероятности наличия программных дефектов в верифицируемой расчетной программе.

Заключение. Таким образом, можно констатировать, что проактивно безопасные КС привлекательны для пользователей КС, которые, по большому счету, могут даже „не заботиться“ о безопасности своих информационных и функциональных ресурсов. В настоящей работе предлагается лишь один из потенциально мощных инструментов для создания таких систем — самокорректирующиеся программно-аппаратные программы. К тому же, как видно из настоящей работы и работ [2, 3], самокорректирующиеся программы уже сегодня имеют самую широкую прикладную область применения.

СПИСОК ЛИТЕРАТУРЫ

1. *Казарин О. В., Скиба В. Ю.* Об одном методе верификации расчетных программ // Безопасность информационных технологий. 1997. № 3. С. 40—43.

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

3. *Казарин О. В.* Проактивная безопасность вычислительных систем // Математика и безопасность информационных технологий. Мат. конф. МаБИТ-04. М.: МЦНМО, 2005. С. 306—320.

4. *Редькин Н. П.* Асимптотически минимальные самокорректирующиеся схемы для одной последовательности булевых функций // Дискретный анализ операций. Сер. 1. 1996. Т. 3, № 2. С. 62—79.

5. *Редькин Н. П.* О самокорректирующихся схемах и о тестах для инверсных неисправностей элементов // Мат. VIII Междунар. симп. „Дискретная математика и ее приложения“. М.: Изд-во мех.-мат. ф-та МГУ, 2004. С. 4—8.

6. *Blum M., Kannan S.* Designing programs that check their work // Proc. 21st ACM Symp. on Theory of Computing. STOC'89. 1989. P. 86—97.

7. *Blum M., Luby M., Rubinfield R.* Self-testing / correcting with applications to numerical problems // Proc. 22nd ACM Symp. on Theory of Computing. STOC'90. 1990. P. 73—83.

8. *Gemmel P., Lipton R., Rubinfield R., Sudan M., Wigderson A.* Self-testing / correcting for polynomials and for approximate functions // Proc. 23rd ACM Symp. on Theory of Computing. STOC'91. 1991. P. 32—42.

9. *Kumar R. S., Sivakumar D.* Efficient self-testing/self-correcting of linear recurrences // Proc. 37th IEEE Symp. on Foundations of Computer Sci. FOCS'96. 1996. P. 602—611.

10. *Егоркин И. В., Казарин О. В., Скиба В. Ю., Терентьев О. В., Ухлинов Л. М.* Библиотека базовых криптографических функций (Cryptools 1.0). Свидетельство об официальной регистрации программы для ЭВМ № 940518. РосАПО, 1994.

Сведения об авторах

Олег Викторович Казарин — канд. техн. наук; Институт проблем информационной безопасности МГУ им. М. В. Ломоносова, Москва; вед. научный сотрудник; E-mail: okaz2005@yandex.ru, okazarin@iisi.msu.ru

Владимир Юрьевич Скиба — канд. техн. наук; Федеральная таможенная служба, Москва; начальник отдела информационной безопасности; E-mail: skiba@gnivc.customs.ru

Рекомендована институтом

Поступила в редакцию
09.12.08 г.