УДК 004.492

КЛАСТЕРИЗАЦИЯ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ МЕТАИНФОРМАЦИИ ВЫЗЫВАЕМЫХ СИСТЕМНЫХ ФУНКЦИЙ

Бабенко Л.К., Кириллов А.С.

Южный федеральный университет, Таганрог, e-mail: blk@tsure.ru, kirillovalexeys@gmail.com

Предложенный в работе метод кластеризации вредоносного программного обеспечения (ВПО) позволяет выполнять обнаружение не только новых образцов уже известных семейств ВПО, но и выделять не известные ранее семейства. С ростом компьютеризации множества сфер жизнедеятельности получают развитие и новые угрозы, в том числе вирусные. Своевременное обнаружение той или иной вирусной угрозы позволяет предотвратить множество негативных последствий ее реализации, однако решение данной задачи сопряжено с множеством трудностей. В данный момент достаточно широко представлены различные методы обнаружения новых образцов для уже известных семейств, что не полностью покрывает обозначенную проблему. В данной работе предлагается решить задачу обнаружения новых семейств вирусных угроз на основе дополнительной информации о вызове тех или иных системных функций с помощью алгоритма обучения без учителя. В результате предложенный метод позволяет достичь до 71 % лучшего результата чем похожие методы. Совпадение выделенных семейств с эталоном составило 85 %, при том что конкурентные методы, используя алгоритмы обучения с учителем, имеют средний результат 93 %.

Ключевые слова: поведенческий анализ, трассы вызовов, сегментация трассы, избыточные вызовы, обучение без учителя

CLUSTERING MALWARE ON THE BASIS OF METAINFORMATION OF CALLED SYSTEM FUNCTIONS

Babenko L.K., Kirillov A.S.

Southern Federal University, Taganrog, e-mail: blk@tsure.ru, kirillovalexeys@gmail.com

The proposed method of clustering malicious software allows you to perform detection of not only new samples of already known malware families, but also to detect unknown families. With the growth of computerization of many spheres of life activity, new threats, including viral ones, also develop. Timely detection of a particular virus threat can prevent many negative consequences of its implementation, but the solution of this problem involves a lot of difficulties. At the moment, various methods of detecting new samples for already known families are widely enough represented, which does not completely cover the indicated problem. In this paper, we propose to solve the problem of detecting new malware families based on additional information about the invocation of certain system functions using the learning algorithm without a teacher. As a result, the proposed method allows to achieve 71 % better results than similar methods. The match of the selected families with the reference was 85 %, while competitive methods using learning algorithms with the teacher have an average score of 93 %.

Keywords: behavioral analysis, call traces, trace segmentation, redundant calls, unsupervised learning

На сегодняшний день существует множество антивирусных продуктов, однако вредоносное программное обеспечение (ВПО) продолжает успешно противостоять этим решениям, доказательством тому являются периодические вспышки заражений различными типами ВПО, например некоторое время назад широкое распространение имели банковские трояны, сейчас же на сотни процентов в год растет число заражений программами-вымогателями [1], последствиями работы которых может быть полная потеря данных. Очевидно, что вместе с ростом количества заражений растет и количество семейств подобного ПО, своевременное выявление того или иного семейства позволяет составлять разного рода сигнатуры для оперативного выявления уже на стороне пользователя или межсетевого экрана, таким образом, обезвреживая угрозу до того, как она получит широкое распространение. Антивирусные компании для решения проблемы выявления новых семейств используют исключительно свои и закрытые системы, подтверждением тому может служить факт использования различных вариантов классификации образцов, а также часто полная закрытость подробной информации об обозначенном антивирусом классе. В открытой печати, безусловно, представлены работы по данной тематике, однако, как правило, авторы предлагают выполнять обнаружение новых образцов для уже известных семейств путем классификации, что не полностью решает обозначенную проблему, так как упускаются из вида образцы новых, не известных на момент обучения классификатора семейств. Для того чтобы успешно обнаруживать образцы неизвестных семейств и таким образом выявлять неизвестные ранее семейства образцов ВПО, требуется выполнить кластеризацию, которая относится к классу за-

дач обучения без учителя. В данной работе предлагается метод для кластеризации образцов ВПО на основе метаинформации вызываемых системных функций.

Современное состояние исследований

Подавляющее число представленных в открытой печати методов основываются на поведенческом анализе, который предполагает выполнение образца ВПО в специальной среде, которая может осуществить сбор различных метрик об этом образце. Акцент внимания исследователей смещен в пользу именно поведенческого анализа по нескольким основным причинам:

1. Возможность принципиального обхода техник обфускации, упаковки вредоносного кода.

2. Возможность получить данные о фактическом функционале образца.

Безусловно, у поведенческого анализа есть свои недостатки, главным из которых является сложность его реализации [2]. Поэтому исследователи часто предпочитают использовать готовые решения, коих, к счастью, достаточно много. Представленный в данной работе метод также основывается на результатах проведения поведенческого анализа. Кратко рассмотрим несколько современных методов, основанных на поведенческом анализе.

В работе [3] авторы делают предположение о том, что трасса вызовов по своему строению схожа с текстом, и, соответственно, выполняют обработку трассы методами из данной тематики, в частности рекуррентной и сверточной нейронной сетью. В работе [4] авторы из трассы вызовов формируют N-граммы и выполняют обработку с использованием нейронных сетей глубокого обучения. В работе [5] авторы предлагают формировать граф потоков данных на основе трассы вызовов, после чего выполнять сравнение графов образцов. В работе [6] авторы предлагают соотносить функции, присутствующие в трассе вызовов с категориями согласно их назначению и формировать последовательности из этих категорий, которые преобразуются с помощью функции хеширования ssdep, которая предполагает возможность сравнения близости каждого хеш-значения. В работе [7] авторы предлагают интегрированный метод, комбинирующий N-граммы из трассы вызовов и N-граммы, полученные на основе статического анализа. Полученные данные обрабатываются с помощью машины опорных векторов и классификатора случайных лес.

Из описаний представленных выше методов видно, что основное их отличие заключается в использовании различных математических методов и вариантов комбинации данных. Как было сказано выше, поведенческий анализ сложен в реализации, вследствие чего исследователи широко используют готовые решения, без должного внимания к особенностям работы анализатора и оставляя без внимания некоторые особенности работы программ так как анализатор в явном виде не предоставляет необходимых данных.

Описание метода

Предлагаемый в данной работе метод так же, как и многие, предполагает использовать трассы вызовов системных функций для последующей кластеризации. Однако есть несколько существенных отличий в части формирования трассы:

1) при формировании результирующей трассы вызовов учитывается метаинформация вызова, в частности данные о процессе и потоке, что позволяет повысить качество кластеризации до 16%, за счет отражения структуры образца в трассе;

2) при формировании результирующей трассы вызовов учитываются только вызовы из исполнимого модуля исследуемого образца, что также позволяет повысить качество кластеризации до 9%, за счет удаления избыточных и не имеющих непосредственного отношения к образцу вызовов из трассы, например вызовов, выполненных из используемых библиотек;

3) метод предполагает кластеризацию с неизвестным числом кластеров за счет использования алгоритма DBSCAN (Density-based spatial clustering of applications with noise), что позволяет обнаруживать новые образцы неизвестных семейств и, соответственно, выделять новые семейства образцов ВПО.

Совместное применение преобразований 1 и 2 дает улучшение качества кластеризации до 68%.

Возможность получения обозначенных результатов реализована за счет углубленного анализа десятков семейств ВПО, модификации системы поведенческого анализа, что позволило извлечь дополнительную информацию о работе исследуемых образцов, которая не доступна для референсных решений, которыми пользуются большинство исследователей.

Рассмотрим предлагаемый в данной работе метод подробнее. Общая структура процесса и трассы исследуемого образца и результат ее преобразования приведен на рис. 1.

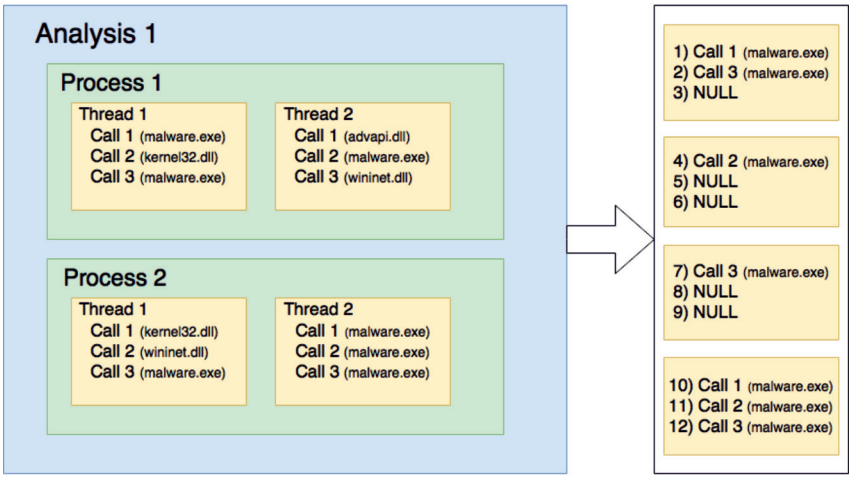

Рис. 1. Процесс и результаты преобразования трассы исследуемого образца

Первым этапом преобразования данных, полученных в результате поведенческого анализа, является удаление паразитных вызовов, не принадлежащих исполнимому модулю исследуемого образца, таким образом устраняя зашумляющие данные из трассы вызовов. На рисунке 1 данный этап выглядит как удаление вызова «Call1» в потоке 1 («Thread 1»), «Call 1» и «Call 3» в потоке 2 («Thread 2»), «Call 1» и «Call 2» в потоке 3 («Thread 3»). Данный этап очень важен в связи с тем, что любая программа, в том числе вредоносная, в своей работе использует высокоуровневые функции, которые предоставляют системные библиотеки и вызов такой функции сопряжен с вызовом более низкоуровневых функций самой библиотекой для реализации запрашиваемого функционала. Например, при использовании библиотеки WinInet для обращения к ресурсам интернета, данная библиотека выполняет чтение и последующую запись файлов cookie, извлечения настроек прокси из реестра, вызов функций библиотеки WinHttp, которая в свою очередь использует функции библиотеки Ws2_32 для сетевого взаимодействия. Очевидно, что в трассе вызовов следует оставлять только те функции, которые были вызваны непосредственно исполнимым модулем образца, таким образом, отражая функциональные особенности именно образца, а не используемых им библиотек.

Вторым этапом преобразования данных является предварительное формирование трассы для каждого потока процесса отдельно. На рис. 1 данный этап представлен сохранением последовательности внутри блоков «Thread 1», «Thread 2», «Thread 3», «Thread 4» и их последовательным объединением. Операционные системы, в том числе Windows, как правило, не гарантируют, что новый поток в процессе создастся в какой-то конкретный момент, соответственно, невозможно зафиксировать место в трассе, где появится вызов из нового потока, более того, с каждым новым запуском образца это место может быть разным. Ситуация еще больше усложняется, если имеется несколько потоков в нескольких процессах. Таким образом, игнорирование данной проблемы влечет за собой получение совершенно разных трасс при каждом запуске для экземпляров, с множеством процессов и потоков, что, в свою очередь, ставит под вопрос результаты дальнейшей обработки данных. Для того чтобы избежать негативного влияния данного факта на последующую обработку данных, следует выполнять сегментирование трассы согласно потоку и процессу, в которых выполняются вызовы контролируемых функций. Еще одним позитивным моментом такого подхода является фактически отражение структуры образца, что несомненно положительно сказывается на результатах дальнейшей обработки данных.

После формирования трассы для каждого потока выполняется удаление последовательно повторяющихся, но более двух раз вызовов. То есть предполагается, что последовательных вызовов одной и той же функции в трассе может быть не более двух, данное допущение вводится с несколькими целями:

1) устранить избыточность, которая в последующем может негативно повлиять на дальнейшую кластеризацию. Злоумышленник может сознательно внедрять вызовы произвольных функций с целью «зашумить» трассу;

2) учесть особенности функционирования образца, так как повторный вызов одной и той же системной функции может быть вполне легитимным, например функция WINAPI ConvertBinaryToString вызывается первый раз для получения размера буфера необходимого для хранения выходных данных, а второй раз – для непосредственного преобразования данных с помещением их в предварительно выделенный буфер.

Третьим этапом преобразования данных является последовательное слияние трасс вызовов каждого потока. Для того, чтобы такое слияние действительно отражало структуру образца и единообразно характеризовало каждый образец, вводятся некоторые ограничения (выравнивание), а именно:

- 50 вызовов в одном потоке, в случае недостатка данных выполняется дополнение последовательностью 0 векторов;
- 2 потока в каждом процессе, в случае недостатка выполняется дополнение потоком из последовательности 0 векторов;
- 2 процесса, в случае недостатка выполняется дополнение согласно описанным ранее правилам.

Данные ограничения основаны на статистике выборки исследуемых образцов, в частности на среднем значении для каждого из них по всей выборке вредоносного ПО (данные о выборке представлены ниже). В результате получается вектор вызываемых функций длиной 200. Данный этап представлен на рис. 1 в виде «NULL» значений, которыми заполняются недостающие вызовы в трассе потока, в частности в позициях 3, 5, 6, 8, 9.

Четвертым этапом преобразования данных является кодирование вызовов в полученной трассе. Полученная трасса содержит имена вызываемых функций, что является категориальными признаками, работать с которыми может ограниченное количество алгоритмов, соответственно, требуется ввести кодирование имен функций. В качестве такого кодирования предлагается использовать метод OneHot, он является широко распространённым методом для обработки данных и выполняет кодирование каждого признака единичным битом в векторе, чья длина соответствует количеству возможных признаков, в нашем случае количеству контролируемых функций. Общее количество функций из 438 перехватываемых, вызовы которых зафиксированы в ходе поведенческого анализа, равно 288. Таким образом, каждый вызов представляется бинарным вектором длиной 288, что влечет за собой следующие сложности:

- усложняет дальнейшую обработку данных в части поиска подходящего алгоритма;
- увеличивает вычислительную сложность;
- усложняет визуализацию данных;
- увеличивает требование к объему хранилища данных.

Пятый этап преобразования данных заключается в решении обозначенных проблем, для чего было принято решение сократить размерность данного вектора до удобоваримого значения. Для решения данной задачи существует целый класс алгоритмов как на основе популярных нейронных сетей, так и на основе статистических методов. Из всех представленных методов снижения размерности был выбран метод главных компонент (PCA).

Основным критерием для выбора метода снижения размерности была именно гибкость в выходных данных и возможность оценки результатов с учетом того, что данные, которые подвергаются обработке, имеют линейный характер. Метод главных компонент позволяет не только задать необходимую размерность выходного вектора признаков, но и оценить, насколько точно заданная размерность описывает исходный набор данных.

Как результат выполнения данного этапа бинарный вектор длиной 288 преобразуется в вектор одинарной длины, средняя объяснительная дисперсия которого составляет 0,8, то есть 80 % вариаций данных описывается первой главной компонентой. Соответственно, полная трасса преобразуется в вектор действительных чисел длиной 200. Что позволяет использовать для дальнейшей обработки массу возможных алгоритмов.

После того, как данные были преобразованы в подходящий вид, следует этап непосредственно кластеризации. Большая часть представленных алгоритмов кластеризации предполагает задание в качестве параметра количества кластеров, однако в данном случае количество кластеров не известно достоверно. Безусловно, существуют различные алгоритмы, которые позволяют оценить предполагаемое количество кластеров и использовать это значение в дальнейшем. Однако данный подход не до конца корректен, так как его цель разрешить принципиальный недостаток соответствующих алгоритмов и подобрать оптимальное значение количества кластеров, что в данном случае неприемлемо, так как кластер должен четко представлять определенное семейство образов ВПО. Таким образом, требуется использовать ал-

горитм, который изначально предполагает самостоятельное разбиение на кластеры, в качестве такого алгоритма был выбран DBSCAN (Density-based spatial clustering of applications with noise, плотностной алгоритм пространственной кластеризации с присутствием шума) [8]. Также в качестве достоинств данного алгоритма стоит отметить корректную работу с несимметричными выбросами, способность обнаруживать кластеры разной размерности.

Тестовая выборка

Важнейшим этапом, предшествующим проведению экспериментального исследования, является составление репрезентативной тестовой выборки. Для решения этой задачи использовались материалы с сайта virusshare.com, где можно получить архивы ВПО, обнаруженного за последнее время в различных источниках. Месячный архив насчитывает порядка 120000 экземпляров. Из данного архива были случайно выбраны более 5000 экземпляров и проанализированы с использованием тестовой среды, в результате было получено 5198 успешно проанализированных экземпляра, которые были использованы в ходе экспериментальных исследований.

Экспериментальные исследования

Как уже было сказано выше, перед тем как провести кластеризацию образцов, следует выполнить предварительную обработку данных методом главных компонент. Для того, чтобы получить вектор данных, удобный для дальнейшей работы, было принято решение что для каждого вектора, представляющего вызов, должна быть вычислена только первая главная компонента. Среднее значение объяснительной дисперсии для первой компоненты составило 0,8. Если исключить из процесса формирования результирующей трассы предварительную сегментацию по потокам и процессам, среднее значение объяснительной дисперсии соответствует 0,6. Средняя объяснительная дисперсия для трассы построенной с использованием сегментации вызовов согласно процессу и потоку, но без выделения вызовов модуля исследуемого образца, составила 0,65. Объяснительная дисперсия для случая, когда в трассе не было сегментации и присутствовали не только вызовы модуля исследуемого образца, составила 0,55.

Проведенные эксперименты говорят о том, что выделение вызовов именно модуля исследуемого образца и сегментация трассы позволяет существенно повысить качество дальнейшей кластеризации в частности до 68 %. Результаты описанных экспериментов сведены в табл. 1.

Следующим этапом обработки является непосредственно кластеризация. Так же как для метода главных компонент, процесс кластеризации был выполнен для 4 вариантов преобразований трассы вызовов. Результаты данных экспериментов приведены в табл. 2.

Таблица 1

Результаты работы метода главных компонент в зависимости от преобразований трассы вызовов

№ п/п	Преобразования трассы вызовов	Объясненная дисперсия
1	Сегментация трассы + вызовы только образца	0,8
2	Вызовы только образца	0,6
3	Сегментация трассы	0,65
4	Без преобразований	0,55

Таблица 2

Результаты работы метода кластеризации DBSCAN в зависимости от преобразований трассы вызовов

№ п/п	Преобразования трассы вызовов	Обнаружено классов	Кластеризовано	Не кластеризовано	Средний размер кластера
1	Сегментация трассы + вызовы только образца	480	3648	1550	7,6
2	Вызовы только образца	409	3565	1633	7,112
3	Сегментация трассы	514	3378	1820	6,57
4	Без преобразований	438	2542	2656	5,803

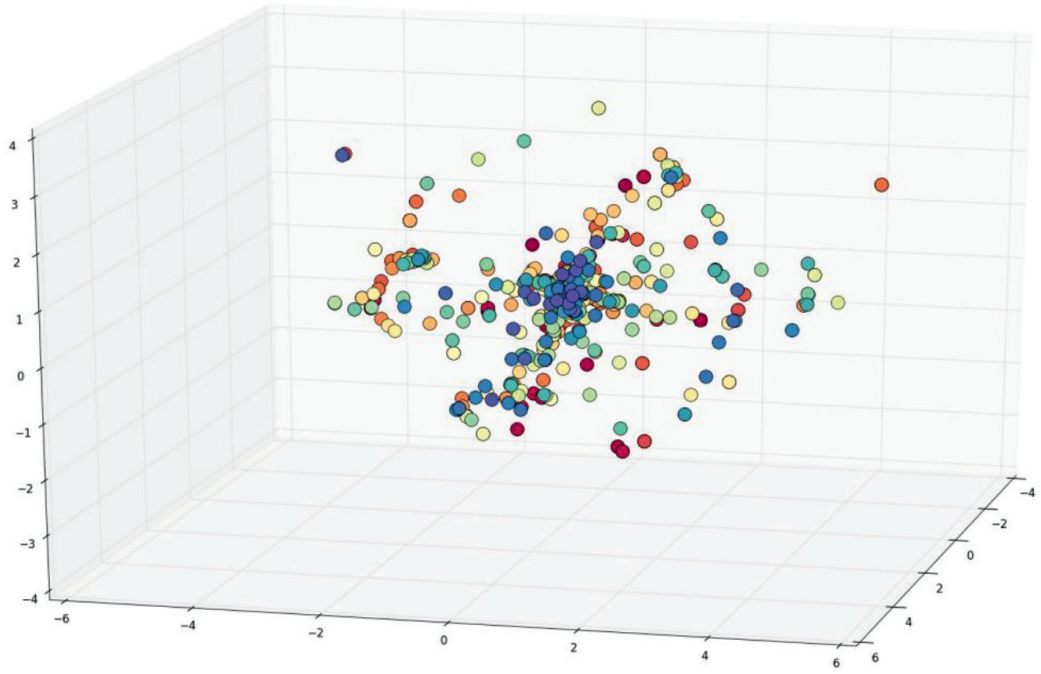

Рис. 2. Графическое представление результатов кластеризации

В результате проведенных экспериментов выявлено:

1) сегментация трассы позволяет лучше разграничивать кластеры между собой за счет учета структуры образца в трассе, о чем говорит самое низкое значение числа обнаруженных кластеров без ее использования, что говорит о том, что кластеры слились между собой;

2) выделение вызовов модуля исследуемого образца позволяет четче разграничивать кластеры, за счет удаления избыточных вызовов, не относящихся к модулю исследуемого образца, о чем говорит большое количество обнаруженных кластеров без учета данного преобразования, избыточные вызовы вносят шум, ведущий к распаду кластеров;

3) совокупное использование обоих преобразований трассы вызовов позволяет получить до 71% меньше некластеризованных образцов и выделить до 17% больше кластеров.

Для валидации результатов кластеризации применялись следующие методы:

1) вычисление коэффициента Silhouette, который составил 0,9943;

2) кроссвалидация методом K-средних (кол-во кластеров на данный момент известно), коэффициент схожести результатов составил 0,9949.

Для оставшихся 1550 некластеризованных образцов процедура кластеризации была запущена дополнительно, в результате чего новых кластеров найдено не было. Далее из начальной выборки были выбраны экземпляры, количество которых в кластере кратно двум и больше двух, после чего выборка была разбита на 2 части, и каждая из них была кластеризована. В результате сопоставления результатов кластеризации выявлено, что какого-либо слияния между другими кластерами не произошло. Каждый экземпляр оказался именно в том кластере, в котором он был ранее. Описанные эксперименты позволяют сказать, что выбранный алгоритм кластеризации дает стабильные результаты. Таким образом, метод позволяет выделять конкретные кластеры вне зависимости от характеристик выборки.

Графическое представление результатов кластеризации приведено на рис. 2.

Заключительным этапом проведения экспериментальных исследований стало сравнение разбиения на кластеры (то есть на семейства) образцов тестовой выборки с разбиением, предложенным популярным и стандартным антивирусом для Windows – Microsoft Essentials. Из 1150 образцов, для которых данный антивирус зафиксировал угрозу по данным сервиса virustotal.com,

совпадение кластеров с кластерами, полученными в результате использования предложенного метода, составило 85 %, без учета того факта, что в некоторых случаях фактически одно семейство имеет несколько названий. Например, kelihos так же называется wildrak. То есть в реальности результат может даже превышать обозначенную цифру.

Выводы

В данной работе предложен метод кластеризации ВПО, который показывает до 71 % лучше результат, чем конкурентные методы, за счет учета только системных вызовов из исполнимого модуля исследуемого образца и сегментации трассы вызовов согласно процессу или потоку. Общая эффективность метода соответствует высокому уровню, что подтверждается фактом совпадения выделенных семейств с, в данном случае, эталонными результатами антивируса Microsoft Essentials на 85 %. При этом метод, в отличие от конкурентных, позволяет выделять не только новые образцы известных семейств, но и выполнять выделение новых семейств вредоносного программного обеспечения за счет использования алгоритмов обучения без учителя.

Список литературы

1. Bhardwaj A. et al. Ransomware Digital Extortion: A Rising New Age Threat // Indian Journal of Science and Technology. – 2016. – T. 9. – P. 14.

2. Egele M. et al. A survey on automated dynamic malware-analysis techniques and tools // ACM Computing Surveys (CSUR). – 2012. – T. 44, № 2. – P. 6.

3. Tobiyama S. et al. Malware Detection with Deep Neural Network Using Process Behavior // Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual. – IEEE, 2016. – T. 2. – P. 577–582.

4. Kolosnjaji B. et al. Deep Learning for Classification of Malware System Call Sequences // Australasian Joint Conference on Artificial Intelligence. – Springer International Publishing, 2016. – P. 137–149.

5. Wuchner T., Ochoa M., Pretschner A. Malware detection with quantitative data flow graphs // Proceedings of the 9th ACM symposium on Information, computer and communications security. – ACM, 2014. – C. 271–282.

6. Gupta S., Sharma H., Kaur S. Malware Characterization Using Windows API Call Sequences // International Conference on Security, Privacy, and Applied Cryptography Engineering. – Springer International Publishing, 2016. – P. 271–280.

7. Shijo P.V., Salim A. Integrated static and dynamic analysis for malware detection // Procedia Computer Science. – 2015. – T. 46. – P. 804–811.

8. Ester M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise // Kdd. – 1996. – T. 96, № 34. – C. 226–231.