ПРЕДСТАВЛЕНИЕ ПРОГРАММ ИНВАРИАНТАМИ ПОДОБИЯ ДЛЯ КОНТРОЛЯ ИСКАЖЕНИЯ ВЫЧИСЛЕНИЙ

Харжевская А.В.[1], Ломако А.Г.[2], Петренко С.А.[3]

Для вычислительных систем, выполняющих программы специальных расчетных задач, важно контролировать целостность производимых вычислений. В силу того, что современные средства контроля целостности, основными методами которых являются расчеты контрольных сумм, не позволяют в полной мере контролировать вычислительный процесс, загруженный в оперативную память процессора, необходимо предложить метод, позволяющий исключить возможность модификаций арифметических операций – ключевых для проведения специальных расчетов. В противном случае, незначительная модификация одного оператора может привести к накоплению ошибки, вследствие чего будут получены неверные результаты расчетов. Кроме того, данные результаты могут находиться в конкретном доверительном интервале, поэтому ошибку без дополнительных средств контроля обнаружить невозможно. Авторами статьи предложено решение проблемы контроля корректности выполнения арифметических операторов, используя математический аппарат теории размерности и подобия. Разработанный подход позволяет представить вычислительный процесс определяющими соотношениями или уравнениями подобия в заранее установленных контрольных точках, решение которых позволяет получить инвариантные информативные признаки процесса. Эталонные признаки (или инварианты) образуют паспорт программы. Штатные инварианты формируются в процессе вычислений и сравниваются с эталоном. Такой метод позволяет контролировать семантику производимых вычислений и гарантировать корректность получаемого результата.

***Ключевые слова**: безопасность программ, методы контроля целостности, паспортизация программ, семантика вычислений, организация вычислений с памятью*

DOI: 10.21681/2311-3456-2017-2-9-20

Введение

Выделение инвариантных признаков классификации вычислительных процессов (в данной задаче – на два класса: корректное и некорректное выполнение) тождественно вопросу об изоморфности двух систем относительно некоторого отображения. Для выяснения необходимых и достаточных условий изоморфности систем, а также определения качественных и количественных параметров отображения изоморфизма был разработан математический аппарат теории подобия.

Основные положения теории подобия были сформулированы Седовым Л.И.[4] и развиты Вениковым В.А., Гухманом А.А. и др. Первоначально положения теории разрабатывались применительно к моделированию механических, электрических процессов и процессов теплообмена. Однако в конце 1980-х годов полученные результаты были применены в области моделирования с использованием универсальных цифровых ЭВМ, а затем перенесены для решения гораздо более широкого спектра задач, в том числе защиты информации.

Наиболее детально положения теории подобия проработаны в отношении процессов, описываемых системами однородных степенных многочленов [9]. Основополагающими в теории подобия являются три теоремы: прямая, обратная и π-теорема. Пусть рассматриваются два процесса p_1 и p_2, полные уравнения которых имеют вид:

$$\sum_{i=1}^{q} \varphi_{ui} = 0, \; u = 1,2, \ldots, r;$$

$$\sum_{i=1}^{q} \Phi_{ui} = 0, \; u = 1,2, \ldots, r;$$

где $\varphi_u = \prod_{j=1}^{n} x_j^{\alpha_{u1}}$ и $\Phi_u = \prod_{j=1}^{n} X_j^{\alpha_{u1}}$ – однородные функции своих параметров.

Прямая теорема подобия утверждает, что если процессы однородно подобны, то имеет место система соотношений:

$$\frac{\varphi_{ui}}{\varphi_{uq}} = \frac{\Phi_{ui}}{\Phi_{uq}},$$

$u = 1,2, \ldots, r; \; s = 1,2, \ldots, (q-1).$

---

1 Харжевская Александра Владимировна, Военно-космическая академия им. А.Ф. Можайского, Санкт-Петербург, Россия, a92_zotova@mail.ru

2 Ломако Александр Григорьевич, Заслуженный деятель науки РФ, доктор технических наук, профессор, Военно-космическая академия им. А.Ф. Можайского, Санкт-Петербург, Россия, lomako_ag@mail.ru

3 Петренко Сергей Анатольевич, доктор технических наук, профессор, Университет Иннополис, г. Иннополис, Татарстан, Россия, s.petrenko@rambler.ru

4 Седов Л.И. Методы теории размерности и теории подобия в механике. – М.; Л.: ОГИЗ. Гос. изд-во технико-теорет. лит., 1944. – 136 с.

Выражения

$$\pi_{us} = \frac{\varphi_{ui}}{\varphi_{q}},$$

u = 1,2, …, r; s = 1,2, …, (q-1)

называются критериями или инвариантами подобия и, как следствие из теоремы, численно равны для всех процессов, принадлежащих одному подклассу взаимно подобных процессов.

Таким образом, прямой теоремой формулируются необходимые условия для соотнесения исследуемого процесса с одним из подклассов. Достаточные условия однородного подобия двух процессов приведены в обратной теореме подобия: если существует возможность привести полные уравнения процессов к изоструктурной относительной форме с численно равными инвариантами подобия, то такие процессы являются однородно подобными. Теорема подобия, известная как «π-теорема», позволяет выявить функциональную зависимость между переменными процессами в относительной форме.

Теория подобия для контроля правильности программ была применена учеными В.В. Ковалевым [1], В.А. Романюк, С.А. Петренко [2-4] в диссертационных исследованиях. Следствие из прямой и «π-» теорем подобия позволило сформулировать инвариантные информативные признаки для вычислительных процессов.

Математическая постановка задачи контроля корректности вычислений инвариантами подобия

Представим вычислительный процесс в виде:

$$ВП = <T, X, Y, Z, F, Ф>,$$

где T - множество моментов времени t, в которые наблюдается вычислительный процесс

X, Y - множества входных и выходных параметров вычислительного процесса

Z - множество состояний вычислительного процесса. Всякое состояние z_{jk} ($j = \overline{1,m}$) вычислительного процесса характеризуется в каждый момент времени $t \in T$ последовательностью выполнения арифметических операций в выбранной контрольной точке k.

F - множество операторов переходов f_i, отражающих механизм изменения состояний вычислительного процесса при его выполнении, в том числе при выполняемых арифметических операциях.

$Ф$ - множество операторов выходов ϕ_i, описывающих механизм формирования результата в ходе выполнения вычислений.

Введем следующие обозначения:

λ – отображение нарушения арифметической операции в определенный момент времени t_i при заданных входных параметрах;

ψ – отображение формирования штатных инвариантов вычислительного процесса;

μ – отображение сравнения штатных и эталонных инвариантов вычислительного процесса;

υ – отображение формирования сигнала о некорректности производимых вычислений;

ξ – отображение восстановления арифметических операций на основе эталонных инвариантов подобия;

χ – отображение корректности производимых вычислений на основе восстановленных арифметических операций.

Чтобы исключить возможность скрытной модификации производимых программой вычислений необходимо осуществить динамический контроль выполняемого вычислительного процесса (рис.1). Под динамическим контролем корректности программ специальных расчетных задач будем понимать контроль правильности выполняемой семантики арифметических операций в ходе их реального выполнения [5]. Данные для дина-

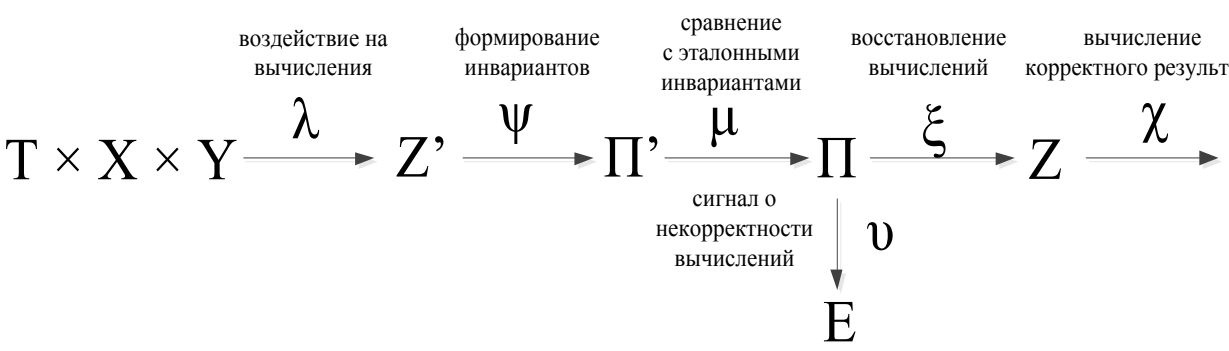

***Рис. 1.** Обобщенная диаграмма отображений процесса восстановления корректности вычислений*

мического контроля необходимо предварительно получить в виде паспорта программы в результате ее дополнительного статического анализа.

Тогда для формирования паспорта программы необходимо:

1. Решить задачу наблюдения (моделирование вычислительного процесса ориентированным управляющим графом программы).

2. Решить задачу представления вычислений уравнениями подобия на линейных участках графа, т.е. преобразовать арифметические операции вида:

$$z_i(x_1, x_2, ..., x_m) = \sum_{j=1}^{p} z_j(x_1, x_2, ..., x_m)$$

в безразмерный вид:

$$[z_{ij}(x_1, x_2, ..., x_m)] = [z_{il}(x_1, x_2, ..., x_m)],$$

$j, l = \overline{1, p}$.

Для обеспечения корректности вычислений необходимо:

1. Решить задачу управления вычислительным процессом путем сравнения семантических инвариантов с паспортом программы, то есть необходимо найти отображения:

ψ: Z'→Π'

μ: Π'→Π

ξ: Π→Z

Ограничения и допущения:

1. Рассматриваемое множество арифметических операций {+, –, *, / , =}

2. $t_i < t_{max}$, где t_i – время восстановления корректности вычислений, t_{max} – предельно допустимое время восстановления корректности вычислений.

Решение данных задач позволит разработать новый метод контроля семантической корректности выполнения специальных расчетных программ, который дополнит возможности существующих средств контроля целостности информации [6-7].

Моделирование вычислительного процесса ориентированным управляющим графом

Для контроля целостности программы, независимо от метода, необходимо выполнить графовое моделирование ее выполнения, в процессе которого анализируется (и исследуется) функциональность программы с учетом структуры предметной области и определенных свойств ее переменных.

Моделирование вычислительного процесса необходимо для наблюдения корректного функционирования программы. Одним из способов представления вычислительного процесса является ориентированный управляющий граф. Представим вычислительный процесс *ВП* в виде управляющего графа программы:

$$\Gamma(B, D)$$

где $B = \{B_i\}$ – множество вершин (линейных участков программы),

$D = \{B \times B\}$ – множество дуг (связей по управлению) между ними.

Здесь каждому линейному участку $B_i \in B$ графа соответствует своя последовательность арифметических операторов, т.е.

$$B_i = (b_{i1}, b_{i2}, ..., b_{il}).$$

Каждому элементарному (без циклов) пути входной в выходную вершину графа соответствует упорядоченная последовательность вершин

$$B^k = (B_1^k, B_2^k, ..., B_t^k)$$

где $B^k \subseteq B$ и $B_i^k = (b_{i1}^k, b_{i2}^k, ..., b_{il}^k)$, $\forall i = \overline{1, p}$ образуют последовательность выполняемых арифметических операторов, называемой реализацией программы или вычислительным процессом. Данные последовательности арифметических выражений являются потенциально опасными фрагментами программы.

Алгоритм вычислительного процесса приводится к графовой форме представления с целью вывести операторы арифметических выражений из управляющих операторов (условных переходов, ветвлений, циклов) [8].

Как результат, управляющий граф процесса приводится к виду, в котором все операторы арифметических выражений сгруппированы на множестве линейных участков программы – вершинах графа (рис.2), в которые необходимо установить контрольные точки (КТ). Контрольные точки необходимы для определения контекста пути, в рамках которого происходят вычисления.

В каждой КТ для арифметических операторов необходимо строить системы определяющих соотношений [9] в виде уравнений подобия. Решение данных систем позволит сформировать матрицы инвариантов, которые позволят контролировать семантику вычислительного процесса.

Построение системы уравнений подобия на линейных участках графа

Исследования показали, что наиболее эффективным способом контроля целостности программ является проверка соотношений, опирающихся на теоретически обоснованные соотношения и свойства вычислений [11]. Ключевым соотношением в

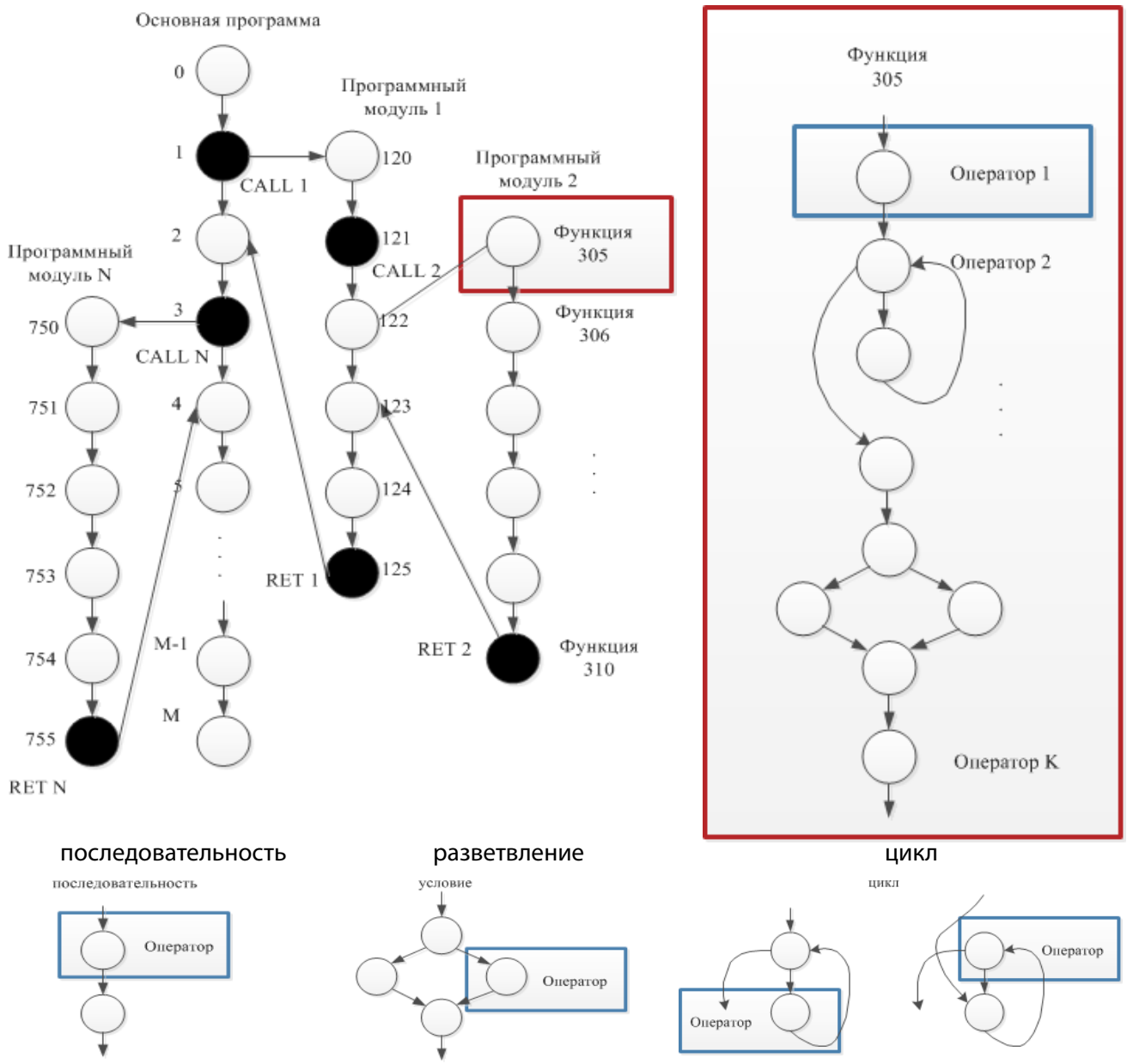

Рис. 2. *Декомпозиция управляющего графа программы*

подходе к обнаружению параметров некорректного функционирования вычислительных процессов будем использовать инвариант, под которым понимается нечто неизменное при выполнении программы . Задачи генерации инвариантов из различных представлений программы наименее формализованы и плохо поддаются реализации. В динамике выполнения программы полностью вычислимыми (воспроизводимыми) остаются только семантические инварианты (так как не зависят от конкретных значений программных переменных). На данный момент не существует реализованных способов контроля семантики в вычислительных процессах.

Реализацию B^k управляющего графа программы можно представить в виде упорядоченной последовательности первичных соотношений [9], соответствующих арифметическим операторам:

$$\begin{cases} y_1 = f_1^k(x_1, x_2, \ldots, x_N) \\ y_2 = f_2^k(x_1, x_2, \ldots, x_N, y_1) \\ \ldots \\ y_M = f_M^k(x_1, x_2, \ldots, x_N, y_1, y_2, \ldots, y_{M-1}) \end{cases} \quad (1)$$

Выполнив в правых частях соотношений суперпозицию $\{y_i\}$ на X, получим систему соотношений инвариантных относительно перемещения:

$$\begin{cases} y_1 = z_1^k(x_1, x_2, \ldots, x_N) \\ y_2 = z_2^k(x_1, x_2, \ldots, x_N) \\ \ldots \\ y_m = z_m^k(x_1, x_2, \ldots, x_N) \end{cases} \quad (2)$$

Соотношение $y_i = z_i^k(x_1, x_2, ..., x_N)$ можно представить в виде:

$$y_i = \sum_{i=1}^{p_i} z_j(x_1, x_2, ..., x_N), \quad (3)$$

где $z_j(x_1, x_2, ..., x_N)$ – степенной одночлен.

В соответствии с правилом Фурье слагаемые суммы (3) должны быть однородными по размерностям, т.е.

$$[y_i] = [z_{ij}(x_1, x_2, ..., x_N)], \ j = \overline{1, p_i} \text{ или}$$

$$[z_{ij}(x_1, x_2, ..., x_N)] = [z_{il}(x_1, x_2, ..., x_N)], \quad (4)$$

$$j, l = \overline{1, p_i}$$

Система (4) является системой определяющих соотношений или системой уравнений подобия.

С помощью функции $\rho = X \to [X]$ сопоставим каждому $x_j \in X$ некоторую абстрактную размерность $[x_j] \in [X]$. Тогда размерности слагаемых суммы (3) выразятся как

$$[z_{ij}(x_1, x_2, ..., x_n)] = \prod_{n=1}^{N} [x_n]^{\lambda_{jn}}, \ j = \overline{1, p_i} \quad (5)$$

Используя (4) и (5), построим систему определяющих соотношений

$$\prod_{n=1}^{N} [x_n]^{\lambda_{jn}} = \prod_{n=1}^{N} [x_n]^{\lambda_{ln}}, \ j, l = \overline{1, p_i},$$

которую преобразуем к виду

$$\prod_{n=1}^{N} [x_n]^{\lambda_{jn} - \lambda_{ln}} = 1, \ j, l = \overline{1, p_i} \quad (6)$$

Используя прием логарифмирования, как это обычно делается при анализе соотношений подобия, из системы (6) получим однородную систему линейных уравнений

$$\sum_{n=1}^{N} (\lambda_{jn} - \lambda_{ln}) \ln[x_n] = 0, \ j, l = \overline{1, p_i} \quad (7)$$

Выражение (7) является критерием семантической корректности.

Выполнив подобное построение для $\forall B_i^k \in B^k$ получим для k-ой реализации систему однородных линейных уравнений:

$$A^k \omega = 0 \quad (8)$$

В общем случае, можно предположить, что функция $\rho = X \to [X]$ сюръективна и, следовательно, реализация B^k представляется матрицей $A^k = \|a_j\|$ размером $m_k \times n_k$, у которой число столбцов не меньше, чем число строк, т.е. $n_k \geq m_k$.

Будем говорить, что реализация B^k представительна, если ей соответствует матрица A^k с $m_k \geq 1$, т.е. реализация позволяет построить хотя бы один критерий подобия.

Обычно программа соответствует отдельному функциональному модулю или состоит из взаимосвязанной группы таковых и описывает общее решение некоторой задачи. Каждая из реализаций $B^k \in B$ описывает частное решение этой же задачи, соответствующее определенным значениям компонент *X*. Так как $B^k \cap B^l \neq \varnothing$, $\forall B^k, B^l \in B$, то структура математических зависимостей при переходе от одной реализации к другой должна в основном сохраняться, т.е. критерии подобия должны быть общими. Тогда матрицы $\{A^k\}$, соответствующие реализациям $\{B^k\}$, могут быть объединены в одну систему.

Пусть программа имеет q реализаций. Обозначим через *A* объединение матриц $\{A^k\}$, соответствующих реализациям $\{B^k\}$, т.е.

$$A = \begin{pmatrix} A_1 \\ \dots \\ A_q \end{pmatrix} \quad (9)$$

Построение A можно осуществить по выборочным, обеспечивающим покрытие вершин реализациям.

Таким образом объединение матриц A является частью паспорта программы и представляет собой базу данных семантических эталонов $\{A^k\}$ для линейных участков программы $\{B^k\}$.

Пример представления вычислений уравнениями подобия

Рассмотрим в качестве примера оператор присваивания:

$$p = a * b + c / (d - e). \quad (10)$$

Корректное выражение должно порождаться некоторой выбранной грамматикой, которая зависит как от возможных значений термов, так и от выбранного набора операций. Для контекстно-свободной грамматики каждому выражению может быть поставлено в соответствие дерево вывода единственным способом. Таким образом, дерево вывода может использоваться как альтернативное представление выражения.

При построении дерева по выражению свою роль играет порядок вычислений. Очевидно, что значения вершин-потомков вычисляются раньше, чем значение вершины-предка. Поэтому на вершине дерева будет находиться операция, выполняемая в последнюю очередь. Для однозначного построения дерева нужно определить порядок

вычисления операций в выражении, учитывая их приоритеты и порядок выполнения операций с одинаковым приоритетом, в том числе при вычислении одной и той же операции (свойство ассоциативности). Обычно такие выражения вычисляются слева направо.

Построенное дерево с учетом порядка вычисления будет однозначно соответствовать заданному выражению. Приведенному выше выражению (10) будет соответствовать дерево, представленное на рисунке 3:

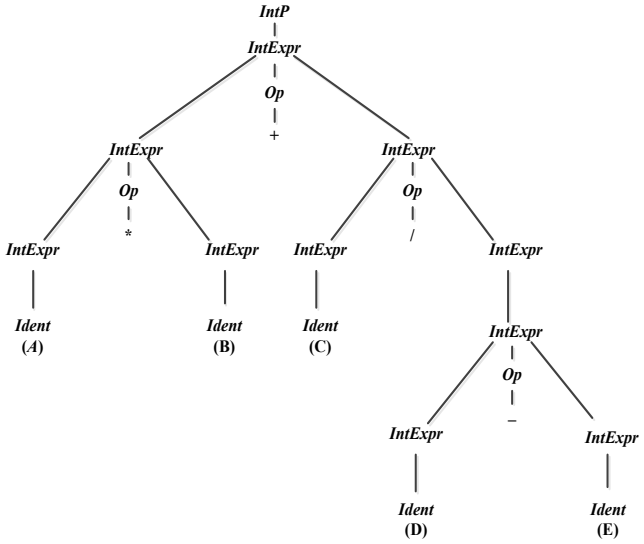

***Рис.3**. Дерево порождения арифметического выражения*

Формализуем арифметические выражения:

Пусть Op {+, –, *, /} – множество рассматриваемых арифметических операций.

Terms – множество термов, включающее возможные объекты, которые могут быть аргументами операции.

Expr – множество всех возможных выражений, причем $Terms \subset Expr$.

$elem(o,e) \in Expr$ – множество других элементов, причем $o \in Op$, $e \in Expr$.

Таким образом, арифметическое выражение представляет собой либо терм, либо операцию, связывающую несколько выражений.

Выражение (10) при множестве термов *Terms* = {*p, a, b, c, d, e*} и множестве бинарных операций Op {+, -, *, /} будет представлено как: *elem*: (=, p, (+, (*, a, b), (/, c, (–, d, e)))).

Корректность выполнения арифметического оператора можно оценить, используя соответствующую семантическую функцию. Применительно к выражениям семантическая функция $T: a \to [\,a\,]$ ставит в соответствие каждому аргументу a некоторую абстрактную сущность или размерность $[a]$. Таким образом, арифметические операции, выполняемые над программными переменными во время выполнения программы, фактически являются операциями над физическими размерностями, а отражения семантики, которые осуществляются во время выполнения – линейными отображениями. Аксиоматика расширенной семантической алгебры, определяющая операции над размерностями переменных, представлена в табл. 1.

Для корректно выполняющейся программы в контексте данного оператора должны выполняться следующие соотношения между физическими размерностями термов {*p, a, b, c, d, e* }:

$$[p] = [a * b] = [a][b],$$
$$[d] = [e], \qquad (11)$$
$$[p] = [c / (d-e)] = [c][d]^{-1} = [c][e]^{-1},$$

где [X] – физическая размерность объекта X.

Модель вычислений в памяти можно представить, используя аппарат контекстно-свободных грамматик. Он позволяет описать структуру про-

Таблица 1.
Операции над размерностями программных переменных

Оператор	Обозначение	Условие корректности	Линейные уравнения	Критерии подобия
Сложение	R = L + P	[L] = [P]	$[R]^0[L]^1[P]^{-1} = 1$	0 1 -1
Вычитание	R = L – P	[L] = [P]	$[R]^0[L]^1[P]^{-1} = 1$	0 1 -1
Умножение	R = L * P	[R] = [L][P]	$[R]^1[L]^{-1}[P]^{-1} = 1$	1 -1 -1
Деление	R = L / P	$[R] = [L][P]^{-1}$	$[R]^1[L]^{-1}[P]^1 = 1$	1 -1 1
Возведение в степень	$R = L^s$	$[R] = [L]^s$	$[R]^1[L]^{-s}[P]^0 = 1$	1 -s 0
Присваивание	L = P	[L] = [P]	$[R]^0[L]^1[P]^{-1} = 1$	0 1 -1

где R – результат операции; L, R – левый и правый операнды; [ ] – размерность.

Таблица 2.
Множества нетерминальных символов

НЕТЕРМИНАЛЬНЫЙ СИМВОЛ N	ОБОБЩАЮЩИЙ ПРИЗНАК	ТЕРМИНАЛЬНЫЕ СИМВОЛЫ Σ
Addition	Команды сложения	fiadd | fadd | faddp | …
Subtraction	Команды вычитания	fisub | fsub | fsubr | …
Multiplication	Команды умножения	fimul | fmul | fmulp | …
Division	Команды деления	fidiv | fdiv | fdivr | …
Appropriate	Команды передачи данных	fist | fst | fstp | …

цесса вычислений в целом. Контекстно-свободная грамматика имеет следующий вид:

$$G = (\Sigma, N, R, S), \qquad (12)$$

где $\Sigma = \{identifier,\ condtant,\ adress\ \ldots\ register\}$ – набор терминальных символов языка ассемблер (таблица 1);

$N = \{Addition,\ Substraction,\ Multiplication,\ Division,\ Appropriate\}$ – набор нетерминальных символов;

$R = \{AddCommand,\ SubComand\ Mulcommand,\ \ldots,\ DivCommand\}$ –множество правил вывода;

$S \in \Sigma$ – стартовый символ.

Терминальные символы включают в себя лексемы команд арифметического сопроцессора, в том числе команды сложения, вычитания, умножения, деления, присваивания (передачи данных). Набор нетерминальных символов представляет собой множества лексем, объединенных по обобщающему признаку, а также их комбинации, с использованием продукций. Пример нетерминальных символов приведен в табл. 2.

Правило вывода, представленное выражением (13) обуславливает применение команды «fadd». Таким образом, представим все возможные правила вывода в языке ассемблер.

$$\begin{aligned} AddCommand \rightarrow\ & Addition\_Register, Address \\ & \mid Addition\_Register, Register \\ & \mid Addition\_Register, Register \Rightarrow faddp\ st(1), st \\ & \mid \ldots \end{aligned}$$

где *Addition* – нетерминальное множество команд сложения сопроцессора;

Register – нетерминальное множество регистров стека сопроцессора;

Address – множество идентификаторов памяти или непосредственно адресов памяти.

Каждому выводу в КС-грамматике, начинающемуся с нетерминального символа, однозначно сопоставляется ориентированный граф, являющийся деревом и называемый деревом вывода (разбора).

Пример дерева вывода, относящегося к дизассемблированному коду выражения (10), а также его представление в виде уравнений подобия в терминах теории размерностей приведен на рис. 4.

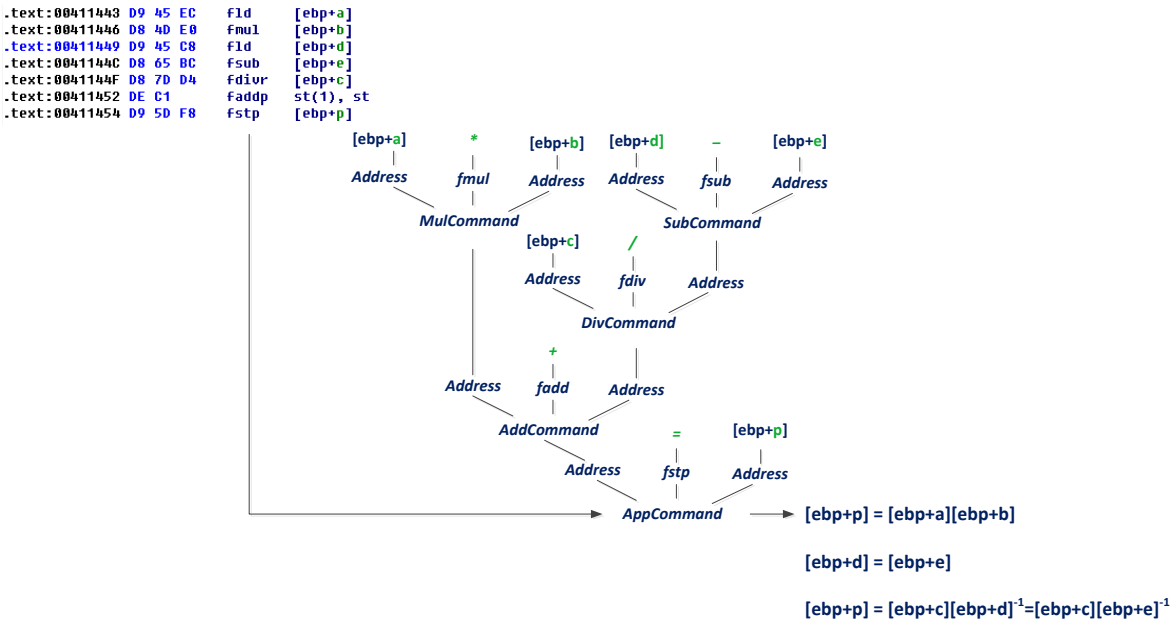

Рис.4. ***Рис. 4.*** *Представление вычислений уравнениями подобия*

Решением данной системы уравнений является матрица коэффициентов подобия, построенная следующим образом:

$$\begin{array}{l}[ebp+p] = [ebp+a][ebp+b] \\ [ebp+d] = [ebp+e] \\ [ebp+p] = [ebp+c][ebp+d]^{-1}=[ebp+c][ebp+e]^{-1}\end{array} \Rightarrow \begin{array}{l}[ebp+p]^1[ebp+a]^{-1}[ebp+b]^{-1}[ebp+c]^0[ebp+d]^0[ebp+e]^0=1 \\ [ebp+p]^0[ebp+a]^0[ebp+b]^0[ebp+c]^0[ebp+d]^1\ [ebp+e]^{-1}=1 \\ [ebp+p]^0[ebp+a]^0[ebp+b]^0[ebp+c]^{-1}[ebp+d]^1[ebp+e]^0=1\end{array}$$

Логарифмируя, получаем однородную систему линейных уравнений с матрицей коэффициентов:

$$A^1 = \begin{pmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 \end{pmatrix}$$

Для организации построения соотношений подобия необходимо построить трансляционную грамматику для операторов присваивания арифметического типа. Трансляционные (атрибутные) грамматики помимо синтаксиса позволяют описать символы действия, которые реализованы в виде функций, процедур, алгоритмов. С точки зрения размерностей эти функции должны реализовывать алгоритмические вычисления и построение соотношений подобия, степенные одночлены, уравнения и решения.

Таким образом, решение задач наблюдения (управляющий граф) и представления (уравнения подобия) вычислений позволило сформировать облик системы контроля искажений и восстановления процессов вычислений в рамках решения задачи управления вычислительным процессом.

Облик системы контроля искажений и восстановления процессов вычислений

Схема этапов контроля искажений и восстановления процессов вычислений включает подготовительный и основные этапы. К подготовительному этапу относится этап формирования паспорта программы в инвариантах подобия, к основным относятся этапы: формирования инвариантов подобия в условиях воздействий, формирования базы данных инвариантов подобия в контрольных точках УГ программы, проверки критерия семантической корректности вычислительный процессов, формирования сигнала о нарушении семантики вычислений и частичного восстановления вычислений по паспорту программы.

Общее представление вычислительной системы, производящей корректные вычисления в условиях скрытых воздействий отражено на рисунке 5.

Раскроем этапы схемы контроля искажений и восстановления процессов вычислений более подробно.

Этап 1. Формирование паспорта программы в инвариантах подобия.

Для осуществления динамического контроля необходимо использовать результаты статической верификации в виде паспорта программы.

На этапе статической верификации по дизассемблированному коду корректных вычислений (рис. 6) строится управляющий граф программы.

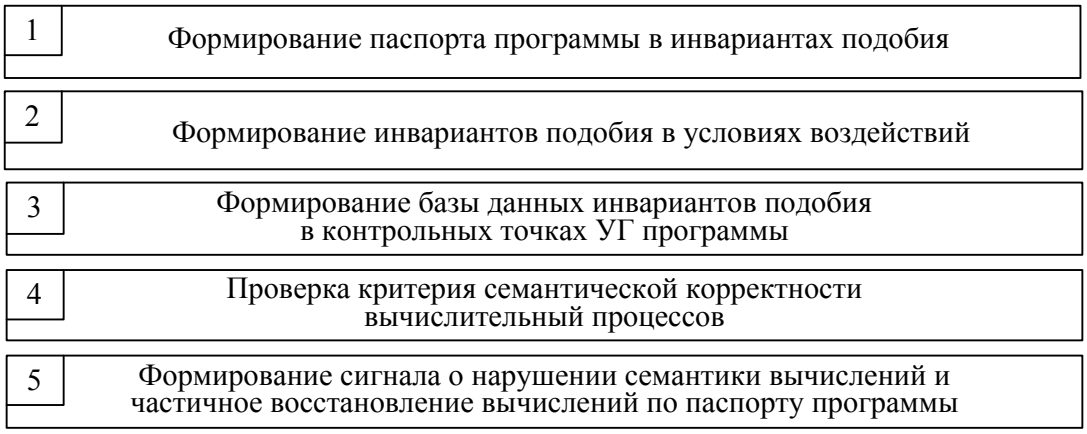

***Рис. 5**. Схема контроля искажений и восстановления процессов вычислений*

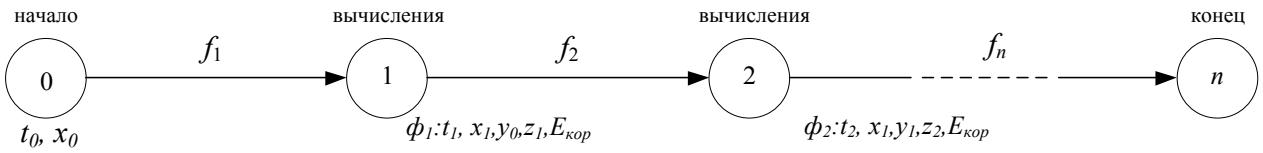

Рис. 6. *Схема корректных вычислений*

В каждой КТ для каждого арифметического оператора генерируется дерево порождения арифметического выражения с целью построения системы линейных однородных уравнений в терминах размерностей. Результатом решения систем уравнений для каждого линейного участка программы является матрица инвариантов подобия. Базу данных семантических эталонов составляют эталонные матрицы инвариантов подобия для каждой КТ (рис.7).

Рис. 7. *Схема формирования паспорта программы в инвариантах подобия*

Этап 2. Формирование инвариантов подобия в условиях воздействий

Формирование инвариантов подобия вычислительного процесса, подвергающегося скрытному воздействию арифметических операций, происходит по тому же алгоритму, что и формирование эталонных инвариантов вычислительного процесса.

Для заданной программы формируется множество контрольных точек (КТ), которые встраиваются в исследуемую программу. Исходной моделью программы является управляющий граф процесса вычислений в терминах линейных участков программы.

Во встроенных КТ для каждого линейного участка программы, где происходят вычисления, происходит анализ уравнений подобия и строится матрица коэффициентов (рис. 8).

Некорректные вычисления будут отличаться множеством состояний вычислительного процесса Z, т.е. последовательностью выполняемых арифметических операторов. Схема некорректных вычислений представлена на рисунке 9.

Этап 3. Формирование базы данных инвариантов подобия в контрольных точках УГП.

На данном этапе построенные для каждой контрольной точки матрицы инвариантов подобия формируют базу данных инвариантов подобия. Схема добавления матриц в базу данных представлена на рисунке 10.

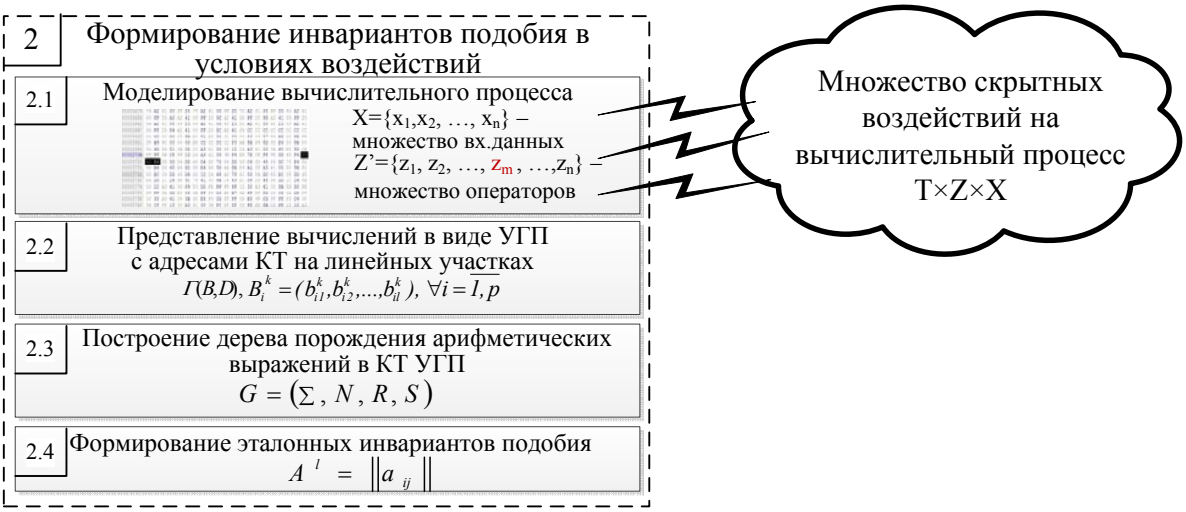

Рис. 8. *Схема формирования инвариантов подобия в условиях воздействий*

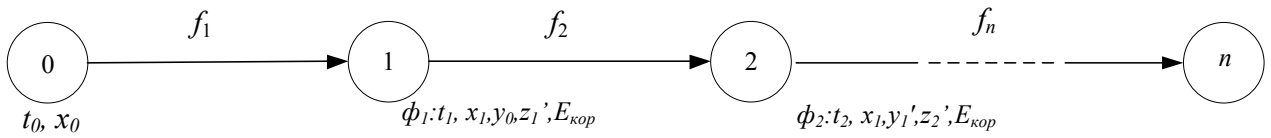

***Рис.9**. Схема выполнения некорректных вычислений*

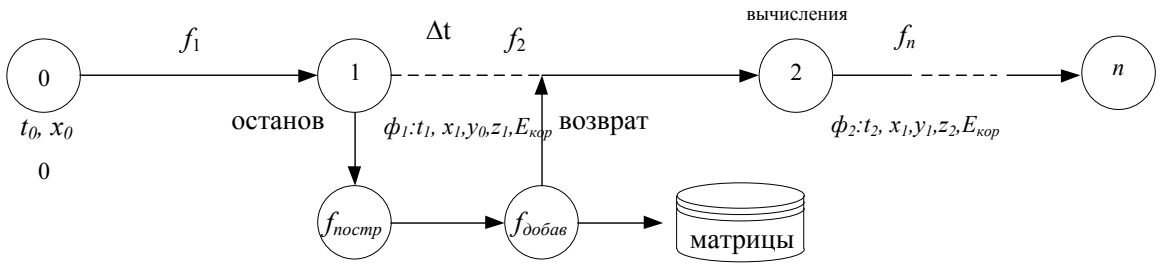

***Рис.10**. Схема формирования базы данных матриц инвариантов подобия*

Этап 4. Проверка критерия семантической корректности вычислительных процессов.

Для контроля корректности семантики выполняемых вычислений необходимо используя эталонную и штатную матрицы инвариантов (рис.11) проверить критерий семантической корректности по формуле (7). Необходимым критерием семантической корректности вычислений является существование решение системы, в котором ни одна из переменных ($\ln[x_j]$) не обращена в 0.

Если проверка для данной контрольной точки выполнилась, то переходим к проверке критерия в следующей КТ, пока не завершится программа.

Этап 5. Формирование сигнала о нарушении семантики вычислений и частичное восстановление вычислений по паспорту программы

В случае обнаружения нарушения семантической корректности выполнения программы, то есть если для данной контрольной точки $\lambda_{jn} - \lambda_{in} \neq 0$, то формируется сигнал и реализуется попытка вос-

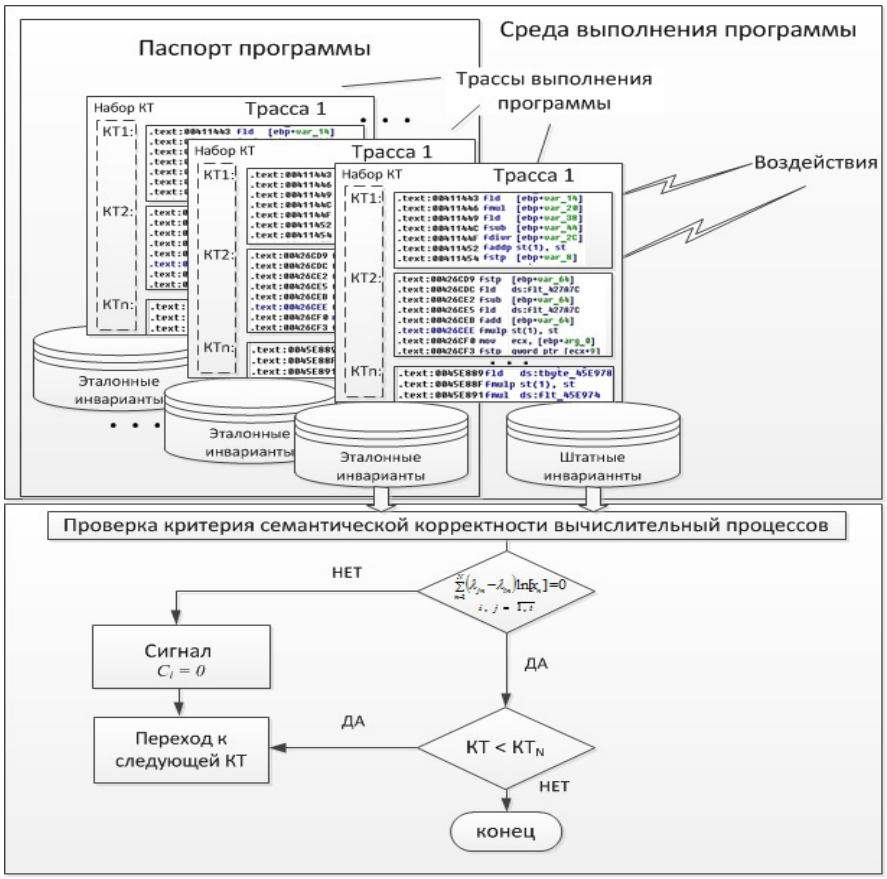

***Рис.11.** Схема проверки корректности вычислительных процессов*

становления вычислений по обратному преобразованию инвариантов эталонной матрицы (рис.11).

Данный подход позволяет определить не просто факт нарушения семантики вычислений, а указать конкретное место воздействия на программу, используя механизм внедрения контрольных точек.

Таким образом, в общем виде представлено абстрактное решение научной задачи разработки научно-методического аппарата верификации корректности алгоритмов программ при искажениях вычислений.

Заключение

Исследование современной ситуации в области обнаружения скрытных воздействий на вычисления, а также анализ нормативно-правовых аспектов в сфере информационной безопасности РФ позволили сделать вывод об актуальности разработки новых моделей и методов рационального применения средств защиты информации для выполнения требований своевременности обнаружения ИТВ, учитывая скрытность их проведения вероятным противником [12-15]. Современные методы верификации и контроля не распространяются на этап выполнения программ, когда естественные ошибки и уязвимости, а также преднамеренные закладки проявляются через используемые данные и ресурсы [16, 17].

Анализ следствий теории подобия и теории размерностей позволили предложить применение новых информативных признаков – инвариантов подобия для контроля корректности вычислительных процессов. Применение инвариантов подобия в терминах теории размерностей для обнаружения нарушения семантической корректности вычислительных процессов позволяет максимально приблизить систему контроля искажений и восстановления процессов вычислений к семантике контролируемого процесса. Полученные результаты позволяют представить вычислительный процесс в виде соответствующей системы уравнений размерностей и инвариантов подобия вычислений, а ее решение позволяет исследовать семантику вычислений в условиях скрытных модификаций. Построенная система контроля искажений и восстановления процессов вычислений подтверждает результативность предлагаемого подхода.

***Рецензент**: Марков Алексей Сергеевич, доктор технических наук, профессор кафедры ИУ8 «Информационная безопасность» МГТУ им.Н.Э.Баумана, Москва, a.markov@bmstu.ru*

Литература:

1. Зотова А.В., Компаниец Р.И., Ковалев В.В. Способ паспортизации расчетных алгоритмов программ // Защита информации. Инсайд. 2016. № 5 (71). С. 26-33.
2. Петренко С.А., Ломако А.Г., Амельченкова О.Н., Зотова А.В. Вычисления с памятью критически важных информационных систем в условиях кибератак // Защита информации. Инсайд. 2012. № 6 (48). С. 58-69.
3. Петренко С.А., Шамсутдинов Т.И., Петренко А.С. Научно-технические задачи развития ситуационных центров в Российской Федерации // Защита информации. Инсайд. 2016. № 6 (72). С. 22-27.
4. Петренко А.С., Петренко С.А. Проектирование корпоративного сегмента СОПКА // Защита информации. Инсайд. 2016. № 6 (72). С. 28-30.
5. Ломако А.Г., Еремеев М.А., Новиков В.А. Метод выявления дефектов и недокументированных возможностей программ // Информационное противодействие угрозам терроризма. 2010. № 14. С. 46-49.
6. Зима В.М., Котухов М.М., Ломако А. Г., Марков А.С., Молдовян А.А. Разработка систем информационно-компьютерной безопасности. СПб: ВАК им. А. Ф. Можайского, 2003. 327 с.
7. Козачок А.В., Кочетков Е.В. Обоснование возможности применения верификации программ для обнаружения вредоносного кода // Вопросы кибербезопасности. 2016. № 3 (16). С. 25-32.
8. Новиков В.А., Компаниец Р.И., Ломако А.Г. «Спецпроверка» программ // Защита информации. Инсайд. 2006. № 3 (9). С. 18-27.
9. Ковалев В.В., Компаниец Р.И., Новиков В.А. Верификация программ на основе соотношений подобия // Труды СПИИРАН. 2015. № 1. С. 233-245.
11. Петренко С.А., Курбатов В.А., Бугаев И.А., Петренко А.С. Когнитивная система раннего предупреждения о компьютерном нападении // Защита информации. Инсайд. 2016. № 3 (69). С. 74-82.
12. Петренко С.А., Петренко А.С. Новая доктрина информационной безопасности Российской Федерации // Защита информации. Инсайд. 2017. № 1 (73). С. 33-39.
13. Петренко А.С., Петренко С.А. Первые межгосударственные киберучения стран СНГ: «Кибер-Антитеррор-2016» // Защита информации. Инсайд. 2016. № 5 (71). С. 57-63.
14. Петренко А.А., Петренко С.А. Киберучения: методические рекомендации ENISA // Вопросы кибербезопасности. 2015. № 3 (11). С. 2-14.
15. Петренко А.А., Петренко С.А. НИОКР Агенства DARPA в области кибербезопасности // Вопросы кибербезопасности. 2015. № 4 (12). С. 2-22.
16. Barabanov A.V., Markov A.S., Tsirlov V.L. Methodological framework for analysis and synthesis of a set of secure software development controls // Journal of Theoretical and Applied Information Technology. 2016. V. 88. N 1. P. 77-88.
17. Markov A.S., Fadin A.A., Tsirlov V.L. Multilevel metamodel for heuristic search of vulnerabilities in the software source code // International Journal of Control Theory and Applications. 2016. V. 9. N 30. P. 313-320.

REPRESENTING PROGRAMS WITH SIMILARITY INVARIANTS FOR MONITORING TAMPERING WITH CALCULATIONS

Kharzhevskaya A.[5], Lomako A.[6], Petrenko S.[7]

It is important for the computation systems that run special computing programs to control integrity of the calculations. Since modern integrity control tools mainly using control computation methods cannot comprehensively control the computation process that is uploaded into the processor's RAM, it is necessary to suggest a method aimed to prevent any modification in the arithmetic operations, which are the key operations for special calculations. Otherwise, insignificant modification of one operator may result in accumulation of errors, which will lead to incorrect computation results. Moreover, these results may fall within a certain confidence interval, therefore an error cannot be identified without additional control tools. The authors of the paper suggest solving the problem of the accuracy control of the arithmetic operators using mathematical tools of the theory of dimensions and similarity. The developed approach can present a computational process as defining relations or similarity equations in pre–set control points, which solution results in obtaining invariant information properties of the process. Reference attributes (or invariants) form a program passport. Conventional invariants are generated in the process of calculations and are compared with the reference. Such method can be used to monitor the semantics of the calculations and guarantee accuracy of the result.

***Keywords**: software security, integrity check method, software certification, calculations semantics, organization of calculations with memory*

Reference:

1. Zotova A.V., Kompaniets R.I., Kovalev V.V. Sposob pasportizatsii raschetnykh algoritmov programm, Zashchita informatsii. Insayd. 2016, N 5 (71), pp. 26-33.
2. Petrenko S.A., Lomako A.G., Amel'chenkova O.N., Zotova A.V. Vychisleniya s pamyat'yu kriticheski vazhnykh informatsionnykh sistem v usloviyakh kiberatak, Zashchita informatsii. Insayd. 2012, N 6 (48), pp. 58-69.
3. Petrenko S.A., Shamsutdinov T.I., Petrenko A.S. Nauchno-tekhnicheskie zadachi razvitiya situatsionnykh tsentrov v Rossiyskoy Federatsii, Zashchita informatsii. Insayd. 2016, N 6 (72), pp. 22-27.
4. Petrenko A.S., Petrenko S.A. Proektirovanie korporativnogo segmenta SOPKA, Zashchita informatsii. Insayd. 2016, N 6 (72), pp. 28-30.
5. Lomako A.G., Eremeev M.A., Novikov V.A. Metod vyyavleniya defektov i nedokumentirovannykh vozmozhnostey programm, Informatsionnoe protivodeystvie ugrozam terrorizma, 2010, N 14, pp. 46-49
6. Zima V. M., Kotukhov M. M., Lomako A. G., Markov A. S., Moldovyan A. A. Razrabotka sistem informatsionno-komp'yuternoy bezopasnosti. SPb: VAK im. A. F. Mozhayskogo, 2003. 327 p.
7. Kozachok A.V., Kochetkov E.V. Obosnovanie vozmozhnosti primeneniya verifikatsii programm dlya obnaruzheniya vredonosnogo koda, Voprosy kiberbezopasnosti. 2016, N 3 (16), pp. 25-32.
8. Novikov V.A., Kompaniets R.I., Lomako A.G. «Spetsproverka» programm, Zashchita informatsii. Insayd. – 2006, N 3. – P. 18 – 27.
9. Kovalev V.V., Kompaniets R.I., Novikov V.A. Verifikatsiya programm na osnove sootnosheniy podobiya, Trudy SPIIRAN. 2015, N 1, pp. 233-245.
11. Petrenko S.A., Kurbatov V.A., Bugaev I.A., Petrenko A.S. Kognitivnaya sistema rannego preduprezhdeniya o komp'yuternom napadenii, Zashchita informatsii. Insayd. 2016, N 3 (69), pp. 74-82.
12. Petrenko S.A., Petrenko A.S. Novaya doktrina informatsionnoy bezopasnosti Rossiyskoy Federatsii, Zashchita informatsii. Insayd. 2017, N 1 (73), pp. 33-39.
13. Petrenko A.S., Petrenko S.A. Pervye mezhgosudarstvennye kiberucheniya stran SNG: «Kiber-Antiterror-2016», Zashchita informatsii. Insayd. 2016, N 5 (71), pp. 57-63.
14. Petrenko A.A., Petrenko S.A. Kiberucheniya: metodicheskie rekomendatsii ENISA, Voprosy kiberbezopasnosti, 2015. N 3 (11), pp. 2-14.
15. Petrenko A.A., Petrenko S.A. NIOKR Agenstva DARPA v oblasti kiberbezopasnosti, Voprosy kiberbezopasnosti. 2015, N 4 (12), pp. 2-22.
16. Barabanov A.V., Markov A.S., Tsirlov V.L. Methodological framework for analysis and synthesis of a set of secure software development controls, Journal of Theoretical and Applied Information Technology. 2016. V. 88. N 1. P. 77-88.
17. Markov A.S., Fadin A.A., Tsirlov V.L. Multilevel metamodel for heuristic search of vulnerabilities in the software source code, International Journal of Control Theory and Applications. 2016. V. 9. N 30. P. 313-320.

5 Alexsandra Kharzhevskaya, Mozhaisky Military Space Academy, St. Petersburg, Russia, a92_zotova@mail.ru

6 Aleksandr Lomako, Dr.Sc., Professor, Honored Scientist of Russia, Mozhaisky Military Space Academy, Saint-Petersburg, Russia, lomako_ag@mail.ru

7 Sergey Petrenko, Dr.Sc., Professor, University of Innopolis, Innopolis, Tatarstan, Russia, s.petrenko@rambler.ru