ISSN 2782-2001
Системы анализа и обработки данных
том 83, № 3, 2021, с. 37–52

http://journals.nstu.ru/vestnik
Analysis and data processing systems
Vol. 83, No. 3, 2021, pp. 37–52

ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

INFORMATICS, COMPPUTER ENGINEERING AND CONTROL

УДК 004.852 + 004.492.3 DOI: 10.17212/2782-2001-2021-3-37-52

Технология идентификации ключевых особенностей в последовательностях API-вызовов вредоносных программ*

В.В. ВОРОНИН[a], А.В. МОРОЗОВ[b]

680035, РФ, г. Хабаровск, ул. Тихоокеанская, 136, Тихоокеанский государственный университет

[a] *004183vvv@mail.ru* [b] *2014102127@pnu.edu.ru*

Сегодня с компьютерной безопасностью так или иначе сталкивается практически каждый. Для обеспечения контроля угроз безопасности вредоносного программного обеспечения (ПО) используют антивирусные программы.

Обычные способы обнаружения вредоносного ПО уже недостаточно эффективны, в настоящее время для этих целей стали применять нейронные сети и технологию поведенческого анализа. Анализ поведения программ – сложная задача, так как не существует четкой последовательности действий, исполнив которые можно точно идентифицировать программу как вредоносную. Кроме того, такие программы используют меры противодействия подобному обнаружению, например, зашумление последовательности своей работы бессмысленными действиями. Существует также проблема однозначной идентификации класса вредоносного ПО вследствие того, что вредоносные программы могут использовать схожие методы и при этом относиться к разным классам. В таком случае принадлежность вредоносных программ можно фиксировать, используя комплексные методы. В настоящей работе предлагается использовать методы NLP, такие как word embedding, и LDA применительно к задачам анализа последовательностей API вызовов вредоносного ПО с целью установления наличия семантических зависимостей и оценки эффективности применения данных методов.

Полученные результаты свидетельствуют о реальной возможности выделения ключевых особенностей поведения вредоносных программ. Применение этих особенностей в тексте работы для сканирования вредоносного программного обеспечения помогает обнаружить, например, попытки зашумления API-последовательностей, что само по себе может говорить о попытке сокрытия вредоносных целей исполнения программы или дать возможность выполнить классификацию вредоносных программ с использованием взаимозависимых совокупностей отдельных частей последовательности. Применяемая в настоящей работе технология в перспективе позволит существенно улучшить технологию обнаружения и идентификации вредоносных программ.

* *Статья получена 05 февраля 2021 г.*

Ключевые слова: векторное представление слов, обработка естественного языка, API-вызовы, вредоносное ПО, латентное размещение Дирихле, *n*-граммы, тематическое моделирование, кластеризация

ВВЕДЕНИЕ

Обработка API-последовательностей по ряду причин имеет сложный характер. Обработать или даже проанализировать последовательность вручную достаточно сложно, в лучшем случае такая последовательность будет содержать сотни вызовов, а иногда – сотни тысяч. Зачастую последовательность содержат от 2 до 10 тысяч вызовов. Кроме того, самих типов вызовов достаточно много, только часто используемых более 400. В случае когда речь идет о последовательностях API-вызовов вредоносного ПО, часто можно говорить еще и о целенаправленном зашумлении таких последовательностей лишними, не несущими нагрузки вызовами для затруднения их анализа. Проанализировать такие последовательности вручную крайне затруднительно, найти зависимость между тысячами таких последовательностей еще сложнее. Однако для такой задачи хорошо подходит машинное обучение [1].

Применение машинного обучения для анализа API-последовательностей не новая идея, в том или ином виде такая обработка уже применялась, давая хороший результат. Однако при усложнении зашумления и в некоторых других случаях эти методы оказываются неэффективны. Например, обнаружение возможно, а верная классификация оказывается затруднительна. Кроме того, в ходе дальнейшего развития вредоносного ПО эффективность чистого анализа последовательностей будет падать, так как могут меняться механизмы его функционирования.

Конечной целью применения NLP (Natural Language Processing) для обработки API-последовательностей является обнаружение локальных целей и механизмов работы анализируемой программы, а также последующий анализ совокупностей их взаимодействия. Кроме того, такой анализ направлен и на выявление механизмов зашумления или попыток зашумления. Это позволит улучшить работу сканера вредоносного ПО, выявить и предсказать отдельные механизмы функционирования такого ПО и, как следствие, усложнить попытки обхода сканера.

В настоящей статье предлагается оригинальное использование методов NLP для обработки последовательностей API-вызовов вредоносных программ. В технологию обработки этих последовательностей включены семантический анализ [2, 3], векторизация отдельных API-вызовов, поиск и отсев н-грамм, а также методика LDA (Latent Dirichlet Allocation) для тематического моделирования и кластеризации результатов обработки на разных этапах.

Основная задача настоящей работы – обоснование реальной возможности использования методов семантического анализа текста основанных на вложениях для обработки последовательностей API-вызовов на примере решения проблем, обнаруженных в работе [1].

1. ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА API-ПОСЛЕДОВАТЕЛЬНОСТЕЙ

В статье исследуются API-последовательности вредоносных программ, организованные в виде двух наборов данных, которые открыто распространяются в Интернете. Эти наборы данных объединены в один набор большего размера, насчитывающий до обработки более 13 000 образцов. Аналогичные наборы данных были использованы в работе [1].

Набор данных – это собранные при помощи специального программного обеспечения (например, Cuckoo Sandbox) последовательности API-вызовов различных программ, которые представляют собой строки различной длины, хранящие API-вызовы в той последовательности, в которой они были вызваны анализируемой программой. Дополнительно к каждой последовательности имеется обозначение принадлежности программы.

Для эффективной работы алгоритмов обработки текста необходима предварительная обработка текстовых последовательностей [4]. В задаче анализа последовательностей API-вызовов предварительная обработка также необходима.

В качестве предварительной обработки последовательностей API-вызовов были предложены и реализованы следующие два этапа.

1. Из анализируемых последовательностей исключаются многократно повторяющиеся подряд API-вызовы. Такие вызовы не несут дополнительной полезной информации и засоряют обрабатываемую последовательность.

2. После этапа 1 исключаются последовательности менее 500 и более 5000 вызовов.

2. АНАЛИЗ API-ВЫЗОВОВ

Для эффективной обработки последовательностей API-вызовов необходимо выполнять компоновку таких вызовов в группу по определенному признаку. Процесс такой компоновки сложен. Кроме того, возможно выполнить компоновку на основе различных признаков, получив отличные друг от друга варианты группировки. В случае обработки API-вызовов это является проблемой, так как различные варианты группировки могут оказать значительное влияние на точность работы и скорость обучения нейронной сети [1]. Однако ручная группировка по принципу функционирования того или иного вызова исходя из документации для каждого из этих вызовов может на деле быть неэффективной или вообще не отражать сути применения таких вызовов в программах.

Для выявления схожести и последующей группировки предложен другой метод, а именно: предлагается использовать вложения (embeddings) и способ представления семантической близости [5, 6]. Так, каждому слову или в данному случае API-вызову ставится в соответствие вектор, обозначающий его смысловое положение относительно других слов пространства, обозначенного данными вектором. Для вычисления сходства двух векторов используется коэффициент Отиаи (геометрический коэффициент) [7].

Для таких векторов обычно используется большое количество измерений, так как зачастую для описания множества свойств определенного слова может потребоваться много различных характеристик. Однако зачастую

невозможно точно установить необходимое количество измерений для смысловых векторов. Особенно в данном случае, так как область применения используемых методов имеет не совсем обычный характер.

Далее приведены результаты экспериментальных исследований векторов API-вызовов в различных ситуациях.

Оттенком обозначена степень близости значения ячейки к единице, знак плюс внутри ячейки обозначает положительное ее значение, знак минус – отрицательное значение ячейки. Чем ближе ячейки по оттенку и знаку, тем более схожи векторы в отношении этого свойства.

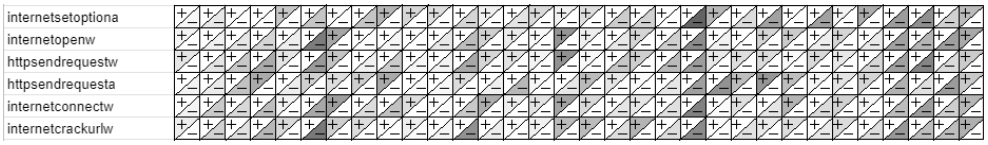

Рис. 1. Векторы API-вызовов для взаимодействия с сетью

Fig. 1. API vectors for interacting with the network

Через некоторые из столбцов для части из вызовов проходит линия из совпадающих по оттенку и знаку ячеек, это означает их близость по этим параметрам, однако точное значение этих параметров неизвестно. Кроме того, правая часть изображения имеет схожий вид, что также означает близость смыслового значения этих вызовов в определенной области.

На рис. 2 изображены API-вызовы, сгруппированные по параметру близости к API-вызову Ntopenkey. Деление происходит на две группы: наиболее близкие и наиболее удаленные по отношению к Ntopenkey.

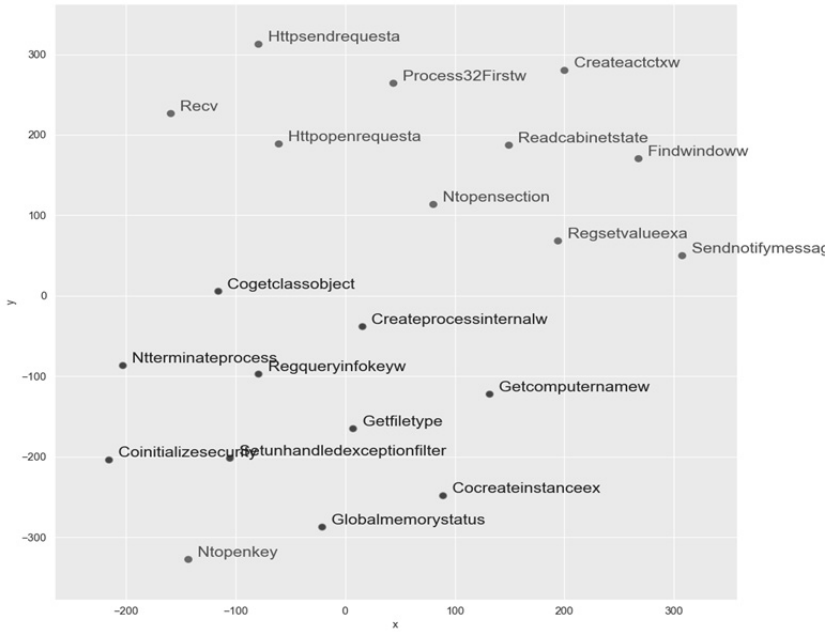

Рис. 2. API-вызовы, сгруппированные относительно Ntopenkey

Fig. 2. API calls grouped relative to Ntopenkey

В [8] авторами получены и проанализированы результаты кластеризации API-вызовов с применением технологии t-SNE, сформированные вокруг группы API-вызовов *ntfreevirtualmemory, findwindowa, shutdown* и *setfileattributesw.*

В настоящей статье для отображения *n*-мерного пространства векторов на двухмерное пространство с целью снижения размерности использован t-SNE – стохастическое вложение соседей с *t*-распределением [9, 10] с параметром perplexity, равным 45.

Кроме крупных групп, близких к определенным вызовам, можно также видеть и меньшие (например, такие как на рис. 3) группы, близость для одной из них также можно наблюдать на рис. 1.

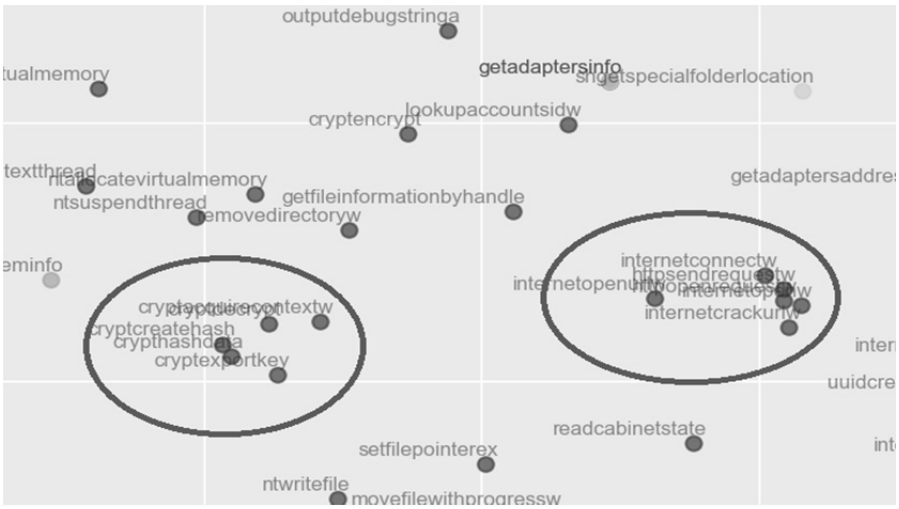

Рис. 3. Кластеры API-вызовов

Fig. 3. API call clusters

Область применения приведенных на рис. 3 API-вызовов достаточно компактна, и это отражено в полученных представлениях, однако в некоторых особых случаях компактно отображаются и слабо связанные по описанию вызовы. Причины такого неожиданного поведения еще предстоит выяснить.

3. ПОИСК УСТОЙЧИВЫХ СОЧЕТАНИЙ API-ВЫЗОВОВ

Помимо важности единичного значения каждого конкретного API-вызова, их совокупности могут нести значительное количество информации о протекающем процессе.

В общем случае при выделении из текста всех возможных значений пар и троек можно получить биграммы, триграммы и далее *n*-граммы соответственно [11]. Однако большая часть таких значений не будет иметь никакого смысла, так как они могут встречаться лишь однажды или случайным образом. В таком случае имеет смысл выделять устойчивые словосочетания, такие словосочетания называются коллокациями [12–14]. Обнаружение таких значений позволит решать сразу две задачи. Во-первых, обнаруживать значения,

относящиеся непосредственно к работе вредоносной нагрузки, и обнаруживать попытки зашумления последовательности в случае, если для такого зашумления API-вызовы слабо рандомизированы или имеют неслучайный характер. Кроме того, возможно обнаружение последовательностей, принадлежащих определенному классу вредоносного ПО с большей вероятностью, чем к другому, или часто встречающихся во вредоносных программах определенного типа.

После выполнения первоначального поиска *n*-грамм, результат которого изображен на рис. 4, было выявлено, что часть обнаруженных *n*-грамм может часто встречаться в последовательностях, однако количество таких последовательностей по отношению к общей выборке достаточно мало. На рис. 4 биграммы показаны на оси ординат, а на оси абсцисс отмечено количество вхождений этих биграмм в API-последовательности.

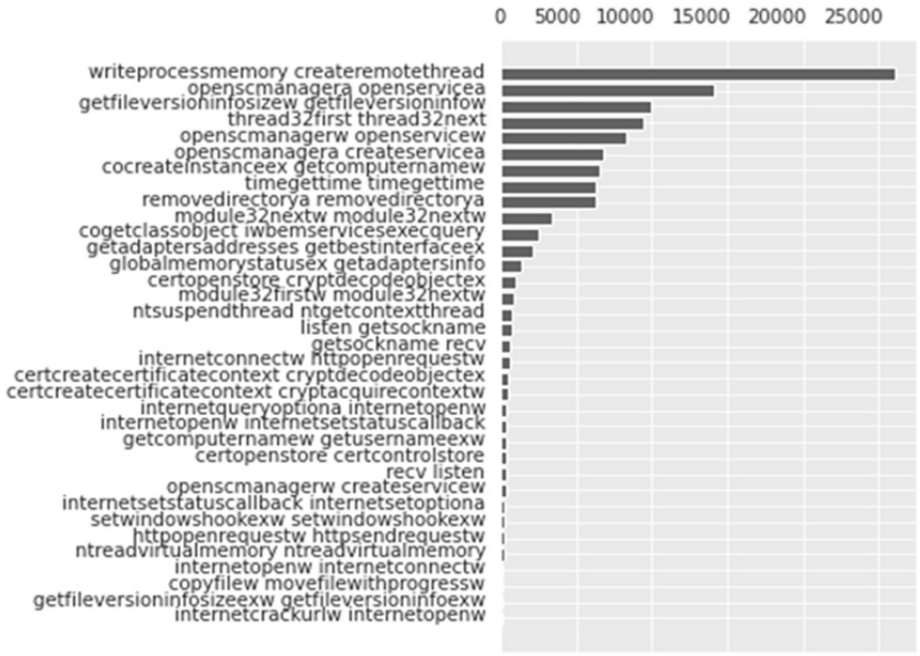

Рис. 4. Биграммы, выделенные из API-последовательностей

Fig. 4. Bigrams extracted from the API sequence

Так, в левой части рис. 5 для диаграммы, соответствующей биграмме *writeprocessmemory createremotethread*, было обнаружено 26 139 вхождений для 333 образцов. Для диаграммы в правой части рис. 5 было обнаружено 12 260 вхождений для 56 образцов. На данной диаграмме отображены: ось *x* в процентах от общего количества образцов данного типа, ось *y* – образцы данного типа. В этих и подобных им случаях такие биграммы представляли собой много раз повторяющуюся последовательность API-вызовов, соответствующих биграмме.

Такое поведение мешает обнаруживать биграммы, соответствующие реальным действиям, и, вероятно, представляет собой попытки зашумления процесса исполнения программы. Поэтому подобные *n*-граммы исключены

из последовательностей, и процесс поиска *n*-грамм был повторен снова, его результаты представлены на рис. 6, где на оси абсцисс показано количество вхождений *n*-грамм в последовательности, а на оси ординат – сама биграмма.

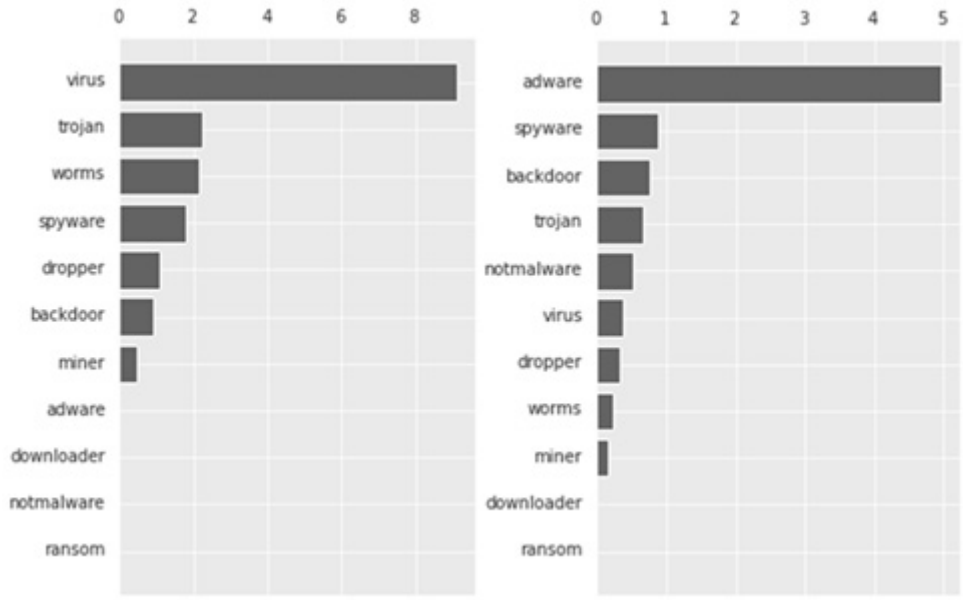

Рис. 5. Распределение биграмм writeprocessmemory createremotethread (слева) и timegettime timegettime (справа) по типам последовательностей

Fig. 5. Distribution of bigrams writeprocessmemory createremotethread (left) and timegettime timegettime (right) by sequence types

Аналогичные действия выполнены для последовательностей из трех и четырех вызовов.

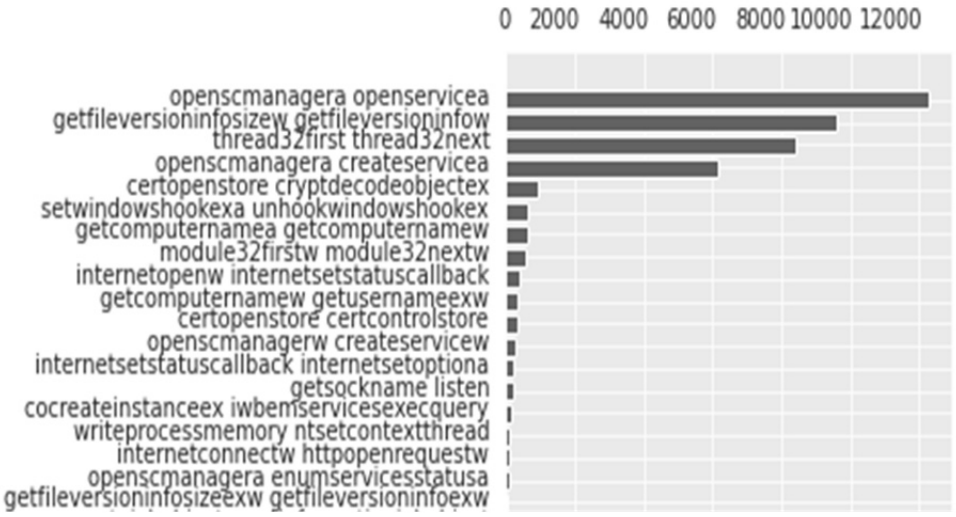

Рис. 6. Выделенные биграммы после очистки данных

Fig. 6. Dedicated bigrams after data clearing

На рис. 7 и 8 приведены частоты вхождения биграмм (ось ординат) в API-последовательностях определенного типа в процентах от общего количества (ось абсцисс).

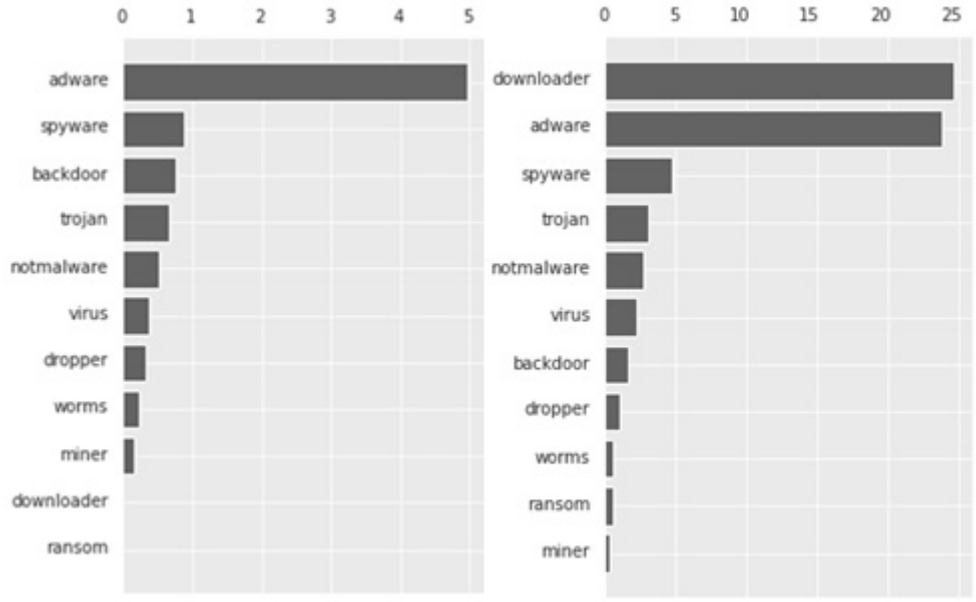

Рис. 7. Частота вхождения биграмм openscmanagera createservicea (слева) и internetopenw internetsetstatuscallback (справа)

Fig. 7. Frequency of occurrence of bigrams openscmanagera createservicea (left) and internetopenw internetsetstatuscallback (right)

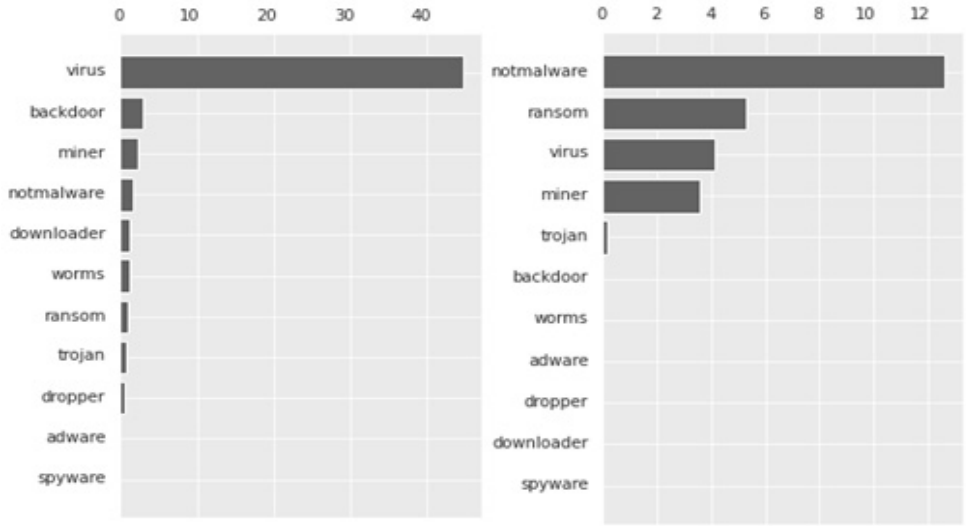

Рис. 8. Частота вхождения биграмм thread32first thread32next (слева) и getcomputernamea getcomputernamew (справа)

Fig. 8. Frequency of occurrence of bigrams thread32first thread32next (left) and getcomputernamea getcomputernamew (right)

Выделенные *n*-граммы включены в исходный текст последовательностей, для этого выполнена замена соответствующих последовательностей на их *n*-граммы.

В дальнейших исследованиях преобразованные совокупности воспринимаются как отдельные фразы, и они не рассматриваются в тексте по отдельности, при этом каждое слово, включенное в *n*-граммы, но находящееся в тексте отдельно от своей *n*-граммы, не будет заменено или удалено.

Далее по технологии проводится повторная кластеризация с учетом полученных *n*-грамм. Результат повторной кластеризации представлен на рис. 9.

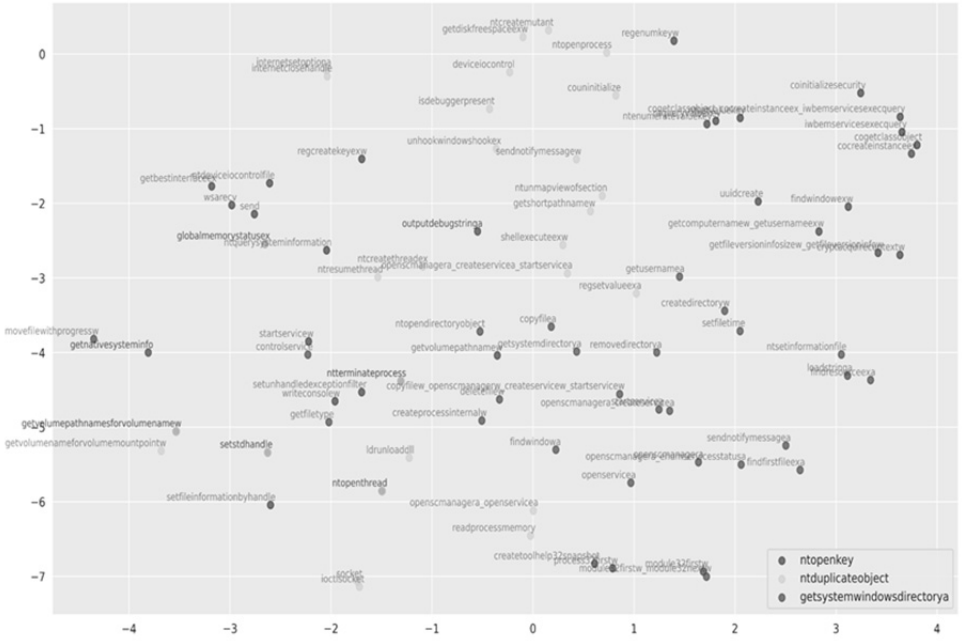

Рис. 9. Кластеризация отдельных API- вызовов с учетом *n*-грамм

Fig. 9. Clustering individual API calls based on *n*-grams

После выделения *n*-грамм в API-последовательностях появилась возможность выделения дополнительных кластеров, содержащих *n*-граммы, и рассмотрения их близости к отдельным API-вызовам. Кроме того, при добавлении API-вызовов оказывается возможным исследовать поведение значимых фрагментов последовательностей в контексте их групп.

4. ТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ API-ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Кроме группирования отдельных слов, методы обработки текстов позволяют группировать последовательности и извлекать из них темы [15]. Для обнаружения тем авторы статьи предлагают использовать алгоритм LDA (Latent Dirichlet Allocation) [16, 17].

В результате применения данного алгоритма в тексте API-последовательности обнаруживается заданное количество тем, и каждый документ (последовательность) можно интерпретировать как совокупность этих тем. Кроме того, для каждой темы выделяются ключевые слова этой темы, а также вес каждого слова в теме.

Для определения эффективного количества тем данного набора текстов выполняется циклическое создание моделей, подсчет когерентности для выбранной модели, сохранение модели и последующий выбор количества тем на основе максимального значения когерентности среди проверенных моделей. На рис. 10 приведены значения когерентности для используемого набора данных при проверке значения когерентности в пределах от двух до девяти.

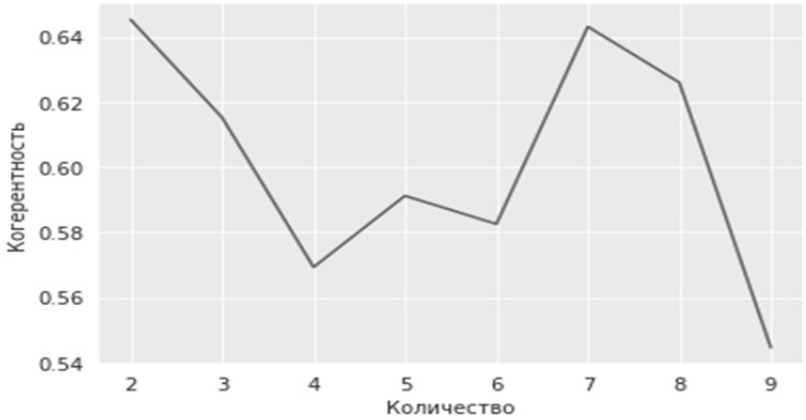

Рис. 10. Когерентность в зависимости от количества тем

Fig. 10. Coherence depending on the number of topics

Полученные результаты проанализированы и представлены на рис. 10. Эффективное количество тем равно семи (ось *x*). Выбор осуществлен на основании максимального значения когерентности (ось *y*).

Ниже для выбранной модели приведены первые 5 слов, каждое из которых представляет собой API-вызов, выделенный из последовательностей и имеющий большее влияние в рамках данной темы. Они расположены в порядке убывания степени значимости слова для этой темы:

1) ntwritefile, ntreadfile, ntfreevirtualmemory, ntallocatevirtualmemory, ldrgetprocedureaddress;

2) getfileattributesw, getsystemmetrics, getcursorpos, getkeystate, loadresource;

3) ntqueryvaluekey, ntopenkeyex, ntquerykey, ntopenkey, ldrgetprocedureaddress, ntreadfile;

4) ntquerydirectoryfile, findfirstfileexw, ntopenmutant, ntdeviceiocontrolfile, getshortpathnamew;

5) getfiletype, ntopenprocess, ntcreatefile, ldrloaddll, ntduplicateobject;

6) findfirstfileexw, regclosekey, getfileattributesw, regopenkeyexw, regqueryvalueexw;

7) regclosekey, regopenkeyexw, regqueryvalueexw, ldrgetprocedureaddress, ldrloaddll.

Каждый API-вызов и его положение относительно других в рамках данной темы имеет важное значение, так как в значительной степени определяет содержание темы.

Однако текст зачастую не имеет однозначной принадлежности к определенной теме, в них может присутствовать несколько тем в том или ином объеме. Пример подобной ситуации приведен в таблице.

Темы для различных образцов

Topics for various samples

Номер образца	Основная тема	Все темы с процентным вкладом
1	6	1 – 44 %; 6 – 52 %; 7 – 3 %
7	7	3 – 1 %; 5 – 5 %; 7 – 93 %
9	4	1 – 7 %; 2 – 16 %; 4 – 69 %; 5 – 4 %; 6 – 2 %

В данном случае в API-последовательности возможно наличие как одной основной группы ключевых слов, которая позволяет определить ее принадлежность (строка 2 таблицы), так и нескольких групп, степень воздействия которых на общую тему последовательности имеет схожий характер (строка 1). Кроме того, возможны случаи, когда последовательность имеет множество тем, каждая из которых влияет на ее глобальное значение (строка 3).

Следующий результат тематического моделирования представлен на рис. 11. Здесь приведено количество образцов (ось *y*), в которых тема под определенным номером (ось *x*) является основной, т. е. имеет больший вклад для данного образца. Общие данные по количеству вхождений определенной темы среди образцов представлены на рис. 12.

Рис. 11. Количество вхождений основной темы в образцах

Fig. 11. Number of occurrences of the main theme in samples

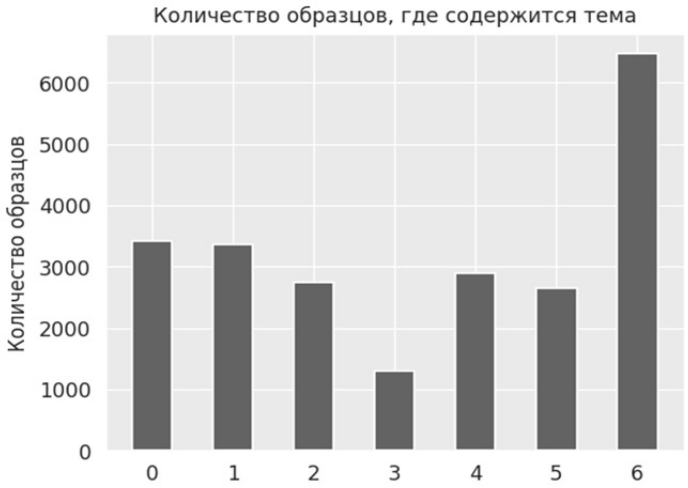

Рис. 12. Количество образцов, содержащих тему

Fig. 12. Number of samples containing a topic

Заключительный результат тематического моделирования представлен на рис. 13. Здесь иллюстрируется кластеризация по полученным темам. Данные образцов, в которых часть тем отсутствует, дополнены до соответствующей размерности.

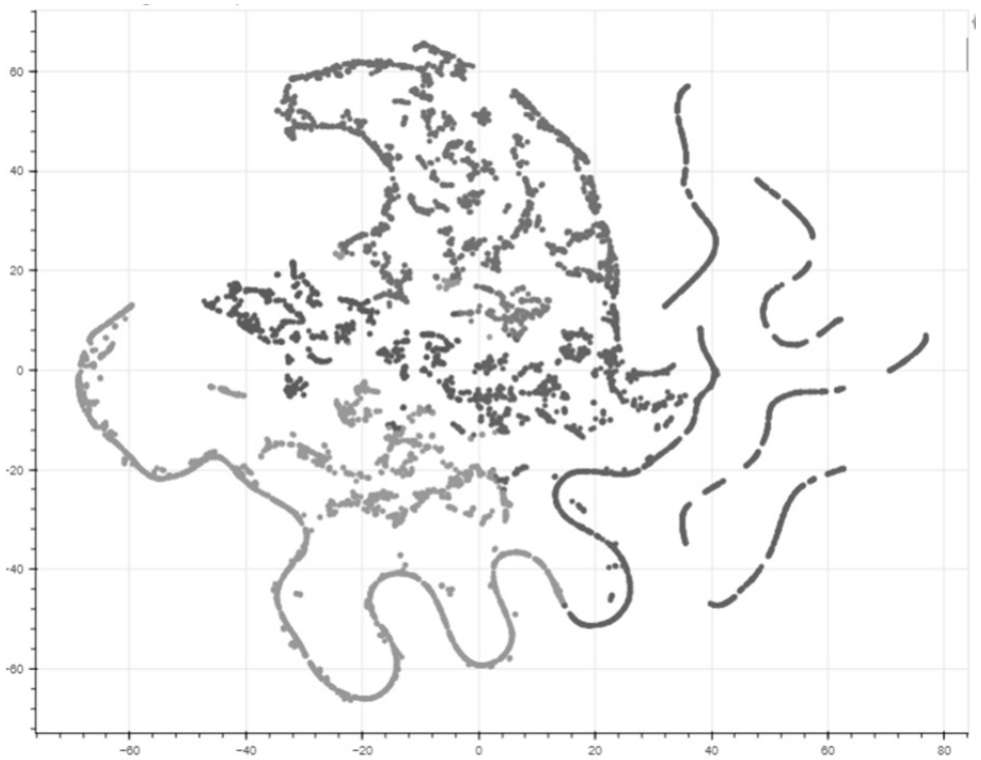

Рис. 13. Кластеризация полученных тем

Fig. 13. Clustering of received topics

В ходе кластеризации по темам сформированы кластеры на основе влияния выделенных тем. Такая обработка необходима и может дать больше информации о полученном результате тематического анализа, чем простое выделение главной темы. Причина этому факту – близость некоторых результатов (например, в строке 1 таблицы, где влияние тем имеет близкое значение). Кластеризация, показанная на рис. 13, говорит о группировании результата выделения тем из образцов. Такие группы формируются исходя из наличия множеств образцов, включающих в себя схожие комбинации выделенных тем. А это может говорить, например, о принадлежности к определенному семейству программ или использовании схожих механизмов работы. Более детальный анализ сгруппированных последовательностей позволит в будущем уточнить информацию о их содержании.

ЗАКЛЮЧЕНИЕ

Результаты, полученные при применении методов NLP к задаче анализа данных API-вызовов, позволяют фиксировать дополнительные данные о семантической связи между различными API-вызовами, а это является основой для дальнейшего совершенствования методов обнаружения вредоносного ПО при помощи нейронных сетей.

В процессе машинных экспериментов установлено, что применение таких методов помогает совершенствовать технологию формирования групп API-вызовов, а также позволяет обнаруживать пересекающиеся классы вредоносного ПО. Последнее, в свою очередь, обеспечивает повышение точности определения типа вредоносноых программ.

Выделенные н-граммы позволяют обнаруживать не только потенциальные совокупности API-вызовов, имеющих весомое значение для исполняемой программы, но и обнаруживать потенциальные попытки зашумления последовательности исполнения.

Тематическое моделирование дает возможность выделять группы ключевых слов для набора данных и обеспечивает возможность фиксировать принадлежность определенной последовательности к такой группе, а также позволяет идентифицировать группы действий.

Кроме того, выявлена необходимость в получении дополнительных данных о последовательности исполнения программ, например, таких как аргументы вызовов, возвращаемые значения, статус исполнения и др. В доступных на данный момент наборах данных такая информация отсутствует. Эта информация помогает идентифицировать связанные между собой части API-последовательностей.

СПИСОК ЛИТЕРАТУРЫ

1. *Воронин В.В., Морозов А.В.* Методика контроля угроз безопасности вредоносного программного обеспечения // Информатика и системы управления. – 2020. – № 3 (65). – С. 3–13.

2. *Bender E.M.* 100 things you always wanted to know about semantics & pragmatics but were afraid to ask. – Melbourne, 2018. – URL: http://faculty.washington.edu/ebender/papers/Bender-ACL2018-tutorial.pdf (accessed: 30.08.2021).

3. A neural probabilistic language model / Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin // Journal of Machine Learning Research. – 2003. – Vol. 3. – P. 1137–1155.

4. Neural network methods for natural language processing / Y. Goldberg, G. Hirst, Y. Liu, M. Zhang // Computational Linguistics. – 2018. – Vol. 44. – P.193–195.

5. Efficient estimation of word representations in vector space / T. Mikolov, K. Chen, G. Corrado, J. Dean. – arXiv: 1301.3781v3 [cs.CL]. – 2013.

6. *Hinton G., Roweis S.* Stochastic neighbor embedding // Proceedings of the 15th International Conference on Neural Information Processing Systems. – Cambridge, MA: MIT Press, 2002. – P. 857–864.

7. *Alammar J.* The illustrated Word2vec. – 2019. – URL: https://jalammar.github.io/illustrated-word2vec/ (accessed: 30.08.2021).

8. *Воронин В.В., Морозов А.В.* Методы NLP в задачах анализа последовательностей API вызовов // Информационные технологии XXI века. – Хабаровск: Изд-во ТОГУ, 2021. – С. 100–104.

9. *Maaten L. van der., Hinton G.* Visualizing data using t-SNE // Journal of Machine Learning Research. – 2008. – Vol. 9. – P. 2579–2605.

10. *Wattenberg M., Viégas F., Johnson I.* How to use t-SNE effectively // Distill. – 2016. – DOI: 10.23915/distill.00002.

11. *Sunny Srinidhi*. Understanding word N-grams and N-gram probability in natural language processing. – 2019. – URL: https://towardsdatascience.com/understanding-word-n-grams-and-n-gram-probability-in-natural-language-processing-9d9eef0fa058 (accessed: 30.08.2021).

12. *Bouma G.* Normalized (pointwise) mutual information in collocation extraction // Biennial GSCL Conference. – Tübingen, 2009. – P. 31–40.

13. *Милославский Е.С.* Семантический анализ текста как средство лучшего понимания смысла // Интеллектуальные технологии и средства реабилитации и абилитации людей с ограниченными возможностями (ИТСР-2018): труды III международной конференции. – М., 2018. – С. 255–258.

14. *Боровых К.О., Плотников А.В.* Семантический анализ текстов на примере интернет-запросов // Российский экономический интернет-журнал. – 2018. – № 2. – С. 1–14.

15. *Попов М.Ю.* Компьютерная обработка текста: визуализация семантической структуры и реферирование // Известия Волгоградского государственного технического университета. – 2004. – № 5. – С. 66–70.

16. *Исмагулов Т.С., Канева О.Н.* Методы тематического моделирования текстов на естественном языке // Информационный бюллетень ОМГТУ и ИМ СО РАН в области математики и информатики. – Омск, 2019. – Т. 3, № 1. – С. 108–111.

17. *Захарова А.А., Махныткина О.В.* Кластеризация текстовых документов с учетом семантической информации // Альманах научных работ молодых ученых Университета ИТМО. – СПб., 2020. – Т. 3. – С. 90–93.

Воронин Владимир Викторович, доктор технических наук, профессор кафедры «Автоматика и системотехника» Тихоокеанского государственного университета. Основное направление научных исследований – анализ и синтез концептуальных диагностических моделей различных типов объектов диагностирования. Имеет более 200 печатных работ и учебных пособий. E-mail: 004183vvv@mail.ru

Морозов Алексей Владимирович, магистр по направлению «Информационные системы и технологии», аспирант кафедры «Автоматика и системотехника» Тихоокеанского государственного университета. Основное направление научных исследований – контроль угроз безопасности вредоносных программ на основе нейросетевой технологии. Имеет 5 печатных работ. E-mail: 2014102127@pnu.edu.ru

Voronin Vladimir V., Phd (eng), professor of the Department of Automation and Systems Engineering, Pacific National University. His research interests – analysis and synthesis of conceptual diagnostic models of various types of diagnostic objects. He has more than 200 publications and teaching manuals. E-mail: 004183vvv@mail.ru

Morozov Aleksey V., Master in direction of Information systems and technologies, postgraduate student of the Department of Automation and Systems Engineering, Pacific National University. The main direction of scientific research is the control of malware threats based on neural network technology. Has 5 publications. E-mail: 2014102127@pnu.edu.ru

DOI: 10.17212/2782-2001-2021-3-37-52

Technology of key features identification in malware API calls sequences[*]

V.V. VORONIN[a], *A.V. MOROZOV*[b]

Pacific National University, 136 Tikhookeanskaya Street, Khabarovsk, 680035, Russian Federation

[a] *004183vvv@mail.ru* [b] *2014102127@pnu.edu.ru*

Abstract

Today, almost everyone is faced with computer security problems in one or another way. Antivirus programs are used to control threats to the security of malicious software.

Conventional methods for detecting malware are no longer effective enough; nowadays, neural networks and behavioral analysis technology have begun to be used for these purposes. Analyzing the behavior of programs is a difficult task, since there is no clear sequence of actions to accurately identify a program as malicious. In addition, such programs use measures to resist such detection, for example, noise masking the sequence of their work with meaningless actions. There is also the problem of uniquely identifying the class of malware due to the fact that malware can use similar methods, while being assigned to different classes. In this paper, it is proposed to use NLP methods, such as word embedding, and LDA in relation to the problems of analyzing malware API calls sequences in order to reveal the presence of semantic dependencies and assess the effectiveness of the application of these methods. The results obtained indicate the possibility of identifying the key features of malware behavior, which in the future will significantly improve the technology for detecting and identifying such programs.

Keywords: word embedding, Natural Language Processing, API calls, malware, Latent Dirichlet Allocation, n-grams, topic modeling, vector representation, clustering

REFERENCES

1. Voronin V.V., Morozov A.V. Metodika kontrolya ugroz bezopasnosti vredonosnogo programmnogo obespecheniya [Malicious software security threat control method]. *Informatika i sistemy upravleniya = Information Science and Control Systems*, 2020, no. 3 (65), pp. 3–13.

2. Bender E.M. *100 things you always wanted to know about semantics & pragmatics but were afraid to ask*. Melbourne, 2018. Available at: http://faculty.washington.edu/ebender/papers/Bender-ACL2018-tutorial.pdf (accessed 30.08.2021).

3. Bengio Y., Ducharme R., Vincent P., Jauvin C. A neural probabilistic language model. *Journal of Machine Learning Research*, 2003, vol. 3, pp. 1137–1155.

4. Goldberg Y., Hirst G., Liu Y., Zhang M. Neural network methods for natural language processing. *Computational Linguistics*, 2018, vol. 44, pp. 193–195.

5. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space. arXiv: 1301.3781v3 [cs.CL]. 2013.

6. Hinton G., Roweis S. Stochastic neighbor embedding. *Proceedings of the 15th International Conference on Neural Information Processing Systems*. Cambridge, MA, MIT Press, 2002, pp. 857–864.

[*] *Received 05 February 2021.*

7. Alammar J. *The illustrated Word2vec*. 2019. Available at: https://jalammar.github.io/illustrated-word2vec/ (accessed 30.08.2021).

8. Voronin V.V., Morozov A.V. Metody NLP v zadachakh analiza posledovatel'nostei API vyzovov [NLP methods for analysis of API call sequences]. *Informatsionnye tekhnologii XXI veka* [Information technology of the XXI century]. Khabarovsk, Pacific National University Publ., 2021, pp. 100–104.

9. Maaten L. van der., Hinton G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008, vol. 9, pp. 2579–2605.

10. Wattenberg M., Viégas F., Johnson I. How to use t-SNE effectively. *Distill*, 2016. DOI: 10.23915/distill.00002.

11. Sunny Srinidhi. Understanding word N-grams and N-gram probability in natural language processing. 2019. Available at: https://towardsdatascience.com/understanding-word-n-grams-and-n-gram-probability-in-natural-language-processing-9d9eef0fa058 (accessed 30.08.2021).

12. Bouma G. Normalized (pointwise) mutual information in collocation extraction. *Biennial GSCL Conference*, Tübingen, 2009, pp. 31–40.

13. Miloslavskiy E.S. [Semantican analysis of literary text in accordance with better understanding of its meaning]. *Intellektual'nye tekhnologii i sredstva reabilitatsii i abilitatsii lyudei s ogranichennymi vozmozhnostyami (ITSR-2018)* [III International Conference Intelligent technologies and means of rehabilitation of people with disabilities (ITSR-2018)], Moscow, 2018, pp. 255–258. (In Russian).

14. Borovykh K.O., Plotnikov A.V. Semanticheskii analiz tekstov na primere internet-zaprosov [Semantic text analysis on the example of internet queries]. *Rossiiskii ekonomicheskii internet-zhurnal = Russian economic online journal*, 2018, no. 2, pp. 1–14.

15. Popov M.Yu. Komp'yuternaya obrabotka teksta: vizualizatsiya semanticheskoi struktury i referirovanie [Computer text processing: visualization of semantic structure and abstracting]. *Izvestiya Volgogradskogo gosudarstvennogo tekhnicheskogo universiteta = Izvestia of Volgograd State Technical University*, 2004, no. 5, pp. 66–70.

16. Ismagulov T.S., Kaneva O.N. [Methods of thematic modeling texts in natural language]. *Informatsionnyi byulleten' OMGTU i IM SO RAN v oblasti matematiki i informatiki* [Information bulletin of the Omsk scientific and educational center of OmSTU and IM SB RAS in the field of mathematics and informatics]. Omsk, 2019, vol. 3, no. 1, pp. 108–111. (In Russian).

17. Zakharova A.A., Makhnytkina O.V. Klasterizatsiya tekstovykh dokumentov s uchetom semanticheskoi informatsii [Clustering text documents based on semantic information]. *Al'manakh nauchnykh rabot molodykh uchenykh Universiteta ITMO* [Almanac of scientific works of. young scientists of ITMO University]. St. Petersburg, 2020, vol. 3, pp. 90–93.

Для цитирования:

Воронин В.В., Морозов А.В. Технология идентификации ключевых особенностей в последовательностях API-вызовов вредоносных программ // Системы анализа и обработки данных. – 2021. – № 3 (83). – С. 37–52. – DOI: 10.17212/2782-2001-2021-3-37-52.

For citation:

Voronin V.V., Morozov A.V. Tekhnologiya identifikatsii klyuchevykh osobennostei v posledovatel'nostyakh API-vyzovov vredonosnykh programm [Technology of key features identification in malware API calls sequences]. *Sistemy analiza i obrabotki dannykh = Analysis and Data Processing Systems*, 2021, no. 3 (83), pp. 37–52. DOI: 10.17212/2782-2001-2021-3-37-52.