https://doi.org/10.25559/SITITO.020.202402.436-444
УДК 004.416.2+004.4'242

Оригинальная статья

Метод мониторинга корректности исполнения процессов в операционных системах на основе контроля систем размерностей переменных

В. Г. Абрамов, Д. А. Пучкин*
ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова», г. Москва, Российская Федерация
Адрес: 119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1
* danila.puchkin@kaspersky.com

Аннотация

Задача контроля корректности вычислений встает перед разработчиками программных продуктов для любой операционной системы. При этом возникает необходимость как определения момента некорректного вычисления, так и локализации команд процесса, приведших к некорректным вычислениям. Для решения данной задачи использование подходов, основанных на контрольном таймере, дублировании процессов или наложенных средствах контроля, является или избыточным, или не гарантирует локализацию нарушения вычислений в максимально короткие сроки. В работе предлагается подход к мониторингу корректности функционирования процессов, основанный на методе контроля размерностей переменных процесса. Данный метод позволяет определять нарушения производимых процессом вычислений непосредственно после исполнения некорректной вычислительной команды. В основе предлагаемого метода лежит моделирование линейных участков программного кода с помощью размерностей переменных, используемых для организации вычислений. Определение нарушений корректности вычислений происходит во время мониторинга потока управления исполняемого процесса. В результате данный метод является способом контроля исполнения базовых блоков программного кода процесса. Предложенный метод позволяет распознавать аномальное функционирование процессов, используя модель размерностей переменных. Обнаружение нарушений функционирования предложенным методом может происходить независимо от причин нарушения корректности исполнения процессом вычислительных инструкций. При этом данный метод не позволяет определить причины нарушения вычислений. В работе рассмотрены подходы к реализации данного метода с помощью возможностей операционных систем и с использованием аппаратного обеспечения. Выявлено ограничение предлагаемого метода, заключающееся в требовании к возможности мониторинга исполняемых процессом вычислительных инструкций.

Ключевые слова: операционные системы, исправность процессов, надежность процессов, достоверность процессов, граф потока управления

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Абрамов В. Г., Пучкин Д. А. Метод мониторинга корректности исполнения процессов в операционных системах на основе контроля систем размерностей переменных // Современные информационные технологии и ИТ-образование. 2024. Т. 20, № 2. С. 436-444. https://doi.org/10.25559/SITITO.020.202402.436-444

Original article

Method of Monitoring Correctness of Execution of Processes in Operating Systems by Control of Dimensional Systems of Variables

V. G. Abramov, D. A. Puchkin*
Lomonosov Moscow State University, Moscow, Russian Federation
Address: 1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation
* danila.puchkin@kaspersky.com

Abstract
A problem of monitoring the correctness of calculations faces developers of software products for any operating system. In this case, both determine the moment of incorrect calculation and localize the process commands are led to incorrect calculations is needed. Approaches based on a control timer, process duplication, or endpoint detection and response methods are either redundant or have no guarantees for the localization of computational disruptions in the shortest possible time. The paper proposes an approach to monitoring the correctness of process execution. The approach is based on the method of program variables dimensions control. This method allows you to detect disruptions of the calculations performing by the process after execution of an incorrect computational command. The method is based on modeling linear blocks of program by dimensions of variables using in calculations. Detection of calculations' disruptions is implemented while monitoring the control flow of the executing process. As a result, this method is a way to control the execution of basic blocks of the program. The method makes it possible to recognize the anomalies of processes execution using the variables dimensional model. Detection of disruptions by the method can be executed regardless of reasons of a process execution disruption. However, this method does not allow you to determine the causes of disruptions. Approaches to implementing this method using the capabilities of operating systems and using hardware are considered. The limitation of the method has been revealed, which comes from the requirement for the ability to monitor the executed by process computational instructions.

Keywords: operating systems, flawless process state, process reliability, control flow graph

Conflict of interests: The authors declares no conflict of interest.

For citation: Abramov V.G., Puchkin D.A. Method of Monitoring Correctness of Execution of Processes in Operating Systems by Control of Dimensional Systems of Variables. *Modern Information Technologies and IT-Education.* 2024;20(2):436-444. https://doi.org/10.25559/SITITO.020.202402.436-444

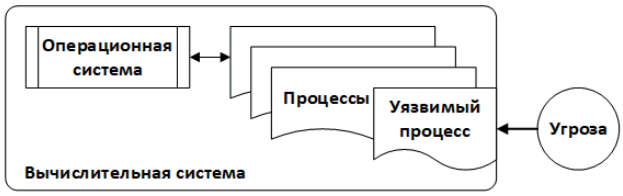
Введение

Исправность – это состояние объекта, при котором его характеристики функционирования соответствуют предъявленным ему требованиям[1]. Одним из критериев исправности является надежность (*достоверность*[2], *reliability*[3]) – свойство соответствия объекта предусмотренному поведению и результатам. Таким образом, в контексте операционных систем исправность и надежность исполнения процесса подразумевают соответствие его поведения и результатов предусмотренному ему функционированию. Предусмотренное процессу функционирование соответствует тому, как это было заложено при проектировании и реализации соответствующих программ.

В настоящее время, как правило, работа программных продуктов под управлением операционных систем подразумевает взаимодействие некоторых своих составляющих с внешними источниками информации (рис. 1). В качестве соответствующих источников угроз могут выступать сетевые ресурсы или датчики физического оборудования[4] [1, 2]. Такое взаимодействие несет в себе риски осуществления вредоносного воздействия на систему [3].

Вредоносное воздействие может приводить к различным последствиям для программного обеспечения. Таким последствием, в частности, может быть нарушение корректности исполнения процессов операционной системы, управляющей вычислительной системой. Чтобы обеспечить возможность обнаружения таких нарушений, при разработке программного обеспечения возникает задача контроля корректности исполнения процессов [4, 5].

Одним из способов вредоносного воздействия на процесс является модификация программного кода процесса путем его изменения или внедрения в него другого программного кода[5]. Такие модификации кода процесса могут приводить к подмене входящих в код команд или их операндов (рис. 2).

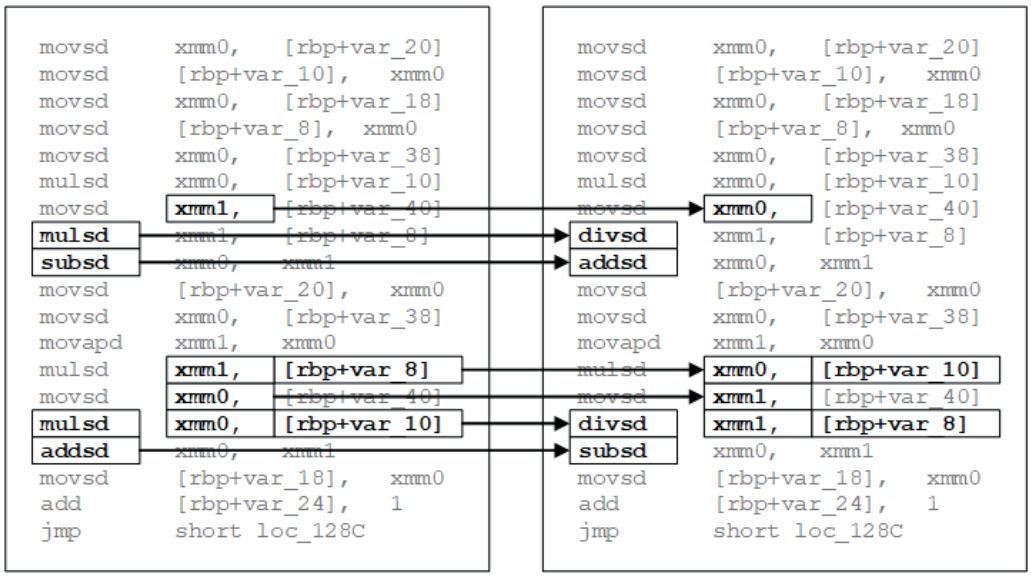


Р и с. 1. Модель воздействия угроз на вычислительную систему

F i g. 1. Model of the threats impact on a computing system

Источник: здесь и далее в статье все рисунки составлены авторами.

Source: Hereinafter in this article all figures were drawn up by the authors.



Р и с. 2. Исходные и модифицированные вычислительные команды

F i g. 2. Original and modified computational commands

[1] ГОСТ 27.002-2015 Надежность в технике. Термины и определения: национальный стандарт Российской Федерации: издание официальное: утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 21 июня 2016 г. № 654-ст: введен впервые: дата введения 2017-03-01. М. : Стандартинформ, 2016. 23 с.

[2] ГОСТ Р ИСО/МЭК 27000-2021 Информационные технологии. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общий обзор и терминология: издание официальное: утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 19 мая 2021 г. № 392-ст: введен впервые: дата введения 2021-11-30. М. : Стандартинформ, 2021. 28 с.

[3] ISO/IEC 27000:2018 Information technology – Security techniques – Information security management systems – Overview and vocabulary. 5th ed. ISO, 2018. URL: https://www.iso.org/standard/73906.html (дата обращения: 29.01.2024).

[4] Борисов М. А., Заводцев И. В., Чижов И. В. Основы программно-аппаратной защиты информации : учебное пособие. Изд. 2-е. М.: Либроком, 2012. 370 с. EDN: QSSLKB

[5] Там же.

В результате такого воздействия на процесс может подвергнуться угрозе один из аспектов достоверности его исполнения – выполнение ожидаемого от него вычислительного алгоритма. Таким образом, решение задачи мониторинга корректности вычислений является частью решения проблемы контроля корректности исполнения процессов. Одним из способов решения данной задачи является проведение мониторинга потока управления процессов в объеме вычислительных инструкций.

Контроль потока управления

Для оперативного ограничения распространения вредоносного воздействия по программному продукту требуется как можно быстрее определить и локализовать его последствия (в рассматриваемом случае – нарушение корректности вычислений). В связи с этим в рамках поставленной задачи использование классических подходов, основанных на сравнении программного кода или соответствующих ему контрольных сумм[6], имеет ряд недостатков. Основным недостатком здесь является тот факт, что периодическое применение таких подходов несет в себе дихотомию между скоростью обнаружения угрозы и потреблением ресурсов вычислительной системы. Иными словами, чем точнее происходит локализация нарушения по времени, тем больше ресурсов системы данный подход будет использовать [6, 7].

Альтернативой сравнению кода выступает контроль потока управления процессов. Моделирование программы для такого контроля проводится с помощью графа потока управления *(Control Flow Graph, CFG)* [8]. Вершины и дуги такого графа соответствуют линейным участкам программы и связям по управлению между ними[7].

Одним из аспектов контроля потока управления является контроль линейных участков программного кода. Прямой подход с контролем соответствия исполняемых команд исходному бинарному коду является ресурсоемким и требует доступа к исполняемому коду. Для обхода данных проблем возможно представление линейных участков программы математическими моделями. Одним из способов моделирования линейных участков кода является их представление уравнениями размерностей переменных[8].

Методы на основе контроля размерностей переменных уже предлагались ранее для статической и динамической верификации корректности программ [9]. В работе[9] предлагается использование размерностей переменных для статического анализа корректности программ. Работа [10] расширяет данный подход методикой динамического контроля с использованием дополнительного аппаратного обеспечения в соответствии с проведенным статическим анализом. В работе [11] предлагается использование деревьев порождения выражений для формирования уравнений размерностей [12]. Наконец, в работе[10] рассматривается практическое применение контроля размерностей с помощью модификации исходных кодов программ на примере драйверов TCP/IP-стека. Однако данные работы не рассматривают практическое использование размерностей переменных для контроля корректности исполнения произвольных процессов операционных систем [13, 14].

Далее предлагается подход к мониторингу корректности вычислений на основе контроля систем размерностей переменных процессов на линейных участках программного кода. Используя аппарат теории размерностей, метод преобразует исполняемые процессом команды и их операнды в матрицу коэффициентов системы линейных уравнений размерностей. Непосредственно контроль производится аппаратом линейной алгебры [15-19].

Предлагаемый метод позволяет определять нарушения корректности, связанные с производимыми процессом вычислениями, непосредственно после исполнения соответствующего линейного участка кода. Это позволяет точно и эффективно по ресурсам производить локализацию нарушений.

Система размерностей переменных программного кода

Рассмотрим алгоритм возведения комплексного числа в натуральную степень :

Input:

a, b – целая и мнимая части исходного комплексного числа

p – натуральное число

Output:

a_cur, b_cur – целая и мнимая части исходного комплексного числа, равного исходному комплексного числу в степени p

```
1: a_old, b_old = 1, 0
2: for i= 1...p do
3:   a_old = a_cur
4:   b_old = b_cur
5:   a_cur = a*a_old – b*b_old
6:   b_cur = a*b_old + b*a_old
7: end for
8: a, b = a_cur, b_cur
```

А л г о р и т м 1. Алгоритм возведения комплексного числа в натуральную степень

A l g o r i t h m 1. Algorithm for raising a complex number to a natural power

Согласно теории размерностей[11], размерности однородных частей математических уравнений должны быть равными. Рассматривая операторы присваивания в рамках программ как уравнения и ставя в соответствие переменным из этих операторов абстрактные размерности в физическом смысле,

[6] Там же.

[7] Aho A. V., Lam M. S., Sethi R., Ullman J. D. Compilers: Principles, Techniques, and Tools. 2nd ed. Addison-Wesley Longman Publishing Co., Inc., USA, 2006. 1040 p.

[8] Годердзишвили Г. М., Ковалев В. В., Романюк В. А. Метод статического контроля правильности программ. СПб.: 1986. 8 с.

[9] Там же.

[10] Петренко С. А., Ступин Д. Д. Национальная система раннего предупреждения о компьютерном нападении. 2-е изд. СПб: Издательский Дом «Афина», 2018. 448 с. EDN: YMWIMP

[11] Huntley H. E. Dimensional analysis Unknown Binding. Rinehart, 1967. 158 p.

размерности левой и правой частей операторов присваивания также должны быть равными. Тогда каждый оператор присваивания задает одно или несколько уравнений размерностей в зависимости того, сколько однородных слагаемых есть в его правой части. На рис. 3 приведены уравнения размерностей линейного участка алгоритма 1 внутри цикла.

```
1: a_old, b_old = 1, 0
2: for i= 1...power do
3:    a_old = a_cur                 [a_old]^1 = [a_cur]^1
4:    b_old = b_cur                 [b_old]^1 = [b_cur]^1
5:    a_cur = a*a_old - b*b_old     [a_cur]^1 = [a]^1[a_old]^1
                                    [a_cur]^1 = [b]^1[b_old]^1
6:    b_cur = a*b_old + b*a_old     [b_cur]^1 = [a]^1[b_old]^1
                                    [b_cur]^1 = [b]^1[a_old]^1
7: end for
8: a, b = a_cur, b_cur
```

Р и с. 3. Представление операторов присваивания уравнениями размерностей

F i g. 3. Representation of assignment operators by dimensional equations

Приведем полученные уравнения к линейному виду с помощью логарифмирования по одному произвольному основанию:

$$\begin{cases} [a\_old]^1 = [a\_cur]^1 \\ [b\_old]^1 = [b\_cur]^1 \\ [a\_cur]^1 = [a]^1[a\_old]^1 \\ [a\_cur]^1 = [b]^1[b\_old]^1 \\ [b\_cur]^1 = [a]^1[b\_old]^1 \\ [b\_cur]^1 = [b]^1[a\_old]^1 \end{cases} \rightarrow \begin{cases} 1 \cdot \log[a\_old] = 1 \cdot \log[a\_cur] \\ 1 \cdot \log[b\_old] = 1 \cdot \log[b\_cur] \\ 1 \cdot \log[a\_cur] = 1 \cdot \log[a] + 1 \cdot \log[a\_old] \\ 1 \cdot \log[a\_cur] = 1 \cdot \log[b] + 1 \cdot \log[b\_old] \\ 1 \cdot \log[b\_cur] = 1 \cdot \log[a] + 1 \cdot \log[b\_old] \\ 1 \cdot \log[b\_cur] = 1 \cdot \log[b] + 1 \cdot \log[a\_old] \end{cases} \quad (1)$$

После логарифмирования полученные уравнения в совокупности задают систему линейных однородных уравнений относительно неизвестных логарифмов введенных размерностей. Перенесем слагаемые системы уравнений влево:

$$\begin{cases} +1 \cdot \log[a\_old] - 1 \cdot \log[a\_cur] + 0 \cdot \log[b\_old] + 0 \cdot \log[b\_cur] + 0 \cdot \log[a] + 0 \cdot \log[b] = 0 \\ +0 \cdot \log[a\_old] + 0 \cdot \log[a\_cur] + 1 \cdot \log[b\_old] - 1 \cdot \log[b\_cur] + 0 \cdot \log[a] + 0 \cdot \log[b] = 0 \\ -1 \cdot \log[a\_old] + 1 \cdot \log[a\_cur] + 0 \cdot \log[b\_old] + 0 \cdot \log[b\_cur] - 1 \cdot \log[a] + 0 \cdot \log[b] = 0 \\ +0 \cdot \log[a\_old] + 1 \cdot \log[a\_cur] - 1 \cdot \log[b\_old] + 0 \cdot \log[b\_cur] + 0 \cdot \log[a] - 1 \cdot \log[b] = 0 \\ +0 \cdot \log[a\_old] + 0 \cdot \log[a\_cur] - 1 \cdot \log[b\_old] + 1 \cdot \log[b\_cur] - 1 \cdot \log[a] + 0 \cdot \log[b] = 0 \\ -1 \cdot \log[a\_old] + 0 \cdot \log[a\_cur] + 0 \cdot \log[b\_old] + 1 \cdot \log[b\_cur] + 0 \cdot \log[a] - 1 \cdot \log[b] = 0 \end{cases} \quad (2)$$

Решение системы (2) определяет зависимость между логарифмами размерностей переменных. Представим систему в виде матрицы коэффициентов:

a_old	*a_cur*	*b_old*	*b_cur*	*a*	*b*
1	−1	0	0	0	0
0	0	1	−1	0	0
−1	1	0	0	−1	0
0	1	−1	0	0	−1
0	0	−1	1	−1	0
−1	0	0	1	0	−1

(3)

Приведем матрицу (3) к верхнетреугольному виду, используя метод Гаусса:

a_old	*a_cur*	*b_old*	*b_cur*	*a*	*b*		*a_old*	*a_cur*	*b_old*	*b_cur*	*a*	*b*
1	−1	0	0	0	0		1	−1	0	0	0	0
0	0	1	−1	0	0		0	1	−1	0	0	0
−1	1	0	0	−1	0	→	0	0	1	−1	0	0
0	1	−1	0	0	−1		0	0	0	0	1	0
0	0	−1	1	−1	0		0	0	0	0	0	1
−1	0	0	1	0	−1		0	0	0	0	0	0

(4)

Из решения системы (4) непосредственно следует система размерностей рассмотренного линейного участка алгоритма – размерности переменных *a_old*, *a_cur*, *b_old* и *b_cur* равны, а переменные *a* и *b* являются безразмерными в силу равенства соответствующих логарифмов нулю.

Таким образом, с помощью аппарата теории размерностей возможно проводить моделирование линейных участков программ как матриц коэффициентов системы линейных уравнений. Результатом такого моделирования является система размерностей переменных соответствующего линейного участка программного кода.

Нарушение системы размерностей переменных

Рассмотрим пример нарушения описанной системы размерностей переменных. Пусть при исполнении алгоритма 1 в строке 5 произошла подмена команды вычитания на команду деления. На рис. 4 приведены модифицированный фрагмент кода и соответствующие ему уравнения размерности. Заметим, что строке 5 соответствует новое уравнение размерностей.

```
1: a_old, b_old = 1, 0
2: for i= 1...power do
3:    a_old = a_cur                 [a_old]^1 = [a_cur]^1
4:    b_old = b_cur                 [b_old]^1 = [b_cur]^1
5:    a_cur = a*a_old / b*b_old     [a_cur]^1 = [a]^1[a_old]^1[b]^-1[b_old]^1
6:    b_cur = a*b_old + b*a_old     [b_cur]^1 = [a]^1[b_old]^1
                                    [b_cur]^1 = [b]^1[a_old]^1
7: end for
8: a, b = a_cur, b_cur
```

Р и с. 4. Представление модифицированного алгоритма уравнениями размерностей

F i g. 4. Representation of the modified algorithm by dimensional equations

Соответствующая полученным уравнениям матрица коэффициентов и ее приведенный вид выглядят следующим образом:

a_old	*a_cur*	*b_old*	*b_cur*	*a*	*b*		*a_old*	*a_cur*	*b_old*	*b_cur*	*a*	*b*
1	−1	0	0	0	0		1	0	0	0	0	0
0	0	1	−1	0	0		0	1	0	0	0	0
−1	1	−1	0	−1	1	→	0	0	1	0	0	−1
0	0	−1	1	−1	0		0	0	0	1	0	−1
−1	0	0	1	0	−1		0	0	0	0	1	0

Решение системы (5) отличается от решения системы : теперь безразмерными являются переменные *a_old*, *a_cur* и *a*, а переменные *b_old*, *b_cur* и *b* имеют одинаковые размерности. В результате подмена вычислительной команды вызвала изменение системы размерностей данного фрагмента алгоритма. Таким образом, изменение системы размерностей линейного участка кода свидетельствует о нарушении выполнения ожидаемого алгоритма программы. В силу этого контроль систем размерностей линейных участков программного кода процессов, исполняемых вычислительной системой, позволяет проводить мониторинг корректности их исполнения.

Мониторинг систем размерностей переменных

Предлагаемый метод контроля корректности вычислений основан на мониторинге систем размерностей переменных и является дополнением к методам мониторинга потока управления. Для мониторинга потока управления предварительно строится граф потока управления программы на основе ста-

тического анализа исходного или исполняемого кода программы, а также на основе профилирования исполнения программы [20, 21]. Для контроля систем размерностей процедура получения графа должна быть дополнена анализом базовых блоков графа с целью их представления в виде систем размерностей переменных. Данный анализ должен производиться в соответствии с методикой, описанной в разделе 2. В результате вершинам модифицированного графа потока управления должны быть поставлены в соответствие системы размерностей переменных соответствующих базовых блоков графа [22].

Для контроля систем размерностей алгоритм мониторинга потока управления должен быть модифицирован с целью сверки эталонных систем размерностей переменных базовых блоков с фактическими. Данный алгоритм помимо контроля пути исполнения программы должен проводить сравнение эталонной и фактической систем размерностей переменных после исполнения соответствующего базового блока программы. При несовпадении систем размерностей для какого-либо базового блока алгоритм принимает решение о нарушении исправности процесса, и работа мониторинга прекращается [23]. Предложенная методика контроля позволяет определять нарушения корректности вычислений и, как следствие, корректности исполнения процесса непосредственно после исполнения очередного линейного участка программного кода. В качестве условия нарушения исправности процесса предлагается использовать следующее: эталонная система размерностей какого-либо линейного участка программного кода после его исполнения не совпадает с фактической.

Таким образом, предлагаемый подход позволяет обнаруживать нарушения корректности вычислений, связанные с такой подменой команд, которая нарушает систему размерностей переменных программы. Следует заметить, что метод не принимает в расчет значения переменных, а оперирует только командами процесса по манипуляции памятью и их операндами [24].

Из специфики метода следуют его ограничения. Во-первых, метод не в состоянии обнаруживать нарушения, связанные с подменой команд, сохраняющей систему размерностей – например, добавление непротиворечивых слагаемых, или же подмена команды сложения на команду вычитание. Во-вторых, метод не контролирует содержимое переменных, в силу чего его подмена может быть отслежена. Это может привести к различным последствиям, начиная с получения некорректного результата вычислений и заканчивая исполнением неверного количества шагов цикла.

Подходы к внедрению мониторинга систем размерностей

Для мониторинга потока управления процессов в вычислительной системе должен быть предусмотрен соответствующий монитор. Такой монитор должен реализовывать модифицированный алгоритм мониторинга потока управления, описанный в разделе 3. Задачей монитора будет являться контроль корректности исполнения процессов [25].

Монитор контроля систем размерностей может быть реализован как программно, так и аппаратно. В случае программного внедрения монитор должен выступать в качестве отдельного процесса операционной системы. При аппаратном подходе к внедрению в качестве монитора должно выступать устройство, реализующее описанные алгоритмы мониторинга.

Для осуществления контроля систем размерностей монитор должен иметь доступ к информации о командах, исполняемых процессами операционной системы. Получение данной информации является основной проблемой для практического использования предложенного подхода. В качестве источников могут выступать непосредственно контролируемые процессы, операционная система или же аппаратные средства вычислительной системы.

Для получения доступа к исполняемым командам непосредственно от контролируемого процесса требуется использование дополнительных команд оповещения в программном кода процесса. Это является причиной возникновения проблемы достоверности получаемой информации. Действительно, ведь если в контролируемом процессе будет совершена подмена вычислительной команды без подмены соответствующих сигнализирующих команд, монитор не получит истинной информации о функционировании процесса. Данная проблема требует придания контролируемым и сигнализирующим командам характера достоверности и атомарности.

Решить эту проблему представляется возможным с помощью механизма системных вызовов операционной системы. Такой механизм должен предоставлять необходимые вычислительные команды, вместе с исполнением которых будет происходить оповещение монитора о них. Очевидным недостатком этого подхода является падение производительности вплоть до уровня, когда подход становится неприменимым. Помимо этого, вопрос достоверности информации может не решаться в операционных системах. Выходом из этого может стать использование аппаратного обеспечения вычислительной системы.

Возможность получения информации о выполненных командах непосредственно от процессора позволяет решить как проблему достоверности информации, так и проблему производительности. Однако такой подход требует наличия аппаратных возможностей наблюдения монитором за командами, которые выполняются процессором. Помимо этого, для однозначного определения соответствия команды и процесса монитору может требоваться дополнительная информация о контексте команды. Описанные возможности требуют специализированного аппаратного обеспечения.

Исходя из вышеперечисленных особенностей внедрения предлагаемого метода, представляется разумным реализовать его с использованием аппаратного обеспечения. Это позволит получить способ контроля вычислений, который позволит обеспечивать эффективный мониторинг при использовании лишь небольшого количества ресурсов контролируемого программного обеспечения. Однако в силу отсутствия требуемого аппаратного обеспечения, на сегодняшний день данный метод возможно использовать с помощью проведения в процессах вычислений посредством специальных системных вызовов управляющей операционной системы в тех случаях, когда производительность программного обеспечения не является его ключевым приоритетом.

Заключение

В работе описан подход к контролю корректности вычислений на линейных участках программного кода процессов на основе мониторинга систем размерностей переменных. На основе подхода предложен метод контроля корректности исполнения процессов посредством мониторинга потока управления с контролем систем размерностей переменных линейных участков программы. Рассмотрены подходы к внедрению предложенного метода и проведен анализ их применимости. В результате анализа определено, что для применимости предложенных подходов в вычислительной системе она должна соответствовать требованиям по мониторингу вычислительных инструкций исполняемых процессов.

Список использованных источников

[1] Мирзабаев А. Н., Самонов А. В. Метод обеспечения устойчивости вычислительного процесса в условиях воздействия вредоносных программ // Вопросы кибербезопасности. 2022. № 2(48). С. 63-71. https://doi.org/10.21681/2311-3456-2022-2-63-71

[2] Петренко С. А., Костюков А. Д. Методика гибридного мониторинга угроз безопасности // Защита информации. Инсайд. 2020. № 2(92). С. 4-16. EDN: UUKTLC

[3] Буханов Д. Г., Сулохин Д. В. Выявление вредоносного программного обеспечения на основе классификации графов потоков исходных кодов // Доклады Томского государственного университета систем управления и радиоэлектроники. 2018. Т. 21, № 3. С. 30-34. https://doi.org/10.21293/1818-0442-2018-21-3-30-34

[4] Скворцов А. А. Проектирование и реализация многозадачных встроенных систем управления на микроконтроллерах: операционная система или автоматы? // Программная инженерия. 2019. Т. 10, № 7-8. С. 311-316. https://doi.org/10.17587/prin.10.311-316

[5] Лебедянцев В. В. Обобщенный инвариантный метод передачи сообщений и оценка его информационной защищенности // Инфокоммуникационные технологии. 2014. Т. 12, № 3. С. 28-32. EDN: THAWHT

[6] Bonfante G., Kaczmarek M., Marion J. Y. Architecture of a morphological malware detector // Journal in Computer Virology. 2009. Vol. 5. P. 263-270. https://doi.org/10.1007/s11416-008-0102-4

[7] Enforcing C/C++ Type and Scope at Runtime for Control-Flow and Data-Flow Integrity / M. Ismail [et al.] // Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24), Vol. 3. New York, NY, USA: Association for Computing Machinery, 2024. P. 283-300. https://doi.org/10.1145/3620666.3651342

[8] Shahini S., Zhang M., Payer M., Ricci R. Arvin: Greybox Fuzzing Using Approximate Dynamic CFG Analysis // Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security (ASIA CCS '23). New York, NY, USA: Association for Computing Machinery, 2023. P. 232-246. https://doi.org/10.1145/3579856.3582813

[9] Кулямин В. В. Перспективы интеграции методов верификации программного обеспечения // Труды Института системного программирования РАН. 2009. Т. 16. С. 73-88. EDN: MWEEWP

[10] Ковалев В. В., Компаниец Р. И., Новиков В. А. Верификация программ на основе соотношений подобия // Труды СПИИРАН. 2015. № 1(38). С. 233-245. EDN: TQURCF

[11] Харжевская А. В., Ломако А. Г., Петренко С. А. Представление программ инвариантами подобия для контроля искажения вычислений // Вопросы кибербезопасности. 2017. № 2(20). С. 9-20. https://doi.org/10.21581/2311-3456-2017-2-9-20

[12] Харжевская А. В., Ломако А. Г., Петренко С. А. Аудит безопасности на основе многослойных инвариантов подобия // Защита информации. Инсайд. 2018. № 2(80). С. 22-33. EDN: YVTAVP

[13] The duality of memory and communication in the implementation of a multiprocessor operating system / M. Young [et al.] // ACM SIGOPS Operating Systems Review. 1987. Vol. 21, No. 5. P. 63-76. https://doi.org/10.1145/37499.37507

[14] Adonis: Practical and Efficient Control Flow Recovery through OS-level Traces / X. Liu [et al.] // ACM Transactions on Software Engineering and Methodology. 2024. Vol. 33, issue 1. Article number: 2. https://doi.org/10.1145/3607187

[15] Exploiting the Sparseness of Control-Flow and Call Graphs for Efficient and On-Demand Algebraic Program Analysis / G. K. Conrado [et al.] // Proceedings of the ACM on Programming Languages. 2023. Vol. 7, No. OOPSLA2. Article number: 292. https://doi.org/10.1145/3622868

[16] Ramsey N. Beyond Relooper: recursive translation of unstructured control flow to structured control flow (functional pearl) // Proceedings of the ACM on Programming Languages. 2022. Vol. 6, No. ICFP. Article number: 90. https://doi.org/10.1145/3547621

[17] A Procrastinating Control-Flow Integrity Framework for Periodic Real-Time Systems / T. Mishra [et al.] // Proceedings of the 31st International Conference on Real-Time Networks and Systems (RTNS '23). New York, NY, USA: Association for Computing Machinery, 2023. P. 132-142. https://doi.org/10.1145/3575757.3575762

[18] Mishra T., Chantem T., Gerdes R. Survey of Control-flow Integrity Techniques for Real-time Embedded Systems // ACM Transactions on Embedded Computing Systems. 2022. Vol. 21, No. 4. Article number: 41. https://doi.org/10.1145/3538275

[19] Abadi M., Budiu M., Erlingsson U., Ligatti J. Control-flow integrity principles, implementations, and applications // ACM Transactions on Information and System Security. 2009. Vol. 13, No. 1. Article number: 4. https://doi.org/10.1145/1609956.1609960

[20] Wagner D., Dean R. Intrusion detection via static analysis // Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, Oakland, CA, USA: IEEE Press, 2001. P. 156-168. https://doi.org/10.1109/SECPRI.2001.924296

[21] Koppel J., Kearl J., Solar-Lezama A. Automatically deriving control-flow graph generators from operational semantics // Proceedings of the ACM on Programming Languages. 2022. Vol. 6, No. ICFP. Article number: 117. https://doi.org/10.1145/3547648
[22] Allen F. E. Control flow analysis // ACM SIGPLAN Notices. 1970. Vol. 5, issue 7. P. 1-19. https://doi.org/10.1145/390013.808479
[23] Bauman E., Duan J., Hamlen K. W., Lin Z. Renewable Just-In-Time Control-Flow Integrity // Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23). New York, NY, USA: Association for Computing Machinery, 2023. P. 580-594. https://doi.org/10.1145/3607199.3607239
[24] Ritika Wason, Soni A. K., Qasim Rafiq M. Estimating Software Reliability by Monitoring Software Execution through OpCode // International Journal of Information Technology and Computer Science. 2015. Vol. 7, issue 9. P. 23-30. https://doi.org/10.5815/ijitcs.2015.09.04
[25] Hohmuth M., Peter M., Härtig H., Shapiro J. S. Reducing TCB size by using untrusted components: small kernels versus virtual-machine monitors // Proceedings of the 11th workshop on ACM SIGOPS European workshop (EW 11). New York, NY, USA: Association for Computing Machinery, 2004. P. 22-es. https://doi.org/10.1145/1133572.1133615

Поступила 29.01.2024; одобрена после рецензирования 12.03.2024; принята к публикации 20.05.2024.

Об авторах:

Абрамов Владимир Геннадьевич, доцент кафедры алгоритмических языков факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), кандидат физико-математических наук, доцент, **ORCID: https://orcid.org/0000-0002-6818-9378,** vlabr@cs.msu.ru

Пучкин Данила Андреевич, аспирант кафедры алгоритмических языков факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0000-0002-4159-1252,** danila.puchkin@kaspersky.com

Все авторы прочитали и одобрили окончательный вариант рукописи.

References

[1] Mirzabaev A.N., Samonov A.V. Control method of the correct execution of programs by monitoring and analyzing the route-time parameters of the computing process. *Voprosy Kiberbezopasnosti*. 2022;(2):63-71. (In Russ., abstract in Eng.) https://doi.org/10.21681/2311-3456-2022-2-63-71
[2] Petrenko S.A., Kostyukov A.D. Hybrid Security Threat Monitoring. *Zaŝita informacii. Inside*. 2020;(2):4-16. (In Russ., abstract in Eng.) EDN: UUKTLC
[3] Bukhanov D.G., Sulokhin D.V. Detection of malware based on the classification of source code graphs. *Proceedings of the TUSUR University*. 2018;21(3):30-34. (In Russ., abstract in Eng.) https://doi.org/10.21293/1818-0442-2018-21-3-30-34
[4] Skvortsov A.A. Design and implementation of multitasking embedded control systems on microcontrollers: operating system or state machines? *Programmnaya Ingeneria* = Software Engineering. 2019;10(7-8);311-316. (In Russ., abstract in Eng.) https://doi.org/10.17587/prin.10.311-316
[5] Lebedyantsev V.V. Generalized invariant method messaging and assessment of its information security. *Infocommunicationnye Technologii*. 2014;12(3):28-32. (In Russ., abstract in Eng.) EDN: THAWHT
[6] Bonfante G., Kaczmarek M., Marion J.Y. Architecture of a morphological malware detector. *Journal in Computer Virology*. 2009;5:263-270. https://doi.org/10.1007/s11416-008-0102-4
[7] Ismail M., et al. Enforcing C/C++ Type and Scope at Runtime for Control-Flow and Data-Flow Integrity. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24), Vol. 3. New York, NY, USA: Association for Computing Machinery; 2024. p. 283-300. https://doi.org/10.1145/3620666.3651342
[8] Shahini S., Zhang M., Payer M., Ricci R. Arvin: Greybox Fuzzing Using Approximate Dynamic CFG Analysis. In: Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security (ASIA CCS '23). New York, NY, USA: Association for Computing Machinery; 2023. p. 232-246. https://doi.org/10.1145/3579856.3582813
[9] Kulyamin V.V. Prospects for the integration of software verification methods. *Proceedings of the Institute for System Programming*. 2009;16(1):73-88. (In Russ., abstract in Eng.) EDN: MWEEWP
[10] Kovalev V.V., Kompanietc R.I., Novikov V.A. Verification of Programs Based on Similarity. *Informatics and Automation (SPIIRAS Proceedings)*. 2015;(1):233-245. (In Russ., abstract in Eng.) EDN: TQURCF
[11] Kharzhevskaya A.V., Lomako A.G., Petrenko S.A. Representing Programs with Similarity Invariants for Monitoring Tampering with Calculations. *Voprosy Kiberbezopasnosti*. 2017;(2):9-20. (In Russ., abstract in Eng.) https://doi.org/10.21581/2311-3456-2017-2-9-20
[12] Kharzhevskaya A.V., Lomako A.G., Petrenko S.A. Audit of Safety Based on Multilateral Invariants of Similarity. *Zaŝita informacii. Inside*. 2018;(2):22-33. (In Russ., abstract in Eng.) EDN: YVTAVP

[13] Young M., Tevanian A., Rashid R., Golub D., Eppinger J. The duality of memory and communication in the implementation of a multiprocessor operating system. *ACM SIGOPS Operating Systems Review*. 1987;21(5):63-76. https://doi.org/10.1145/37499.37507
[14] Liu X., et al. Adonis: Practical and Efficient Control Flow Recovery through OS-level Traces. *ACM Transactions on Software Engineering and Methodology.* 2024;33(1):2. https://doi.org/10.1145/3607187
[15] Conrado G.K., et al. Exploiting the Sparseness of Control-Flow and Call Graphs for Efficient and On-Demand Algebraic Program Analysis. *Proceedings of the ACM on Programming Languages*. 2023;7(OOPSLA2):292. https://doi.org/10.1145/3622868
[16] Ramsey N. Beyond Relooper: recursive translation of unstructured control flow to structured control flow (functional pearl). *Proceedings of the ACM on Programming Languages*. 2022;6(ICFP):90. https://doi.org/10.1145/3547621
[17] Mishra T., et al. A Procrastinating Control-Flow Integrity Framework for Periodic Real-Time Systems. In: Proceedings of the 31st International Conference on Real-Time Networks and Systems (RTNS '23). New York, NY, USA: Association for Computing Machinery; 2023. p. 132-142. https://doi.org/10.1145/3575757.3575762
[18] Mishra T., Chantem T., Gerdes R. Survey of Control-flow Integrity Techniques for Real-time Embedded Systems. *ACM Transactions on Embedded Computing Systems*. 2022;21(4):41. https://doi.org/10.1145/3538275
[19] Abadi M., Budiu M., Erlingsson U., Ligatti J. Control-flow integrity principles, implementations, and applications. *ACM Transactions on Information and System Security*. 2009;13(1):4. https://doi.org/10.1145/1609956.1609960
[20] Wagner D., Dean R. Intrusion detection via static analysis. In: Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, Oakland, CA, USA: IEEE Press; 2001. p. 156-168. https://doi.org/10.1109/SECPRI.2001.924296
[21] Koppel J., Kearl J., Solar-Lezama A. Automatically deriving control-flow graph generators from operational semantics. *Proceedings of the ACM on Programming Languages*. 2022;6(ICFP):117. https://doi.org/10.1145/3547648
[22] Allen F.E. Control flow analysis. *ACM SIGPLAN Notices*. 1970;5(7):1-19. https://doi.org/10.1145/390013.808479
[23] Bauman E., Duan J., Hamlen K.W., Lin Z. Renewable Just-In-Time Control-Flow Integrity. In: Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23). New York, NY, USA: Association for Computing Machinery; 2023. p. 580-594. https://doi.org/10.1145/3607199.3607239
[24] Ritika Wason, Soni A.K., Qasim Rafiq M. Estimating Software Reliability by Monitoring Software Execution through OpCode. *International Journal of Information Technology and Computer Science*. 2015;7(9):23-30. https://doi.org/10.5815/ijitcs.2015.09.04
[25] Hohmuth M., Peter M., Härtig H., Shapiro J.S. Reducing TCB size by using untrusted components: small kernels versus virtual-machine monitors. In: Proceedings of the 11th workshop on ACM SIGOPS European workshop (EW 11). New York, NY, USA: Association for Computing Machinery; 2004. p. 22-es. https://doi.org/10.1145/1133572.1133615

Submitted 29.01.2024; approved after reviewing 12.03.2024; accepted for publication 20.05.2024.

About the authors:

Vladimir G. Abramov, Associate Professor of the Department of Algorithmic Languages, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), Cand. Sci. (Phys.-Math.), Associate Professor, **ORCID: https://orcid.org/0000-0002-6818-9378,** vlabr@cs.msu.ru

Danila A. Puchkin, Postgraduate Student of the Department of Algorithmic Languages, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0000-0002-4159-1252,** danila.puchkin@kaspersky.com

All authors have read and approved the final manuscript.