ОБ УЛУЧШЕНИИ ИЗБИРАТЕЛЬНОСТИ ОДНОГО АЛГОРИТМА ОПРЕДЕЛЕНИЯ АВТОРСТВА ВРЕДОНОСНОГО КОДА

Кононов Д.С.[1]

В статье рассмотрен подход к определению авторства вредоносного кода на основе сопоставления исполняемых программных кодов. На примере ПО BinDiff выявлен недостаток существующих алгоритмов, реализующих данный подход, заключающийся в низкой избирательности. Для преодоления данного недостатка была рассмотрена структура, строящаяся по исходному орграфу и представляющая собой дерево, у которого множество всех возможных орцепей совпадает с множеством всех возможных орцепей в исходном орграфе. Так как орцепи в графа передачи управления отождествляются с трассами исполнения программ, то в статье был предложен термин «дерево трасс» для описания этой структуры. Предшествующие алгоритмы определения изоморфизма деревьев трасс имели экспоненциальную временную сложность от порядка исходного орграфа. В статье представлен линейный от числа дуг исходного орграфа алгоритм определения изоморфизма деревьев трасс, основанный на каноническом индексе Земляченко. Данного результата удалось добиться за счёт оптимизации структуры дерева трасс в виде сжатого дерева. Включение в хеш–функцию инварианта орграфов на основе индекса Земляченко позволило улучшить избирательность по сравнению с аналогами на 50% уже для орграфов с порядком 51, не ухудшая при этом асимптотическую временную сложность.

***Ключевые слова**: ориентированный граф, дерево, инвариант, канонический индекс, временная сложность*

Для эффективного противодействия угрозам информационной безопасности [1] необходимо идентифицировать нарушителей, реализующих эти угрозы[2]. Эта задача особенно актуальна в сфере антивирусной защиты при обработке «входного потока» предположительно вредоносного программного обеспечения (ПО), имеющего жёсткие временные рамки. Так как существует большое количество модификаций одного и того же вредоносного ПО, то автоматическая идентификация его разработчика позволяет сократить время обнаружения действительно вредоносного ПО [2]. Схожая задача возникает и при поиске недекларируемых возможностей ПО (закладок) при сертификации по ранее выявленным экземплярам [3].

В связи с важностью данной задачи её изучению было посвящено множество научных исследований, не ограничивающееся работами [1, 4, 5, 7, 8]. Наиболее известной реализацией для идентификации нарушителей, атакующих информационные системы, следует признать ПО BinDiff. Разработке и описанию алгоритмов ПО BinDiff посвящён ряд научных работ, выполненных под руководством Томаса Дулейна. Основные научные результаты, положенные в основу ПО BinDiff, были представлены в журнале института ORT, ассоциированного с НАТО [4].

Несмотря на высокую скорость работы алгоритма, используемого в ПО BinDiff, он отличается низкой избирательностью (рис. 1). Указанный недостаток обусловливается методом определения совпадения по хеш-функции над отдельным узлом графа вызова или графа передачи управления. Вследствие этого недостатка в ПО Bindiff дополнительно используются более 10 различных эвристик [4]. В основе алгоритма ПО BinDiff лежит инвариант ациклических орграфов, названный авторами MD Индекс [5].

MD Индекс представляет собой хеш-функцию от графа управления программой размером 80 бит[3]. Для вычисления MD-индекса применяется следующий алгоритм [4]: пусть g – граф управления, e – дуга в графе управления, E_g – множество дуг в графе управления, $\mathfrak{G}$ – множество всех графов управления. Тогда каждому графу управления ставим в соответствие множество кортежей из пяти целых чисел

$$tup: \mathfrak{G} \to \mathfrak{P}(\mathbb{Z}^5), \quad (1)$$

1 Кононов Дмитрий Сергеевич, Минобороны России, Москва, sdk516@yandex.ru

2 Методика определения угроз безопасности информации в информационных системах, ФСТЭК России, http://fstec.ru/component/attachments/download/812

3 www.zynamics.com/downloads/vxclass_incident_response.pdf

$$tup(g) \mapsto \left\{ \begin{pmatrix} topologicalorder(src(e)), \\ indegree(src(e)), \\ outdegree(src(e)), \\ indegree(dst(e)), \\ outdegree(dst(e)) \end{pmatrix} | e \in E_g \right\}, \quad (2)$$

где $src(e)$ — исходящая вершина дуги, $dst(e)$ — конечная вершина дуги, $topologicalorder(n)$ — топологический порядок вершины n, $indegree(n)$ — полустепень захода вершины n, $outdegree(n)$ — полустепень исхода вершины n.

Кортежи формируют подмножество векторов в пятимерном пространстве $\mathbb{Q}^5$. Поскольку пятимерное векторное пространство может быть встроено в векторное пространство $\mathbb{Q}[\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{7}]$, постольку, каждому кортежу можно поставить в соответствие уникальное действительное число

$$emb(t) \mapsto t_0 + \sqrt{2}t_1 + \sqrt{3}t_2 + \sqrt{5}t_3 + \sqrt{7}t_4 \quad (3)$$

где $t = \{t_0, t_1, t_2, t_3, t_4\}$ – пятимерный кортеж, принадлежащий множеству кортежей, заданных отображением $tup(g)$.

MD-индексом называется действительное число, заданное следующей формулой

$$MDIndex = \sum_{t \in tup(g)} \frac{1}{\sqrt{emb(t)}}. \quad (4)$$

В приведённом примере, представленном на рис. 1, проявляется слабое место MD индекса, как и многих других инвариантов — в нём не учитывается глобальная структура графа, которая важна при его интерпретации как графа вызовов или графа передачи управления программы.

Далее в статье под любыми графами будем понимать только класс корневых ациклических орграфов. Термин «корневой орграф» определим как орграф, в котором выбрана одна вершина — корень орграфа, причём из неё существуют ормаршруты [6] до всех других вершин орграфа.

Для преодоления данного недостатка предлагается использовать часто встречаемое отображение ациклического орграфа в виде дерева, в котором каждый узел исходного графа дублируется в соответствии с его входящей полустепенью [7-9].

Одной из ошибок, допускаемых авторами при работе с данной структурой, является игнорирование факта о возможном экспоненциальном росте порядка раскрытого дерева по сравнению с порядком исходного орграфа [8]. Особенность такого дерева является то, что множество всех возможных орцепей [6] в нём совпадает с множеством всех возможных орцепей в исходном орграфе. В свою очередь орцепи в орграфе могут быть отождествлены с трассами исполнения программ. Для определённости и краткости в

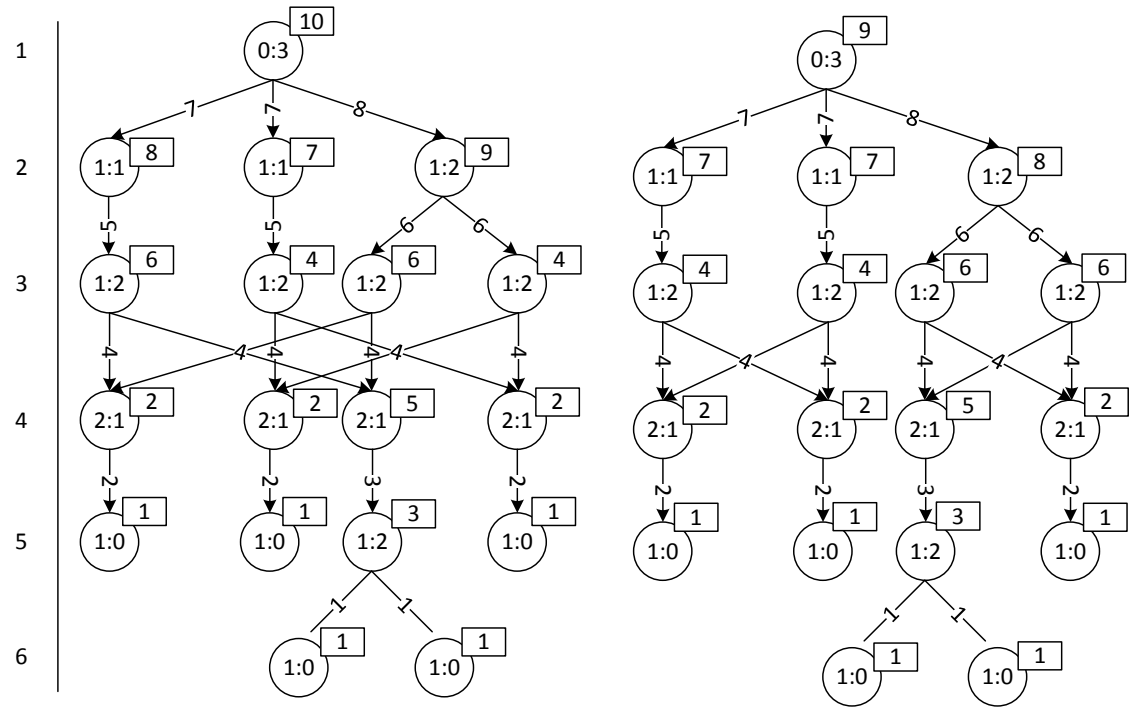

***Рис. 1**. Контрпример неразличимых MD-индексом графов: числа слева соответствует топологическому уровню вершин, числа внутри вершин представляют собой входящие и исходящие полустепени, рёбра с равными числами имеют равный MD-индекс, в квадратах отмечены индексы Земляченко соответствующих вершин раскрытого дерева*

контексте данной статьи будем называть такую структуру деревом трасс для конкретного ациклического орграфа.

Несложно показать, что у изоморфных орграфов их деревья трасс будут изоморфными [8], обратное же не верно. Таким образом, дерево трасс является инвариантом ациклических орграфов, причём в нём сохранена вся информация о множестве орцепей в исходном орграфе.

Существует алгоритм определения изоморфизма деревьев за линейное время — алгоритм Земляченко [10]. Однако с учётом возможного экспоненциального роста порядка дерева трасс, даже такой эффективный алгоритм будет иметь экспоненциальную временную сложность.

Для разрешения противоречия между низкой избирательностью MD Индекса и высокой временной сложностью алгоритмов определения изоморфизма дерева трасс предлагается разработанный автором алгоритм, нивелирующий эти недостатки за счёт учёта особенностей дерева трасс.

Заметим, что в дереве трасс поддеревья, соответствующие одной вершине в исходном графе, дублируются. Индекс Земляченко полностью определяется корневым поддеревом, индуцированным текущей вершиной [10], т.е. для расчёта индекса Земляченко достаточно один раз рассчитать индекс корневого поддерева, а затем распространить этот результат на все вершины раскрытого поддерева соответствующих одной и той же вершине в исходном графе.

Таким образом, можно создать структуру — сжатое дерево (рис. 2), эквивалентную дереву трасс, но, при этом, имеющую порядок, равный количеству рёбер в исходном графе. Сжатое дерево можно получить с использованием модификации алгоритма обхода в глубину [11]:

1) сортируем все дочерние вершины текущей вершины v_i в топологическом порядке;

2) выполняем обход вершин ациклического орграфа, используя алгоритм обхода в глубину, при этом дочерние вершины обрабатываются в соответствии с топологическим порядком;

3) если текущая вершина v_{i+1} ещё не обрабатывалась, то ей выставляется пометка — обработана;

4) если текущая вершина v_{i+1} уже обрабатывалась, то в граф добавляется вершина g, которая является дочерней вершине v_i — текущей на предыдущем шаге рекурсии, и новой вершине g назначается атрибут — ссылка на текущую обрабатываемую вершину v_{i+1} . При этом дуга, связывающая вершины v_i и v_{i+1} , удаляется.

На основе самого преобразования и класса исходного графа можно вывести некоторые свойства сжатого дерева.

1) В сжатом дереве присутствуют два типа вершин: с атрибутом и без атрибута. Вершины сжатого дерева с атрибутом назовём сжатыми, а все остальные — тривиальными.

2) В сжатой вершине в качестве атрибута используется ссылка на одну из тривиальных вершин. Будем говорить, что сжатая вершина ссылается на тривиальную вершину, если ссылка на неё указана в атрибуте сжатой вершины.

3) Порядок сжатого дерева равен количеству дуг в исходном графе, так как, согласно алгоритму, на каждую дугу добавляется одна вершина.

4) Существует биекция между тривиальными вершинами сжатого дерева и вершинами исходного графа.

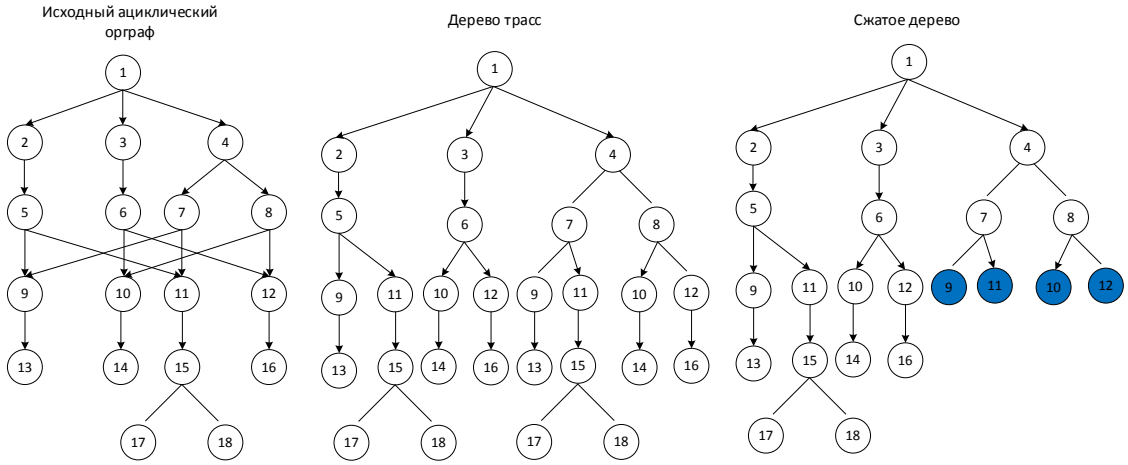

***Рис. 2**. Орграф и его представления в виде дерева трасс и сжатого дерева*

5) Поддерево сжатого дерева, состоящее только из тривиальных вершин, представляет собой остовное дерево [12] исходного графа.

6) Сжатое дерево является корневым деревом, а его корнем вершина, соответствующая корню исходного орграфа.

7) У изоморфных орграфов могут быть разные сжатые деревья, так как порядок обхода вершин, имеющих равный топологический порядок, не определён.

8) Все сжатые вершины являются листьями.

9) Тривиальная вершина, на которую ссылается сжатая вершина, имеет глубину большую или равную глубине сжатой вершины.

Из сжатого дерева может быть получено дерево трасс путём последовательной замены каждой сжатой вершины на дублированное корневое поддерево, индуцируемое тривиальной вершиной, на которую ссылается текущая сжатая вершина.

Кроме того, из сжатого дерева может быть получен граф, изоморфный исходному, путём замены каждой сжатой вершины v на дугу, исходящую из родительской вершины g текущей сжатой вершины v и оканчивающуюся в тривиальной вершине u, на которую она ссылается.

Автором был разработан алгоритм, который по сжатому древу вычисляет канонический индекс дерева трасс. Этот алгоритм заключается в трёх изменениях алгоритма Земляченко [13]:

1) изменение инициализации исходного алгоритма — во вспомогательный массив level заносятся только тривиальные вершины сжатого дерева в соответствии с их высотой;

2) индекс Земляченко для сжатых вершин вычисляется по ссылке на тривиальную вершину;

3) при записи тривиальной вершины в соответствующий вспомогательный массив children[h] обрабатываются все сжатые вершины, ссылающиеся на текущую тривиальную вершину, и записываются в соответствующие массивы children[h].

Кроме того, в индексе Земляченко можно использовать и параметры, связанные с программными элементами, соответствующим вершинам. Для этого достаточно добавить ко всем вершинам дополнительные дочерние вершины с уникальными индексами, после чего рассчитать индекс Земляченко.

Лемма 1. В ходе алгоритма перед обработкой сжатой вершины уже будет вычислен индекс Земляченко для тривиальной вершины, на который он ссылается.

Тривиальная вершина, на которую ссылается сжатая вершина, имеет глубину большую или равную глубине сжатой вершины. Согласно алгоритму, сжатые вершины добавляются только после обработки тривиальной вершины, на который они ссылаются. Следовательно, перед добавлением их в соответствующий массив уже будет вычислен индекс Земляченко для тривиальной вершины, на который они ссылаются.

Лемма 2. Все массивы children[h] в конце каждой итерации над текущим ярусом будут упорядочены.

На первой итерации все массивы упорядочены, так как у всех листьев индекс Земляченко равен 1. Все сжатые вершины, ссылающиеся на одну тривиальную вершину, имеют равные ему индексы, так как в дереве трасс они индуцируют изоморфные корневые поддеревья. При записи тривиальной вершины в массив children[h], он, согласно алгоритму Земляченко, имеет максимальный индекс среди всех вершин, присутствующих в children[h]. Добавление сжатых вершин в массив children[h] происходит сразу за добавлением в него тривиальной вершины, на которую они ссылаются. Таким образом, индексы сжатых вершин также будут не меньше максимального индекса среди всех вершин, уже добавленных в массивы children[h]. Заметим, что если массивы children[h] были отсортированы в неубывающем порядке перед добавлением сжатых вершин, ссылающихся на текущую тривиальную вершину, то и после их добавления массивы children[h] останутся отсортированными в том же порядке.

Теорема. В случае, если сжатое дерево уже построено, то модифицированный алгоритм Земляченко вычисляет инвариант дерева трасс для орграфа за линейное время в зависимости от количества дуг в исходном орграфе $\Theta(|E|)$, при этом требуется памяти не более чем $\Theta(|E|)$.

Индекс Земляченко текущей вершины зависит только от индексов Земляченко её дочерних вершин. Согласно лемме 1, в случае, когда среди дочерних вершин присутствует сжатая вершина, то её индекс уже будет рассчитан. По лемме 2, все массивы children[h] будут отсортированы в неубывающем порядке. Кроме того, индекс сжатой вершины равен индексу Земляченко корневого поддерева, индуцированного вершиной в дереве трасс, соответствующему этой сжатой вершине. Таким образом, значения внутренних параметров алгоритма, используемых при вычислении индексов тривиальных вершин и соответствующих им вершин в дереве трасс, будут эквивалентны. Следовательно, индекс, рассчитанный по сжатому

дереву, будет равен индексу Земляченко, рассчитанному для соответствующего дерева трасс.

В плане вычислительной сложности приведённый алгоритм расчёта индекса по сжатому дереву не отличается от алгоритма расчёта индекса Земляченко для этого же дерева, но без учёта атрибутов. Так как алгоритм Земляченко имеет линейную сложность по времени и по памяти от порядка дерева $\Theta(N)$, а порядок сжатого дерева равен числу дуг в исходном орграфе, то и временная сложность рассматриваемого алгоритма будет $\Theta(|E|)$, а требования к памяти ограничены $\Theta(|E|)$, где E — множество дуг исходного орграфа. Ч.т.д.

Можно показать, что топологический порядок вершин исходного орграфа разбивает множество вершин сжатого дерева на ярусы, то есть для расчёта топологического порядка вершины достаточно присваивать ей значение $deep(u)$, где u – соответствующая ей тривиальная вершина сжатого дерева, а функция $deep$ возвращает значение глубины вершины в дереве.

Добавив к рассмотрению, кроме глубины вершины, ещё и индекс Земляченко этой вершины, можно ввести дополнительный порядок среди вершин яруса, которые иначе были бы неотличимы. Рассмотрим параметр Z, задающийся следующим выражением

$$Z = deep(u) + \frac{1}{Zemlyachenco(u)}, \qquad (5)$$

где $Zemlyachenco(u)$ — функция возвращающая индекс Земляченко для вершины u в раскрытом дереве.

Очевидно, что параметр $Z \in \mathbb{Q}$, так как $deep(u) \in \mathbb{N} \cup 0$ и $Zemlyachenco(u) \in \mathbb{N}$. Следовательно, кортежи с использованием параметра Z также могут быть встроены в векторное пространство $\mathbb{Q}[\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{7}]$.

В итоге, если в определении кортежей MD-индекса вместо одного из параметров, задающегося топологическим порядком вершины, использовать параметр Z (соответствующий инвариант назовём Z-индексом для определённости), то формально можно записать

$$MDIndex(G_1) \neq MDIndex(G_2) \Rightarrow \\ \Rightarrow ZIndex(G_1) \neq ZIndex(G_2), \qquad (6)$$

$$ZIndex(G_1) = ZIndex(G_2) \Rightarrow \\ \Rightarrow MDIndex(G_1) = MDIndex(G_2). \qquad (7)$$

В этом смысле Z-индекс является более точным чем MD-индекс.

Временная сложность алгоритма расчёта топологического порядка вершин орграфа составляет $\Theta(|E| + |V|)$ [13]. Учитывая, что алгоритм построения сжатого дерева имеет временную сложность $\Theta(|E| + |V|)$, а вычисление индекса по сжатому дереву $\Theta(|E|)$, то можно сделать вывод: вычисление предложенного в статье инварианта Z-индекса имеет временную сложность $\Theta(|E| + |V|)$, при этом требуя $\Theta(|E|)$ памяти.

Таким образом, при использовании Z-индекса появляются преимущества в избирательности, и при этом временная сложность остаётся линейной от суммы числа дуг и вершин в исходном графе. Как видно из рис. 1 (числа в прямоугольниках при вершинах), инвариант дерева трасс на основе индекса Земляченко различает два приведённых графа даже без дополнительных параметров, используемых в MD-индексе.

В табл. 1 и 2 представлены результаты вычислительного эксперимента, посвящённого определению преимущества предлагаемого -индекса по сравнению с MD-индексом. Были вычислены соответствующие инварианты для множества всех корневых ациклических графов, имеющих порядок

Таблица 1.

Количество различных значений инвариантов графов для всех корневых ациклических орграфов

Кол-во вершин	MD-индекс (MD)		Z-индекс		Разность различающих способностей $Dis(Z) - Dis(MD)$	$\left(\frac{Dis(Z)}{Dis(MD)} - 1\right) * 100\%$	Отношение к пред. знач. $\frac{Dis(Z)}{Dis(MD)}$
	Различающая способность $Dis(MD)$	Отн. кол-ва св. ац. орграфов	Различающая способность $Dis(Z)$	Отн. кол-ва св. ац. орграфов			
4	20	0,833	20	0,833	0	0%	—
5	182	0,682	182	0,682	0	0%	—
6	3441	0,609	3449	0,611	8	0,2%	—
7	134154	0,565	135688	0,572	1534	1,1%	1,009
8	10622664	0,53	10892825	0,544	270161	2,5%	1,014

Таблица 2.

Количество различных значений инвариантов графов для всех корневых ациклических орграфов

Кол-во вершин	MD-индекс (MD)		Z-индекс		Разность различающих способностей $Dis(Z) - Dis(MD)$	$\left(\frac{Dis(Z)}{Dis(MD)} - 1\right) * 100\%$	Отношение к пред. знач. $\frac{Dis(Z)}{Dis(MD)}$
	Различающая способность $Dis(MD)$	Отн. кол-ва св. ац. орграфов	Различающая способность $Dis(Z)$	Отн. кол-ва св. ац. орграфов			
8 (2)	2	0,5	3	0,75	1	50%	—
9 (3)	7	0,438	12	0,75	5	71%	1.143
10 (4)	36	0,321	64	0,571	28	78%	1.037
11 (5)	340	0,219	623	0,401	283	83%	1.031
12 (6)	6526	0,146	12196	0,273	5 670	87%	1.02
13 (7)	256364	0,096	488075	0,183	231711	90%	1.019
14 (8)	20455825	0,062	39641001	0,121	19185176	94%	1.018

не больше заданного. В связи с ограничениями современной вычислительной техники для более высоких порядков эксперимент проводился над специальным типом орграфов. Показателем избирательности была принята различающая способность — мощность множества различных значений, принимаемых инвариантами на соответствующем множестве графов. В результате проведённого автором эксперимента в обоих случаях было показано, что предлагаемый инвариант является более избирательным по сравнению с MD-индексом, а также продемонстрирована истинность выражений (6) и (7). Кроме того, при увеличении порядка графов выигрыш в избирательности растёт и, согласно экстраполяции, уже при порядке орграфов 51 выигрыш в избирательности достигает 50%.

Эксперименты над случайными орграфами показали, что при сохранении линейной временной сложности средний выигрыш по времени составляет 4,26 раз в пользу MD-индекса из-за дополнительных вычислений в Z-индексе. Однако для случайного орграфа с порядком 1 000 000 и с максимальной исходящей полустепенью вершин, составляющей несколько десятков, время работы алгоритма расчёта предлагаемого инварианта Z индекса измеряется десятками секунд, т.е. данную разность можно считать несущественной на практике.

Таким образом, представленный алгоритм позволит повысить качество решения задач, связанных с обнаружением и идентификацией авторства вредоносного ПО или закладок. Также в ходе экспериментальной проверки исследований была разработана эффективная реализация алгоритма вычисления Z-индекса, которая исключала этап построения сжатого дерева, объединив его с этапом вычисления индекса Земляченко для дерева трасс непосредственно по исходному орграфу. За счёт этого удалось снизить дополнительные требования к вычислительным ресурсам до уровня, которым можно пренебречь на практике. Кроме того, в отличие от предыдущих исследователей, которые также рассматривали структуру аналогичную дереву трасс, удалось добиться полиномиальной временной сложности алгоритма от порядка исходного орграфа. Особо следует отметить, что в предложенном алгоритме учитываются не только локальные особенности программного кода, а ещё и все возможные трассы исполнения в нём.

***Рецензент**: Голов Игорь Юрьевич, кандидат технических наук, Минобороны России, golovnew2@yandex.ru.*

Литература:

1. Шеремет И.А. Угрозы техносфере России и противодействие им в современных условиях // Вестник академии военных наук. 2014. № 1 (46). С. 27-34.
2. Осовецкий Л.Г., Стремоухов В.Д. Определение авторства вредоносного кода с использованием метода сжатия данных // Программные продукты и системы. 2013. № 3. С. 31.
3. Марков А.С., Шеремет И.А. Теоретические аспекты сертификации средств защиты информации // Оборонный комплекс - научно-техническому прогрессу России. 2015. № 4 (128). С. 7-15.
4. Dullien T., Carrera E., Eppler S.M., Porst S. IST-091 – Information Assurance and Cyber Defence. NATO Research and Technology Organization, Automated Attacker Correlation for Malicious Code. Tallinn, Estonia. 2010. V. 26, pp. 1-10.

5. Dullien T., Meyer-Eppler S. Method for Characterization of a Computer Program Part, US20110067010 A1, March 17, 2011.
6. Зыков А.А. Основы теории графов. М.: Наука, 1987. 384 с.
7. Lingas A. Trees in Algebra and Programming 8th Colloquium Proceedings (CAAP'83) // An application of maximum bipartite c-matching to subtree isomorphism. L'Aquila. 1983. С. 284-299.
8. Саргсян С., Курмангалеев Ш., Белеванцев А., Асланян А., Балоян А. Масштабируемый инструмент поиска клонов кода на основе семантического анализа программ // Труды Института системного программирования РАН. 2015. Т. 27. № 1. С. 39-50.
9. Саргсян С., Курмангалеев Ш., Белеванцев А., Аветисян А. Масштабируемый и точный поиск клонов кода // Программирование. 2015. № 6. С. 9-17.
10. Земляченко В.Н. Установление изоморфизма деревьев // Вопросы кибернетики. М., 1973. С. 54-60.
11. Sedgewick R., Wayne K. Algorithms. 4 ed. Addison-Wesley, 2011.
12. Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л., Штайн К. Алгоритмы: построение и анализ. 3-е изд. М.: Изд. «Вильямс», 2014. 1328 с.
13. Dinitz Y., Itai A., Rodeh M. On an Algorithm of Zemlyachenko for Subtree Isomorphism, Information Processing Letters, V. 70, No 3, 1999, pp. 141-146.

ON IMPROVEMENT OF THE SELECTIVITY OF ONE ALGORITHM FOR DETERMINE MALWARE AUTHORSHIP

Kononov D.S.[4]

In the article the approach to the determent of malicious code authorship by comparing executable program code is examined. The lack of selectivity of existing algorithms that implement this approach is revealed by the example of BinDiff. To overcome this disadvantage the tree, which is based on the original digraph and has the set of all possible paths equals to the same set of the initial digraph, is examined. Since the paths of control flow graph represent the execution traces of the program, in the article the term "tree of traces" has been proposed to describe this structure. Previous algorithms for determining the isomorphism of "tree of traces" have exponential time complexity from the order of the initial digraphs. In the article a new algorithm for determining the isomorphism of "tree of traces", which is based on Zemlyachenko canonical index and has linear time complexity from the number of arcs of the initial digraphs, is presented. This result has been achieved by optimizing the "tree of traces" structure in the form of "compressed tree". The inclusion in the hash function the graph invariant based on Zemlyachenko index has improved selectivity in comparison with analogues for 50% for the graphs of the order of 51 without degrading the asymptotic time complexity.

Keywords*: directed graph, tree, invariant, canonical index, time complexity.*

References:

1. Sheremet I.A. Ugrozy tekhnosfere Rossii i protivodeystvie im v sovremennykh usloviyakh, Vestnik akademii voennykh nauk, 2014, No 1 (46), pp. 27-34.
2. Osovetskiy L.G., Stremoukhov V.D. Opredelenie avtorstva vredonosnogo koda s ispol'zovaniem metoda szhatiya dannykh, Programmnye produkty i sistemy, 2013, No 3, pp. 31.
3. Markov A.S., Sheremet I.A. Teoreticheskie aspekty sertifikatsii sredstv zashchity informatsii, Oboronnyy kompleks - nauchno-tekhnicheskomu progressu Rossii, 2015, No 4 (128), pp. 7-15.
4. Dullien T., Carrera E., Eppler S.M., Porst S. IST-091 – Information Assurance and Cyber Defence. NATO Research and Technology Organization, Automated Attacker Correlation for Malicious Code. Tallinn, Estonia, 2010. V. 26, pp. 1-10.
5. Dullien T., Meyer-Eppler S. Method for Characterization of a Computer Program Part, US20110067010 A1, March 17, 2011.
6. Zykov A.A. Osnovy teorii grafov. M.: Nauka, 1987. 384 p.
7. Lingas A. Trees in Algebra and Programming 8th Colloquium Proceedings (CAAP'83), An application of maximum bipartite c-matching to subtree isomorphism. L'Aquila. 1983, pp. 284-299.
8. Sargsyan S., Kurmangaleev Sh., Belevantsev A., Aslanyan A., Baloyan A. Masshtabiruemyy instrument poiska klonov koda na osnove semanticheskogo analiza programm, Trudy Instituta sistemnogo programmirovaniya RAN, 2015. V. 27, No 1, pp. 39-50.
9. Sargsyan S., Kurmangaleev Sh., Belevantsev A., Avetisyan A. Masshtabiruemyy i tochnyy poisk klonov koda, Programmirovanie, 2015, No 6, pp. 9-17.
10. Zemlyachenko B.H. Ustanovlenie izomorfizma derev'yev, Voprosy kibernetiki. M., 1973, pp. 54-60.
11. Sedgewick R., Wayne K. Algorithms. 4 ed. Addison-Wesley, 2011.
12. Kormen T.Kh., Leyzerson Ch.I., Rivest R.L., Shtayn K. Algoritmy: postroenie i analiz. 3-e izd. M.: Izd. «Vil'yams», 2014. 1328 p.
13. Dinitz Y., Itai A., Rodeh M. On an Algorithm of Zemlyachenko for Subtree Isomorphism, Information Processing Letters, V. 70, No 3, 1999, pp. 141-146.

4 Kononov Dmitriy, Russian Defense Ministry, Moscow, sdk516@yandex.ru