УДК 004.931, 004.942

А.А. ПЛАТОНОВ, В.И. ТИМОФЕЕВ

КОНТРОЛЬ ЦЕЛОСТНОСТИ ДИНАМИЧЕСКИХ ОБЪЕКТОВ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ МЕТРИЧЕСКИХ ЭТАЛОНОВ

Платонов А.А., Тимофеев В.И. **Контроль целостности динамических объектов вычислительных систем с использованием метрических эталонов.**
Аннотация. В статье предлагается подход к контролю целостности динамических объектов по их метрическим эталонам. Создание эталона основывается на последовательном преобразовании процесса от дампа памяти до автомата переходов на графе состояний с расчетом структурных, информационных и операционных метрик. Это позволяет выявлять нарушения функциональных состояний объекта в памяти вычислительной системы. Представлен алгоритм контроля целостности динамических объектов антивирусного средства Dr.Web.
Ключевые слова: целостность, метрика программного обеспечения, формальная грамматика, продукционная система, конечный автомат, распознаватель.

Platonov A.A., Timofeev V.I. **Monitoring of Integrity of Dynamic Objects of Computing Systems with Use of Metric Standards.**
Abstract. In article approach to monitoring of integrity of dynamic objects on their metric standards is offered. Creation of a standard is based on sequential conversion of process from a memory dump to the machine gun of transitions on a state graph with calculation of structural, information and operational metrics. It allows to reveal violations of the functional statuses of object in memory of the computing system. The algorithm of monitoring of integrity of dynamic objects of anti-virus means of Dr.Web is provided.
Keywords: integrity, software metrics, the formal grammar, productional system, finite state machine, recognizer.

1. Введение. Для обеспечения информационной безопасности автоматизированных систем (АС) и предотвращения возникновения скрытых каналов доступа (СКД) к информационным ресурсам АС необходимо обеспечивать целостность, конфиденциальность и доступность всех информационных объектов в АС.

Среди всех информационных объектов в АС выделяют статические объекты и динамические объекты. К первым относят программные файлы и сетевые пакеты, то есть объекты, которые в течение относительно длительного времени функционирования АС могут оставаться неизменными. Вторую группу составляют вычислительные процессы и потоки – программные образы, постоянно находящиеся в оперативной памяти. Особенностью второй группы является непрерывно меняющиеся характеристики (исполняемый код и данные) информационных объектов в режиме времени близком к

реальному. Данная особенность приносит с собой дополнительный ряд угроз, которым подвержены динамические объекты. Наиболее актуальная, из которых заключается в трудности контроля целостности вычислительных процессов.

В таких обстоятельствах следует учесть, что для контроля целостности статических объектов (файлы, пакеты) АС существует достаточно много надежных способов. Эти способы основаны на использовании контрольных сумм и криптографических методов шифрования. Методы контроля динамически изменяемых объектов практически отсутствуют, если не считать механизм защиты виртуальной памяти, используемый в операционных системах (ОС).

Механизм защиты виртуальной памяти в основном направлен на обеспечение доступности и конфиденциальности динамических объектов. Это снижает угрозу нарушения целостности динамических объектов, но не устраняет ее полностью. При этом необходимо отметить, что самая распространенная в мире ОС компании Microsoft имеет закрытый код. Это обуславливает отсутствие возможности проверить эффективность данного механизма, а также затрудняет сертификацию этой ОС на предмет наличия скрытых функциональных возможностей этого механизма.

Актуальность поставленной задачи обусловлена необходимостью своевременного выявления фактов нарушения целостности динамических объектов в АС, и несовершенстве существующих методов по контролю целостности.

2. Назначение контроля целостности динамических объектов. Контроль целостности динамических объектов предназначен для реализации мер по защите вычислительных процессов от информационных атак, которые направлены на их уничтожение или модификацию.

Основные направления информационных атак, а также цели таких атак представлены на рисунке 1. Атака, проведенная на динамические объекты АС, может с большой степенью скрытности изменить весь функциональный набор такого объекта или нейтрализовать его.

Рис. 1. Направление вредоносного воздействия на информационные объекты

Под динамическими объектами подразумеваются вычислительные процессы и потоки. Контекст вычислительного процесса включает в себя: виртуальное адресное пространство (память), исполняемый код, дескрипторы объектов, данные, приоритет, максимальный и минимальный размер рабочей области, первичный поток и его потомки (дочерние потоки).

Исследование существующих методов контроля целостности динамических объектов показало, что основным направлением в этой области является метод слежения за вызовом системных функций, а реализуется такой метод с помощью эвристик, закладываемых в специальные программы, так называемые системные мониторы (СМ).

Суть таких эвристик заключается в том что, из всех API (Application Programming Interface) функций выбираются наиболее критические, связанные с доступом к памяти, а также возможные их комбинации. Выбор производится, как правило, на основании статистики и экспертных оценок. Если при работе операционной системы будут возникать такие критические комбинации, то системный монитор немедленно прекратит работу приложения, инициирующего такую комбинацию, и выдаст сигнал администратору безопасности для дальнейшего анализа.

Разработка эвристик и дальнейшая их реализация в системных мониторах должна также учитывать наличие rootkit-программ, которые в своем составе почти всегда имеют API-перехватчик и NativeAPI-перехватчик. Этот факт также имеет большое влияние на эффективность работы системных мониторов.

Вычислительные процессы могут в процессе своей работы

использовать критически важные функции без зловредного умысла а, следовательно, реакция системных мониторов на них будет неоднозначна.

Все вышесказанное позволяет сделать вывод о том, что существующие методы контроля целостности не в полной мере соответствуют задаче поддержания неизменности функциональных возможностей динамического объекта [1].

Для решения задачи контроля целостности предлагается:

1. Создание инвариантного эталона динамического объекта независимого от его состояния.
2. Выявление нарушений целостности динамического объекта на основании сравнения текущего состояния объекта с эталоном.

3. Структурированный эталон динамического объекта. Проведенный системный анализ динамических объектов показал, что универсальной моделью, отражающей процесс, является управляющий граф, который вместе с дампом памяти являются исходными данными для решения задачи контроля целостности. Идея многомодельности программ позволяет интерпретировать универсальную модель в виде формальных грамматик и продукционных правил, конкретизированных под решаемую задачу, а также автомата, реализующего эти модели [1]. Особенности представления процессов в виде таких моделей позволит выявить разного рода воздействия на них, и осуществить контроль целостности.

Для создания эталона динамического объекта необходимо выполнить ряд действий:

– выбрать множество состояний динамического объекта, разнесенных по времени и свойствам;

– рассчитать для каждого из множества состояний объекта метрики, и на их основании построить множество метрических моделей;

– обобщить данные из множества метрических моделей одного объекта, и создать его эталонную метрическую модель.

В соответствии с системным подходом к интерпретации вычислений, различают: формальную модель языка структур, задаваемую грамматикой, формализацию логических свойств вычислений, задаваемую продукционными системами и формальную модель реализации системы действий представленную автоматными моделями вычислений [2].

Общая система многомодельного представления вычислительного процесса представлена на рисунке 2.

Основой для разработки трех моделей вычислительного

процесса являются дамп памяти $C =< \sigma, K >$, где σ – состояние памяти, а K – активный оператор, и управляющий граф программы $G(V, E)$, где V – количество вершин (состояний), а E – число дуг (переходов). Анализ графа позволяет создать регулярную грамматику, систему нормальных продукций, а также конечный автомат.

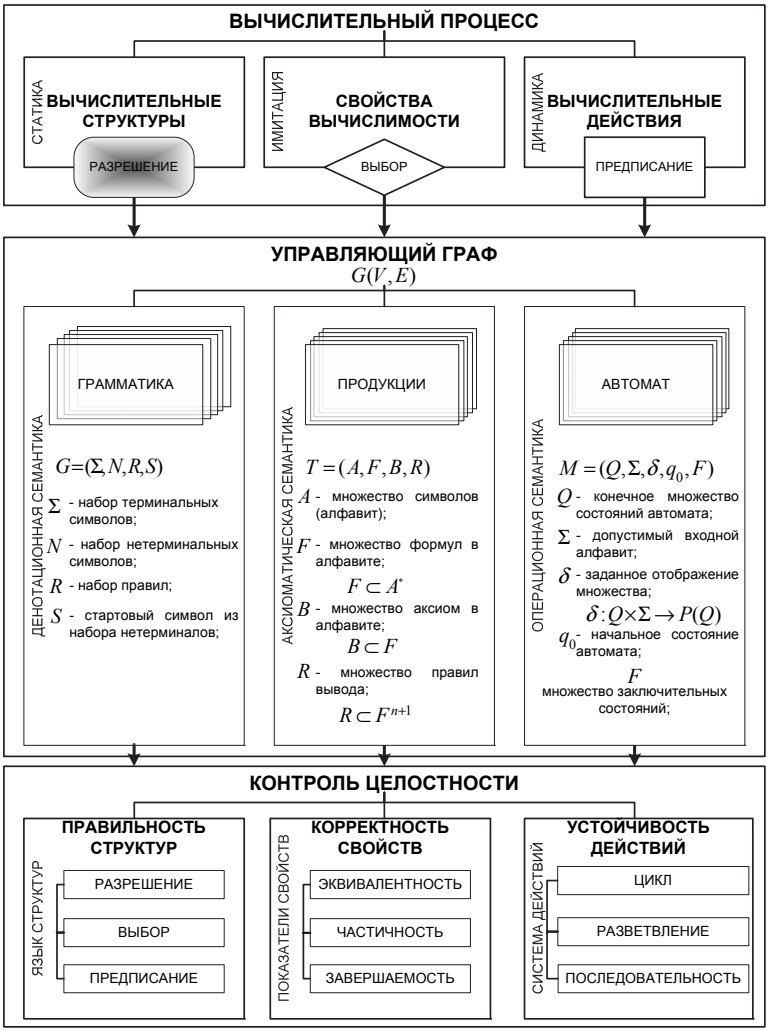


Рис.2. Система многомодельного представления вычислительного процесса

Каждая из предлагаемых моделей, позволяет контролировать один из трех компонентов, выбранного базиса. Под контролем в нашем случае подразумевается слежение за правильностью вычислительных структур, корректностью свойств вычислимости и устойчивостью вычислительных действий. Реализация этого контроля происходит на основании присущих каждой из моделей специфических характеристик.

С точки зрения правильности вычислительных структур, задаваемых в «Разрешениях», различают функциональное («Разрешение разрешений»), логическое («Разрешение условий») и алгебраическое («Разрешение предписаний») программирование. Каждый из этих видов программирования имеет соответствующее понятие правильности исполняемого кода.

Вывод о корректности вычислений можно сделать на основании свойств эквивалентности, частичной правильности и завершаемости.

Общая устойчивость действий вычислительного процесса зависит от выполнения частных участков кода, поэтому разобьем исполняемый код на различные сочетания базовых конструкций языков программирования: «Цикл», «Разветвление» и «Последовательность». Это позволит более тщательно контролировать изменения выполнения инструкций в любой момент времени.

Все вышесказанное, позволяет перейти от общей структурированной модели вычислительного процесса к частной, представленной на рисунке 3. На этом рисунке отображены конструкции основного базиса на множестве численных показателей метрического пространства программ [3].

В аспекте вычислительной структуры и свойств вычислимости программы можно ограничиться рассмотрением её топологических и информационных мер, в основе которых лежат топологические характеристики граф-модели программы, а также структуры и модели данных [4-6].

Именно эти меры удовлетворяют подавляющему большинству требований, предъявляемых к метрическим показателям: общность применимости, адекватность рассматриваемому свойству, существенность оценки, состоятельность, количественное выражение, воспроизводимость измерений, малая трудоёмкость вычислений, возможность автоматизации оценивания.

И, наконец, именно топологические и информационные меры наиболее употребимы в целях построения метрического эталона динамического объекта, чувствующего структурные искажения и нарушения свойств объекта [5].

Для отражения вычислительных действий базовых конструкции управления, которые применяются в языках программирования, был выбран вероятностный аппарат, введенный в работе Карповского Ефима Яковлевича [7].

ТИПЫ		МЕТРИКИ	
СТРУКТУРА	РАЗРЕШЕНИЕ	ТОПОЛОГИЯ	Метрика Маккейба $C = e - n + 2$ Метрика Майерса $[Z(G), Z(G) + h]$ Метрика Вудворда $Y(x)$ Метрика Мак-Клура $M(P) = fp * X(P) + gp * Y(P)$ Функциональная мера $f_1 = \sum c_1$ Метрика Кафура $I = length * (fan\_in * fan\_out)^2$
	ВЫБОР		Метрика Чена $M(G) = (n(G), N, Q_0)$ Метрика точек пересечения $\min(a,b) < \min(p,q) < \max(a,b) \& \max(p,q) > \max(a,b)$ $\min(a,b) < \max(p,q) < \max(a,b) \& \min(p,q) < \min(a,b)$ Метрика Пратта $M(\{F_1, F_2, \ldots F_n\}) = \sum_{i=1}^{n} M(F_i)$ Метрика Чепина $Q = P + 2M + 3C + 0,5T$
	ПРЕДПИСАНИЕ		Метрика Джилба $cl = CL / L, f = N_{cs} / L_{\mathrm{mod}}$ Метрика Пивоварского $N(G) = V^*(G) + \sum P_i$ Метрика регулярных выражений $P(G) = N + L + \sum_{i}^{i} k$ Метрика спена $n, n-1$
СВОЙСТВА	ЭКВИВАЛЕНТНОСТЬ	ИНВАРИАНТ	Информационная энтропия $H(x) = -\sum_{i=1}^{n} p(i) \log_2 p(i)$
	ЧАСТИЧНОСТЬ		Информационное содержание $I = (\frac{\eta_1^*}{\eta_1} * \frac{\eta_2}{N_2}) * (\log_2 \eta * (\eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2))$
	ЗАВЕРШАЕМОСТЬ		Информационная мера Берлингера $I(R) = m(F^*(R) \times F^-(R))^2$
ДЕЙСТВИЯ	ЦИКЛ	ВЕРОЯТНОСТЬ	Время $\tau(\Phi_3) = (1 - P_a)\tau(a) \sum_{i=0}^{U} P_a^i + (1 - P_a)\left[\tau(a) + \tau(g)\right] \sum_{i=0}^{U} i P_a^i$ $\tau(\Phi_2) = P_a\left[\tau(a) + \tau(g) + (1 - P_a)\right]\left[\tau(a) + \tau(h)\right]$ $\tau(\Phi_1) = \tau(g) + \tau(g)$
	РАЗВЕТВЛЕНИЕ		Среднеквадратическое отклонение времени $\sigma_\tau(\Phi_2) = \sqrt{P_a\left[\sigma_\tau^2(a) + \sigma_\tau^2(g)\right] + (1 - P_a)\left[\sigma_\tau^2(a) + \sigma_\tau^2(g)\right]}$ $\sigma_{t\tau}(\Phi_1) = \sqrt{\sigma_\tau^2(g) + \sigma_\tau^2(h)}$ $\sigma_\tau^2(\Phi_3) = (1 - P_a) \sum_{i=0}^{U} P_a^i \left[\sigma_\tau^2(a) + i\sigma_\tau^2(a) + i\sigma_\tau^2(g)\right] + (1 - P_a)$ $\sum_{i=0}^{U} P_a^i \left[(1+i)\tau(a) + i\tau(g)\right]^2 - \left\{\sum_{i=0}^{U} P_a^i (1 - P_a)\left[(1+i)\tau(a) + i\tau(g)\right]^2 - \right\}$
	ПОСЛЕДОВАТЕЛЬНОСТЬ		Вероятность наступления события P_a, P_g $P(A) = K_a / M_a$ $P(G) = K_g / M_g$

Рис. 3. Модель представления вычислительного процесса в базисе {<структура>, <свойства>, <действия>}

Все вышеперечисленные показатели позволяют нам контролировать целостность динамических объектов на основании выдвинутого базиса программ, который является неизменным на всем протяжении работы исполняемого кода. В дальнейшем нам предстоит только конкретизировать некоторые аспекты, предложенной частной структурированной модели динамического объекта (вычислительного процесса), а также подготовить для нее исходные данные.

В целях оптимального выбора метрик, лежащих в основе разработанной модели, предлагается ввести структурированный базис для конструкций языка ассемблер.

Базис конструкций охватывает все возможные типовые структуры, которые могут существовать в исполняемом коде программы. Наличие пересечений стандартных типовых конструкций («цикл-цикл» – «вложение», «разветвление-разветвление» – «ветвление», «последовательность-последовательность» – «составление») наиболее ярко отражают ключевые точки управляющего графа вычислительного процесса.

Простейшая конструкция «разветвление» самая распространенная из всех, а соответственно наиболее ярко отражает структуру процесса.

Анализ всех топологических метрик сложности для структурированного базиса конструкций сводится в метрическое пространство (рисунок 4), составляющее эталон.

Структуры процесса заданы набором показателей:

$$Q_1 = (q_{11}, q_{12}, q_{13}, ..., q_{19}), \quad (1)$$

где q_{1i} – один из метрических показателей, представленных на рисунке 3.

Метрический эталон свойств вычислимости динамических объектов (рисунок 5), содержит информационную энтропию q_{21}, информационное содержание q_{22} и информационную меру Берлингера процесса q_{23}.

$$Q_2 = (q_{21}, q_{22}, q_{23}). \quad (2)$$

Показатели, используемые для создания эталона устойчивости действий, представлены на рисунке 6.

Расчет метрических показателей динамического объекта производится как на этапе эталонирования, так и на этапе контроля целостности.

В качестве модели представления вычислительного процесса

выбрана частная метрическая модель $x_k\{Q_\gamma : \gamma = 1(1)3\}$ основу, которой составляют метрические показатели.

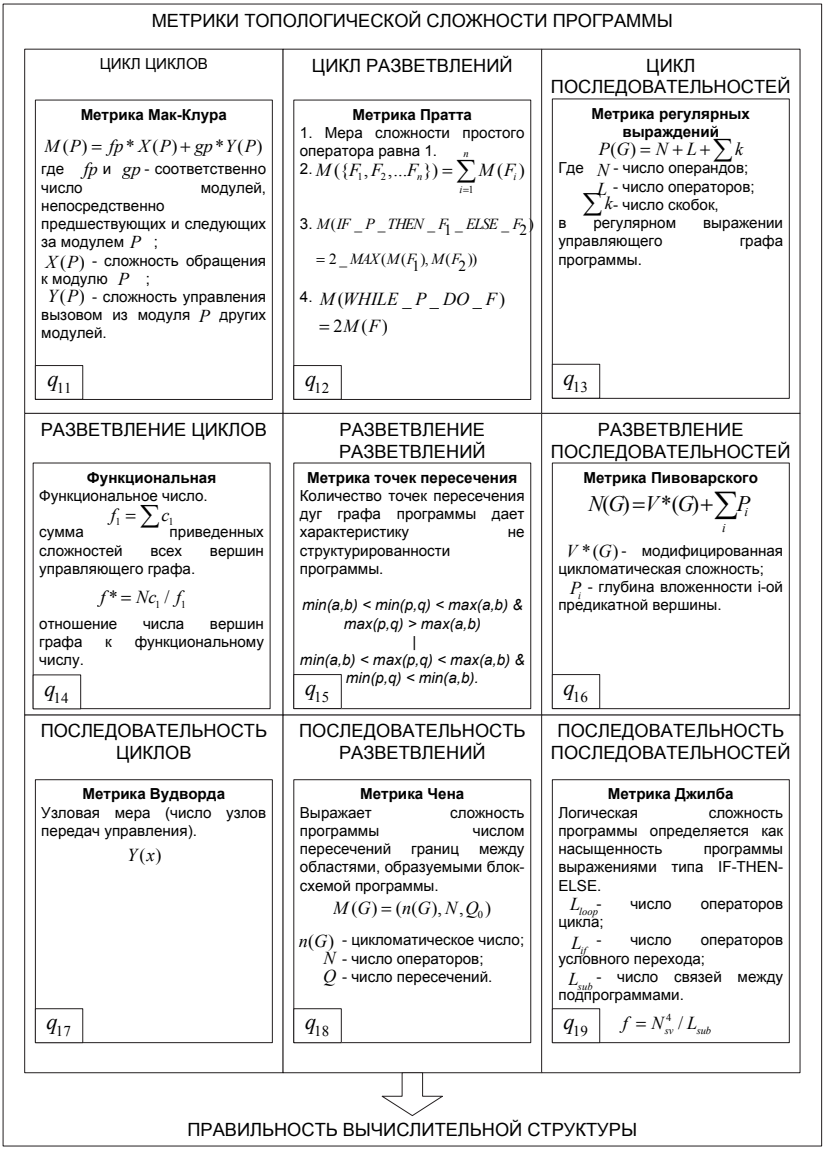


Рис. 4. Метрический эталон структуры вычислительного процесса

Метрические показатели, как уже было сказано выше, разделены на три категории:

1. Метрики структуры вычислительного процесса $Q = (q_{11}, q_{12}, ...q_{19})$.

2. Информационные показатели $Q_2 = (q_{21}, q_{22}, q_{23})$.

3. Критерии устойчивости выполнения вычислительного процесса $Q_3 = (q_{31}, q_{32}, ..., q_{39})$.

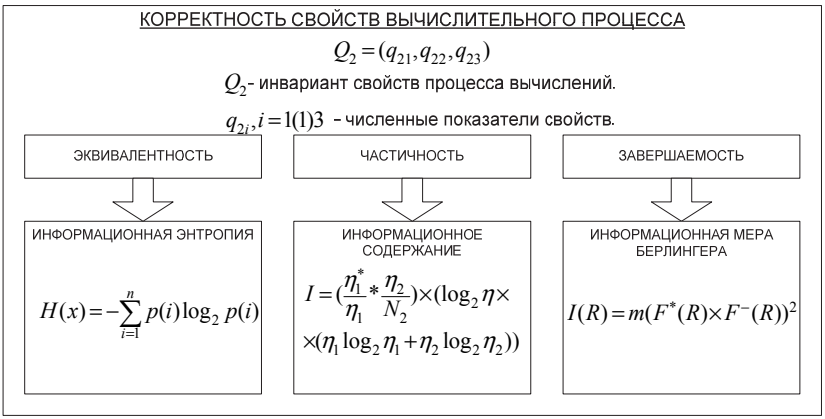
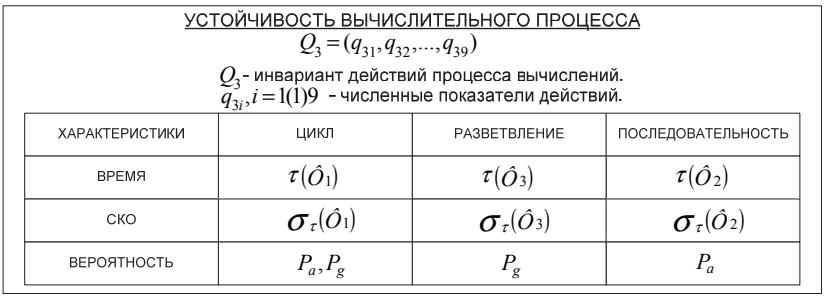


Рис. 5. Метрический эталон свойств вычислимости динамических объектов



ХАРАКТЕРИСТИКИ	ЦИКЛ	РАЗВЕТВЛЕНИЕ	ПОСЛЕДОВАТЕЛЬНОСТЬ
ВРЕМЯ	$\tau(\hat{O}_1)$	$\tau(\hat{O}_3)$	$\tau(\hat{O}_2)$
СКО	$\sigma_\tau(\hat{O}_1)$	$\sigma_\tau(\hat{O}_3)$	$\sigma_\tau(\hat{O}_2)$
ВЕРОЯТНОСТЬ	P_a, P_g	P_g	P_a

Рис. 6. Метрический эталон устойчивости действий динамических объектов

Представленные показатели, отражающие сущность динамического объекта, являются необходимым и достаточным условием для построения частной метрической модели.

После расчета всех метрических показателей на нескольких срезах (образах) эталонируемого динамического объекта можно приступать к создания векторов математического и среднеквадратического отклонения. В результате полученная модель

используется в качестве метрического эталона, который необходим для осуществления контроля целостности динамических объектов.

4. Выявление нарушений целостности динамического объекта. Выявление признаков нарушения целостности связано с задачей распознавания образов. Задача распознавания признаков нарушения целостности, а также существующие исходные данные нечетко определяют метод распознавания. Следовательно, необходимо заимствовать приемы статистических, структурно-лингвистических и кластерных методов. Это обусловлено набором статистических показателей $x_k = \{Q_\gamma : \gamma = 1(1)3\}$, срезов вычислительных процессов $X = (x_1, x_2, ..., x_k, ..., x_n)$ и наличием ситуаций неопределенности по соотнесению процесса вычислений к целостному ω_1 или нецелостному ω_4 классу. Решающим правилом (РП) является выражение 3, а решающей функцией 4.

В качестве ключевых (основных) элементов q_i выступают метрика точек пересечения, информационное содержание и среднее время выполнения конструкции «Разветвление».

$$d(x_k) = \begin{cases} \sum_{i=1}^{3} Q_i = 0 \rightarrow x_k \in \omega_1 \\ \sum_{i=1}^{3} Q_i = 1 \rightarrow x_k \in \omega_2 \\ \sum_{i=1}^{3} Q_i = 2 \rightarrow x_k \in \omega_3 \\ \sum_{i=1}^{3} Q_i = 3 \rightarrow x_k \in \omega_4 \end{cases} \quad (3)$$

$$IF(q_i \in [m_i \pm \sigma_i])THEN(Q_i = 0), \quad (4)$$

где q_i – ключевые показатели трех сущностей процесса вычислений;

m_i – математическое ожидание этих показателей;

σ_i – коэффициент строгости контроля целостности.

Коэффициент строгости позволяет гибко регулировать процесс контроля целостности в зависимости от назначения АС, и контролируемого динамического объекта, а также в зависимости от закона распределения метрического показателя.

Решающее правило определяет четыре класса целостности, которые представлены на рисунке 7.

Для решения дальнейшей задачи, и решение близости образа динамического объекта, относящегося ко второму (ω_2) или третьему (ω_3) классу, к целостному или нецелостному применяется аппарат нечетких распознавателей (кластерный анализ).

Специфика предлагаемого алгоритма нечеткой классификации состоит в том, что в отличие от существующих в данном алгоритме оценок параметра $q_k^* = (1, q_k)$ эта оценка может производиться нечетко и определяться функцией принадлежности (мерой близости) $\mu_0(q_k) = \max(\omega_i(q_k))$ для любых q_i.

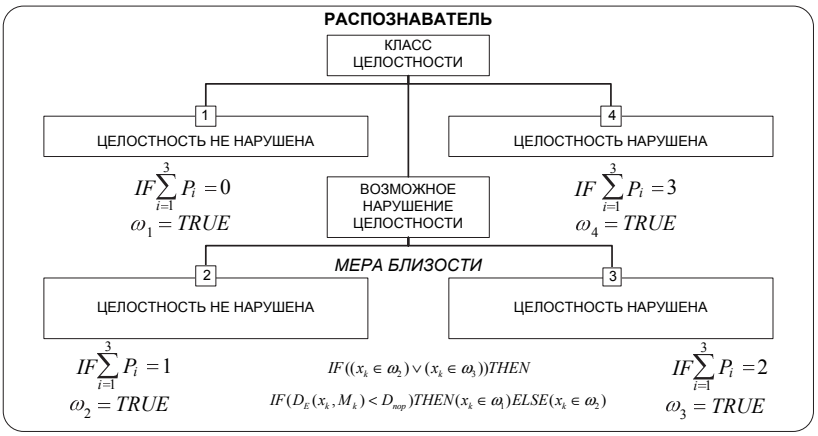


Рис. 7. Распознаватель классов состояний целостности динамических объектов

В качестве функции принадлежности (меры близости) было выбрано евклидово расстояние, которое вычисляется по формуле 5.

$$D_E(X, M_i) = \left\| X - M_i \right\|, \tag{5}$$

где M_i – вектор математического ожидания $M_i = (q_{11}, ..., q_{ki})$ его численных показателей.

В силу естественных рассуждений, очевидно, что предъявляемый для распознавания образ необходимо отнести к тому из классов, евклидово расстояние, до которого является наименьшим. Тогда соответствующее решающее правило может быть записано в представленном ниже виде (6).

$$X \in \omega_i = \arg\min D_E(X, M_i). \tag{6}$$

Приведенный аппарат, а также в частности кластерной анализ, позволил построить распознаватель вредоносных воздействий на исполняемый код вычислительного процесса, который, помимо четкой классификации динамического объекта по признаку целостности, осуществляет процесс анализа при неопределенной конфигурации численных показателей.

5. Контроль целостности динамических объектов антивирусного средства Dr.Web 4.44. Сбор данных о динамических объектах Dr.Web, и последующий их анализ показал, что исходными данными для нашей технологии являются:

1. Процессы, протекающие в рамках функционирования антивирусного средства.
2. Важные характеристики этих процессов, которые необходимы для корректного снятия дампа памяти: страницы виртуальной памяти, регионы виртуальной памяти и их атрибуты.
3. Корректные дампы памяти протекающих процессов. Дамп имеет стартовый и конечный адреса, а также имеет различную форму представления (двоичную, шестнадцатеричную, ASCII-вид и т.п.).
4. Дизассемблированный листинг дампа памяти. Эта форма представления дампа в виде ассемблерных команд. Корректный переход от двоичного вида к дизассемблированному коду является нетривиальной задачей, и содержит ряд трудноразрешимых проблем.
5. Управляющий граф процесса. Он содержит различные уровни детализации, которые зависят от сложности самого исследуемого процесса и от применяемых функций свертки в процессе дизассемблирования и формализации листинга.

Перечисленные пять компонентов являются важнейшими составляющими исходных данных. От их корректной формы зависит правильность дальнейшего эталонирования процессов антивирусного продукта Dr.Web 4.44.

Формализация исходных данных. Этот этап технологии необходим для вычисления последовательности состояний и эквивалентных преобразований кода программы (управляющего графа), а также для моделирования функциональной структуры. Для осуществления этих действий требуется представить исходные данные в виде *автомата, продукции и грамматики.* Автомат позволит построить последовательность состояний (лексический разбор дампа), продукция поможет осуществить синтаксический разбор и эквивалентные преобразования (свертку графа) [8], а грамматика создаст функциональную модель структуры процесса (семантический разбор) [2]. Эквивалентные преобразования производятся только в

случае возникновения необходимости упростить код или уменьшить его объем, так как они могут в последующем привести к искажению функциональной структуры процесса [8]. Кроме того, продукционная модель дает возможность проверить корректность и завершаемость кода, но в целях решения поставленной задачи это не является необходимостью.

Пример формирования автомата на процессе spidernt.exe показан на рисунке 8.

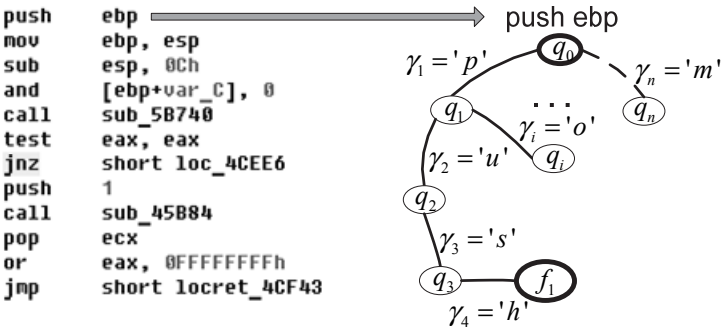


Рис. 8. Построение автомата переходов на графе состояний

На рисунке 8 представлены следующие обозначения:

q_0 – начальное состояние автомата;

$Q = \{q_1, q_2, ...q_n, ...q_i, ...q_{finish}\}$ – конечное множество состояний автомата;

$\Sigma = \{\gamma_1, \gamma_2, ...\gamma_n, ...\gamma_i, ...\gamma_{finish}\}$ – допустимый входной алфавит;

$\delta : Q \times \Sigma \to P(Q)$ – заданное отображение множества;

$F = \{f_1, f_2, ...f_n, ...f_i, ...f_{finish}\}$ - множество заключительных состояний.

Результатом работы автомата будет набор последовательных лексем, которые представляют собой конечные состояния автомата. Дальнейшие исследования потребуют объединение полученных лексем в более крупные синтаксические объединения – блоки. Для этого требуется построить продукционную систему вывода.

Результат работы с использованием продукционной формы представления процесса spidernt.exe показан на рисунке 9.

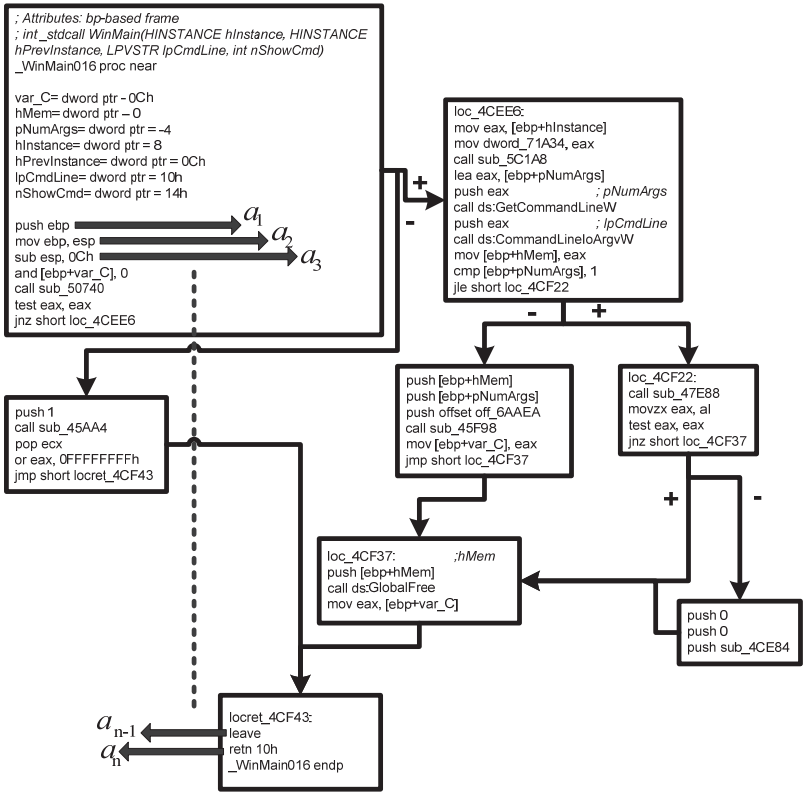


Рис. 9. Продукционная форма представления процесса spidernt.exe

На рисунке представлены следующие обозначения:

$A = \{a_1, a_2, ...a_n\}$ – множество входных символов (лексем) – алфавит;

F – множество формул в алфавите;

$F \subset A^* = a_1 \to a_2 \to a_3 \to ... \to a_n$

B – множество аксиом в алфавите;

$B \subset F$

R – множество правил вывода;

$R \subset F^{n+1}$

Аксиомы позволяют проверить корректность и завершаемость исследуемого кода (в данном случае это не требуется). Правила вывода помогают свернуть код (осуществить эквивалентные преобразования) в случае необходимости.

Третьим результатом процесса формализации исходных данных является модель функциональной структуры, основанная на грамматиках. Этот фрагмент отображает часть структуры, которая начинается с точки запуска процесса spidernt.exe – «StartPoint», и включает в себя все подпрограммы (функции), выполняющиеся с момента запуска.

Итогом процесса формализации являются три модели представления процесса вычислений, которые используются в дальнейшем.

Расчет метрик процесса spidernt.exe. На основе формального представления исходных данных производится расчет метрических показателей (структурных и информационных). Расчет временных метрик не осуществляется, так как он потребует создание дополнительного сложнейшего вычислительного стенда для оценки времени прохождения языковых конструкций процесса.

Для создания эталона требуется несколько временных срезов процессов антивирусного средства Dr.Web, то есть существуют несколько наборов входных данных разнесенных по времени. В качестве примера собрано 5 наборов исходных данных для процесса spidernt.exe.

Расчет полного набора метрических показателей требует существенных вычислительных и временных затрат, поэтому для осуществления эталонирования вычислительного процесса и последующего его контроля в некоторых случаях, когда требуется контролировать целостность процесса в режиме времени приближенном к реальному, допустимо использование меньшего количества метрических показателей. В качестве основных метрик, наиболее полно отражающих структуру вычислительного процесса, предлагается использовать:

1. Метрику точек пересечений. Механизм расчета этого показателя имеет следующий алгоритм:

a) из исходных данных считывается количество переходов условных и безусловных. Считывание производится в два массива (условный и безусловный), каждый переход характеризуется двумя значениями: стартовый и конечный адреса перехода (рисунок 10);

b) два массива сравниваются друг с другом поэлементно с целью поиска пересечений переходов по адресам. В результате находятся элементы и в том и в другом массиве, имеющие минимальное и максимальное количество пересечений. В «безусловном» массиве: $\min(a,b)=0$, $\max(c,d)=49$, где a,b – стартовый и конечный адреса безусловного перехода, имеющего минимально количество пересечений с другими переходами, c,d – стартовый и конечный адреса безусловного перехода, имеющего

максимальное количество пересечений с другими переходами. В «условном» массиве $\min(p,q)=0$, $\max(f,t)=9$, где p,q – стартовый и конечный адреса условного перехода, имеющего минимальное количество пересечений с другими переходами, f,t – стартовый и конечный адреса условного перехода, имеющего максимальное количество пересечений с другими переходами;

c) осуществляется расчет количественного показателя структурированности (метрики точек пересечения):

$$q_{15}=\frac{\max(c,d)}{\max(f,t)}=\frac{49}{9}=5,444 \qquad (7)$$

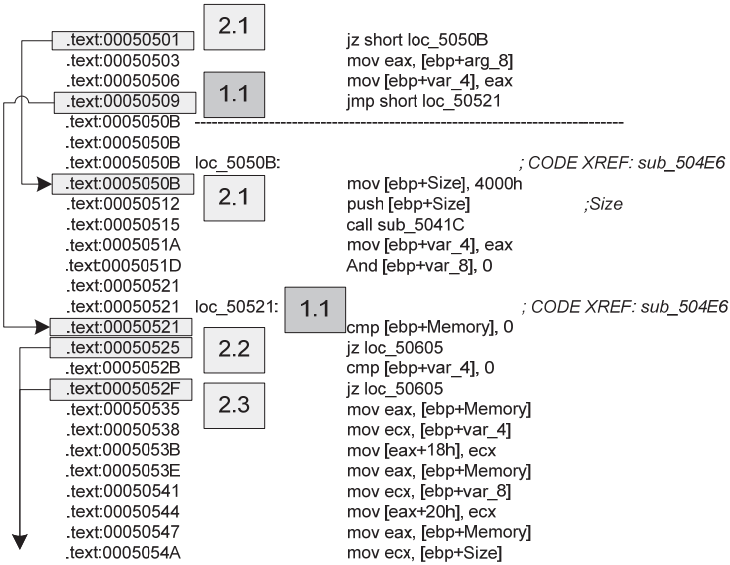


Рис. 10. Расчет метрики точек пересечения

2. Метрику Джилба. Она отражает насыщенность программы конструкциями циклов и переходов. Механизм расчета следующий:

a) считывается общее количество операторов цикла ($L_{loop}=567$), условия ($L_{jne}=26139$), безусловного перехода ($L_{jmp}=7147$). При считывании используются данные, полученные при расчете метрики точек пересечения, что сокращает время расчета;

b) производится подсчет количество модулей или подсистем (функций) $L_{sub}=607$ (рисунок 11);

c) вычисляется метрика Джилба:

$$q_{19} = \frac{N_{sv}}{L_{sub}} = \frac{33853}{607} = 55, \tag{8}$$

где $N_{sv} = L_{loop} + L_{jne} + L_{jmp} = 567 + 26139 + 7147 = 33853$

3. Метрику Вудворда. Она характеризует число узлов передачи управления и тесно связана с функциональным числом i-вершины. Представляет собой следующую тройку:

$$q_{17} = < F, U, P >, \tag{9}$$

где F - общее количество функциональных вершин всех узлов;
U - общее количество объединяющих вершин всех узлов;
P - общее количество предикатных вершин всех узлов.

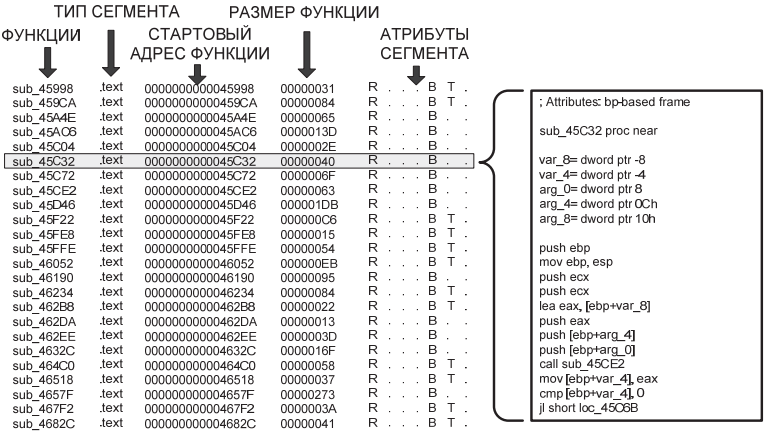


Рис. 11. Количество функций и их содержимое

Механизм расчета следующий:

a) процесс разбивается на узлы (функции) (рисунок 12). Количество узлов равно 607 ($i = \overline{1(1),607}$);

b) в каждой функции подсчитывается количество функциональных вершин(f_i);

c) в каждой функции подсчитывается количество объединяющих вершин(u_i);

d) в каждой функции подсчитывается количество предикатных вершин(p_i);

e) рассчитывается полное количество функциональных

($F = \sum_{i=1}^{607} f_i = 25263$), объединяющих ($U = \sum_{i=1}^{607} u_i = 27103$) и предикатных вершин ($P = \sum_{i=1}^{607} p_i = 42658$);

f) в итоге получается метрика Вудворда:

$$q_{17} =< 25263, 27103, 42658 >$$

Сокращенный эталон, отражающий структуру вычислительного процесса, имеет следующий вид:

$$Etalon_{spidernt.exe} < q_{15}, q_{17}, q_{19} >$$
$$Etalon_{spidernt.exe} < 5,5(\pm 0,5); < 25400(\pm 900), \quad (10)$$
$$27300(\pm 700), 42500(\pm 600) >; 55,5(\pm 0,5) >$$

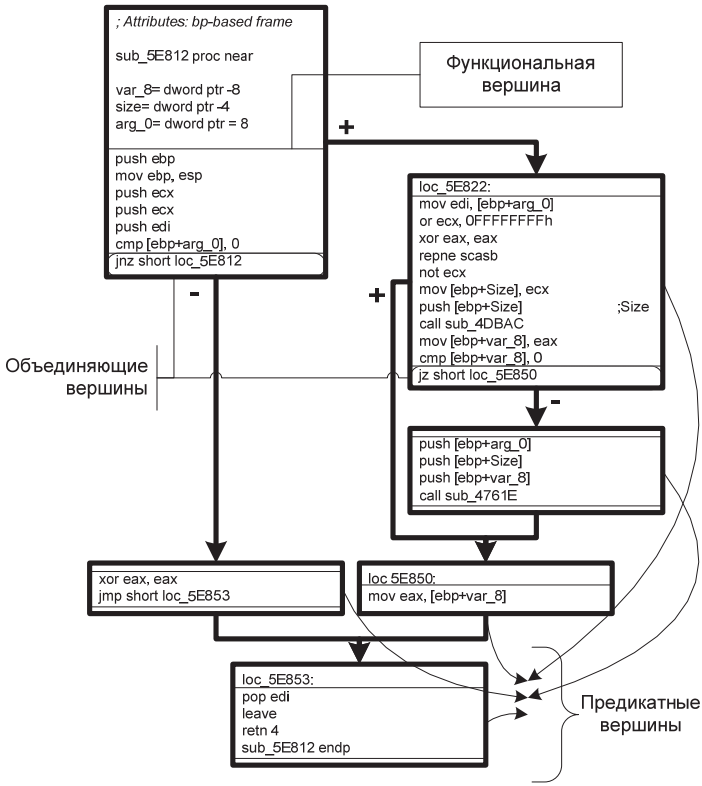


Рис. 12. Функциональные, объединяющие и предикатные вершины в узле

Все структурные метрики вычисляются разбором управляющего графа программы. Они отражены в таблице 1. В этой таблице приняты следующие обозначения: q_{11} – метрика Мак-Клура, q_{12} – метрика Пратта, q_{13} – метрика регулярных выражений, q_{14} – функциональная метрика, q_{15} – метрика точек пересечения, q_{16} – метрика Пивоварского, q_{17} – метрика Вудворда, q_{18} – метрика Чена, q_{19} – метрика Джилба.

Таблица 1. Структурные метрики процесса spidernt.exe

Метрика	Набор исходных данных				
	1	2	3	4	5
q_{11}	5	6	5	5	5
q_{12}	28	30	43	35	38
q_{13}	46 345	54 156	61 239	53 673	67 198
q_{14}	2,7	3,8	7,9	4,1	4,3
q_{15}	5,444	5,444	5,83	5,444	5,444
q_{16}	3,7	3,7	4,6	4,0	4,1
q_{17}	<25263, 27103, 42685>	<25145, 27512, 42698>	<25898, 27768, 43465>	<25345, 27128, 42491>	<25467, 27234, 42159>
q_{18}	1 892	1 543	2 567	1 897	1 954
q_{19}	55	55,9	56	55,3	55

Информационные метрики вычисляются путем синтаксического анализа дизассемблированного дампа памяти. Все информационные метрики, рассчитанные для 5 наборов исходных данных, показаны в таблице 2.

Таблица 2. Информационные метрики процесса spidernt.exe

Метрика	Набор исходных данных				
	1	2	3	4	5
q_{21}	0,32	0,45	0,42	0,7	0,52
q_{22}	5,73	7,67	8,9	5,9	6,23

В этой таблице приняты следующие обозначения: q_{21} – информационная энтропия, q_{22} – информационное содержание.

Полученные расчетные значения из пяти разных входных наборов данных позволяют построить эталонную метрическую модель процесса (spidernt.exe) антивирусного средства Dr.Web. Она состоит из

рассчитанных эталонных значений метрических показателей процесса, отражающих его структуру – структурные метрики (таблица 3), свойства – информационные метрики (таблица 4), а также доверительного интервала каждой метрики.

Полученный эталон позволяет контролировать целостность процесса функционирования spidernt.exe (одного из динамических компонентов антивирусного средства Dr.Web).

В качестве вредоносного воздействия на контролируемый динамический объект была осуществлена атака, связанная с изменением потока данных. Суть этой атаки заключалась в подмене потока сигнатур, считываемых из базы вирусных сигнатур. Это привело к снижению результативности обнаружения вирусов антивирусным средством Dr.Web. Системные мониторы, установленные в вычислительной системе, не зафиксировали искажений в действиях динамического объекта, следовательно, не смогли выявить деструктивное воздействие на этот объект. Предлагаемый распознаватель, позволил выявить существенные искажения в пространстве информационных метрик. Это отклонение является признаком нарушения целостности вычислительного процесса антивирусного средства.

Таблица 3. Структурный эталон процесса spidernt.exe

Метрика	Эталонное значение	Доверительный интервал
q_{11}	$5,2$	$\pm 1,2$
q_{12}	$34,8$	$\pm 16,25546$
q_{13}	$56522,2$	$\pm 21361,73381$
q_{14}	$4,56$	$\pm 5,27727$
q_{15}	$5,5212$	$\pm 0,4632$
q_{16}	$4,02$	$\pm 0,99318$
q_{17}	$< 25423,6;$ $27349;$ $42699,6 >$	$< \pm 778,29876;$ $\pm 764,51396;$ $\pm 1288,12552 >$
q_{18}	$1970,6$	$\pm 995,35389$
q_{19}	$55,44$	$\pm 1,29522$

Таблица 4. Информационный эталон процесса spidernt.exe

Метрика	Эталонное значение	Доверительный интервал
q_{21}	$0,482$	$\pm 0,37966$
q_{22}	$6,886$	$\pm 3,652$

Вредоносное воздействие, которое может осуществляться на вычислительный процесс путем добавления или уменьшения его функций, а также удаления или искажения данных, используемых в нем, моментально отражается в изменениях текущей метрической модели. Это позволяет выявить любое отклонение от эталона, и дает возможность утверждать о факте нарушения целостности функционирования этого процесса.

6. Заключение. Предлагаемый подход к контролю целостности вычислительных процессов с использованием метрического эталонирования, позволяет производить:

- расчёт метрических характеристик правильности структур, корректности свойств и устойчивости действий для эталонирования процесса вычислений;
- выбор ключевых параметров для первичного анализа процесса вычислений на основе построенных моделей представления процесса вычислений;
- распознавание признаков модификации процесса вычислений, на основании наличия которых выполняется классификация процесса вычислений к классам целостных и не целостных процессов, в том числе и с применением нечеткого классификатора;
- оценивать параметры контроля целостности с помощью корректирующих коэффициентов.

Этот подход в сочетании с другими средствами контроля целостности существенно повысит информационную безопасность АС, путем своевременного выявления преднамеренного или случайного искажения штатного состояния динамического объекта.

Литература

1. *Платонов А.А., Тимофеев В.И., Шаршаков В.Н., Ломако А.Г.* Модель угроз целостности вычислений в автоматизированных системах // Труды Института системного анализа Российской академии наук. 2013. №27. С. 321–337.
2. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции // М.: Мир. 1978. Т. 1. 614 с.
3. *Баранов С.Н., Тележкин А.М.* Метрическое обеспечение программных разработок // Труды СПИИРАН. 2014. №5(36). С. 5–27.
4. *Холстед М.Х.* Начала науки о программах // М.: Фин. и статистика. 1981. 128 с.
5. *Watson A.H., McCabe Th.J., Dolores R.* Structured Testing: a Testing Methodology Using the Cyclomatic Complexity Metric // National Institute of Standards and Technology Special Publication 500-235. 1996. 123 p.
6. *Neelamegam C., Punithavalli M.* Enhanced ensemble prediction algorithms for detecting faulty modules in object oriented systems using quality metrics // Journal of Computer Science. 2012. vol. 8. Issue 12. pp. 2075–2082.
7. *Карповский Е.Я., Чижов С.А.* Надежность программной продукции // Киев:

Техника. 1990. 171 с.
8. *Федорченко Л.Н.* Регуляризация контекстно-свободных грамматик на основе эквивалентных преобразований синтаксических граф-схем // Труды СПИИРАН. 2010. №4(15). С. 213–230.

References

1. Platonov A.A., Timofeev V.I., Sharshakov V.N., Lomako A.G. [Model of computation integrity threats in automated systems]. *Trudy Instituta sistemnogo analiza Rossijskoj akademii nauk – Proceedings of the Institute of systems analysis of Russian academy of sciences*. 2013. vol. 27. pp. 321–337. (In Russ.).
2. Aho A., Ulman Dzh. *Teoriya sintaksicheskogo analiza, perevoda i kompilyacii* [The theory of parsing, translation and compilation]. M.: Mir. 1978. vol .1. 614 p. (In Russ.)
3. Baranov S.N., Telezhkin A.M. [Metric for software development]. *Trudy SPIIRAN – SPIIRAS Proceedings*. 2014. vol. 5(36). pp. 5–27. (In Russ.).
4. Holsted M.H. *Nachala nauki o programmah* [Elements of Software Science]. M.: Finansy i statistika. 1981. 128 p. (In Russ.).
5. Watson A.H., McCabe Th.J., Dolores R. Structured Testing: a Testing Methodology Using the Cyclomatic Complexity Metric. National Institute of Standards and Technology Special Publication 500-235. 1996. 123 p.
6. Neelamegam C., Punithavalli M. Enhanced ensemble prediction algorithms for detecting faulty modules in object oriented systems using quality metrics. Journal of Computer Science. 2012. vol. 8. Issue 12. pp. 2075–2082.
7. Fedorchenko L.N. [Regularization of context free grammars on the base of equivalent transformations of syntax graph-schemes]. *Trudy SPIIRAN – SPIIRAS Proceedings*. 2010. vol. 4(15). pp. 213–230. (In Russ.).

Платонов Андрей Анатольевич — к-т техн. наук, старший преподаватель кафедры систем сбора и обработки информации, Военно-космическая академия имени А.Ф. Можайского. Область научных интересов: технология программирования, формальные методы в разработке и анализе программного обеспечения. Число научных публикаций — 29. aplaton80@gmail.com; ул. Ждановская, д. 13, Санкт-Петербург, 197082; р.т.: 8-812-237-19-60.

Platonov Andrey Anatolievich — Ph.D., senior lecturer of systems for collecting and processing information department, Mozhaisky Military Space Academy. Research interests: software engineering, the formal methods in development and the analysis of the software. The number of publications — 29. aplaton80@gmail.com; 13, Zhdanovskaya st., St. Petersburg, 197082, Russia; office phone: 8-812-237-19-60.

Тимофеев Владимир Ильич — преподаватель кафедры систем сбора и обработки информации, Военно-космическая академия имени А.Ф. Можайского. Область научных интересов: организационно-правовое обеспечение защиты информации. Число научных публикаций — 11. tivlil@mail.ru; ул.Ждановская, д. 13, г.Санкт-Петербург, 197198; р.т.: +7-812-237-19-60.

Timofeev Vladimir Ilyich — lecturer of systems for collecting and processing information department, Mozhaisky Military Space Academy. Research interests: organizational and legal protection of information. The number of publications — 11. tivlil@mail.ru; 13, Zhdanovskaya st., St. Petersburg, 197082, Russia; office phone: +7-812-237-19-60.

РЕФЕРАТ

Платонов А.А., Тимофеев В.И. **Контроль целостности динамических объектов вычислительных систем с использованием метрических эталонов.**

Рассмотренный в статье подход к контролю целостности динамических объектов развивает и дополняет методы контроля целостности информационных объектов в автоматизированных системах. Этот подход основывается на представлении вычислительного процесса в виде метрического эталона, включающего в себя метрики из полного базиса – правильности структуры, корректности свойств вычислимости и устойчивости действий. Каждая из составляющих этого базиса отражает наиболее значимые стороны вычислительного процесса, что позволяет отслеживать его искажения, вызванные преднамеренными или случайными внешними воздействиями.

Алгоритмы расчета метрик, составляющих эталон, приведены для динамических объектов антивирусного средства Dr.Web. Основной особенностью этих алгоритмов является, использование в качестве входных данных не исходных кодов программы, а дампа памяти, преобразованного к графу состояний. В алгоритме обработки дампа памяти используются приемы корректного дизассемблирования программ, а также методы эквивалентных преобразований на графах.

Процедура выявления изменений функциональных возможностей основана на решающем правиле и решающей функции, что позволяет отнести вычислительный процесс к классам целостных или не целостных динамических объектов автоматизированной системы. В случаях нечеткого распознавания используется мера близости и коэффициент строгости.

Результаты проведенного исследования позволяют сделать вывод об эффективности контроля динамических объектов с использованием метрических эталонов, но с большими затратами системных ресурсов. Эти затраты ориентируют использовать предлагаемый подход только для вычислительных процессов критически важных информационных объектов, например: средств защиты информации.

SUMMARY

Platonov A.A., Timofeev V.I. **Monitoring of Integrity of Dynamic Objects of Computing Systems with Use of Metric Standards.**

The approach to monitoring of integrity of dynamic objects considered in article develops and adds control methods of integrity of information objects in automated systems. This approach is based on representation of calculating process in the form of the metric standard including metrics from full base – correctness of structure, a correctness of properties of computability and stability of actions. Each of components of this base reflects the most significant sides of calculating process that allows to trace its distortions caused by premeditated or accidental external influences.

Algorithms of calculation of the metrics making a standard are given for dynamic objects of anti-virus means of Dr.Web. The main feature of these algorithms is, use as input data not of source codes of the program, but the memory dump transformed to a state graph. In algorithm of processing of a memory dump methods of incorrect disassembling of programs, and also methods of the equivalent conversions on graphs are used.

Procedure of detection of changes of the functional capabilities is based on the decisive rule and decision function that allows to refer calculating process to classes of integral or not integral dynamic objects of automated system. In cases of indistinct recognition the measure of closeness and coefficient of severity is used.

Results of the conducted research allow to draw a conclusion on efficiency of monitoring of dynamic objects with use of metric standards, but with big expenses of system resources. These expenses orient to use the offered approach only for calculating processes of crucial information objects, for example: information security features.