Ксения И. Салахутдинова [1], Владислав В. Малков[2], Ирина Е. Кривцова[3]
[1]Санкт-Петербургский институт информатики и автоматизации Российской академии наук, 14 линия, 39, г. Санкт-Петербург, 199178, Россия
[2, 3]Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Кронверкский проспект, 49, г. Санкт-Петербург, 197101, Россия
[1]e-mail: kainagr@mail.ru, https://orcid.org/0000-0001-9254-8652
[2]e-mail: vladislav.malkov@mail.ru, https://orcid.org/0000-0002-8479-6362
[3]e-mail: ikr@cit.ifmo.ru, https://orcid.org/0000-0003-2483-1637

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПОДХОДОВ К ИДЕНТИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

DOI: http://dx.doi.org/10.26583/bit.2019.2.04

Аннотация. Целью исследования ставится тестирование различных известных библиотек градиентного бустинга деревьев решений применительно к задаче идентификации программного обеспечения в условиях ограниченного набора исполняемых файлов различных версий одной программы в обучаемой выборке. Обосновывается важность аудита программного обеспечения для бизнес-процессов. В работе рассмотрены средства контроля установленного программного обеспечения на персональные компьютеры пользователей автоматизированных систем. Обоснованы недостатки таких программных решений с примерами обхода их алгоритмов идентификации программ и представлен разработанный подход по идентификации исполняемых файлов при помощи алгоритма машинного обучения – градиентный бустинг деревьев решений на основе библиотек XGBoost, LightGBM, CatBoost. Проведен эксперимент по идентификации исполняемых файлов с помощью XGBoost, LightGBM. На основе бикубической меры качества кластеризации был выполнен сравнительный анализ полученных результатов с предложенным авторами ранее подходом к идентификации программ на основе библиотеки CatBoost, а также с результатами, представленными в других исследованиях. Полученные результаты свидетельствуют, что разработанный подход позволяет выявить нарушения установленной политики безопасности при обработке информации в автоматизированных системах.

Ключевые слова: информационная безопасность, идентификация программ, машинное обучение, градиентный бустинг деревьев решений, XGBoost, LightGBM.
Для цитирования: САЛАХУТДИНОВА, Ксения И.; МАЛКОВ, Владислав В.; КРИВЦОВА, Ирина Е.. СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПОДХОДОВ К ИДЕНТИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. Безопасность информационных технологий, [S.l.], v. 26, n. 2, p. 58-66, 2019. ISSN 2074-7136. Доступно на: <https://bit.mephi.ru/index.php/bit/article/view/1199>. Дата доступа: 03 june 2019. doi:http://dx.doi.org/10.26583/bit.2019.2.04.

Kseniya I. Salakhutdinova [1], Vladislav V. Malkov [2], Irina E. Krivtsova [3]
[1]St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, 14-th Linia, VI, 39, St. Petersburg, 199178, Russia
[2, 3]ITMO University, Kronverksky avenue, 49, St. Petersburg, 191002, Russia
[1]e-mail: kainagr@mail.ru, https://orcid.org/0000-0001-9254-8652
[2]e-mail: vladislav.malkov@mail.ru, https://orcid.org/0000-0002-8479-6362
[3]e-mail: ikr@cit.ifmo.ru, https://orcid.org/0000-0003-2483-1637

A comparative analysis of software identifying approaches

DOI: http://dx.doi.org/10.26583/bit.2019.2.04

Abstract. The aim of the study is to provide a test of various well-known gradient boosted decision trees libraries, which are used here in relation to the software identification problem with limited set of executable files belonged to different versions of the same program in the training sample. The importance of software audit for business processes is substantiated. The paper considers the control means of installed software on personal computers of automated systems users. The disadvantages of such software solutions are substantiated with crawling examples for algorithms of program identification and the developed approach to the identification of executable files using the machine learning algorithm – gradient boosting of decision trees, based on the libraries XGBoost, LightGBM, CatBoost is presented. An experiment to identify executable files with the help of XGBoost, LightGBM is performed. On the basis of bicubic measure of clustering quality, a comparative analysis of the results between previously proposed program identification approach based on the CatBoost library, and the results presented in other studies, is performed. The results show that the developed approach allows identifying violations of the established security policy in automated systems information processing.

Keywords: information security, program identification, machine learning, gradient boosting of decision trees, XGBoost, LightGBM.

For citation: SALAKHUTDINOVA, Kseniya I.; MALKOV, Vladislav V.; KRIVTSOVA, Irina E.. A comparative analysis of software identifying approaches. IT Security (Russia), [S.l.], v. 26, n. 2, p. 58-66, 2019. ISSN 2074-7136. Available at: <https://bit.mephi.ru/index.php/bit/article/view/1199>. Date accessed: 03 june 2019. doi:http://dx.doi.org/10.26583/bit.2019.2.04.

Введение

Современную организацию, государственную или частную, уже невозможно представить без информационных технологий, которые не только позволяют автоматизировать ранее выполняемые вручную процессы, но также стали неотъемлемой частью функционирования самой организации. Важной составляющей в бизнесе является информационная безопасность. Известно, что в число наиболее вероятных инцидентов информационной безопасности входит нарушение установленной политики безопасности. Нарушение пользователями требований политики в сфере использования программного обеспечения, содержащего устаревшие версии или несанкционированно установленные программы, влечет к появлению уязвимости, в дальнейшем эксплуатируемой злоумышленником. Такое программное обеспечение способно содержать в себе дефекты, недекларированные возможности, а также может использоваться в целях получения личной выгоды или являться объектом чужих авторских прав. Не стоит забывать и о специальных программах, направленных на преодоление установленной защиты, либо противоправных действиях внутри сети Интранет или Интернет.

Зачастую использование одних только организационных мер защиты недостаточно, необходимо подкреплять их техническими мерами, например, в отношении задачи по контролю устанавливаемого программного обеспечения на электронные носители информации пользователями автоматизированных систем необходимо применение средств аудита программного обеспечения [1].

1. Средства контроля программного обеспечения

Множество компаний обращаются к использованию систем управления ИТ-активами (ITAM – IT asset management), представляющих собой комплексные решения, нацеленные на физический учёт, финансовый контроль и соблюдение контрактных обязательств, связанных с ИТ-активами, на протяжении всего их жизненного цикла. Здесь под ИТ-активами подразумеваются все аппаратные и программные элементы ИТ-

инфраструктуры, обеспечивающие деятельность бизнес-среды. В свою очередь ITAM подразделяется на управление аппаратными активами (HAM – Hardware Asset Management), охватывающее управление материальными составляющими ИТ-инфраструктуры: пользовательские компьютеры, серверы, телефоны и т.д.; и на управление программными активами (SAM – Software Asset Management), охватывающее управление нематериальными составляющими ИТ-инфраструктуры: программное обеспечение, лицензии, версии, конечные точки инсталляции и т.д. На рынке услуг представлено немало решений, позволяющих идентифицировать программные активы, управлять учетом, а также проводить контроль их изменений и др. Из числа наиболее известных программных продуктов можно выделить Microsoft Assessment and Planning Toolkit, Lansweeper, SAManage, AIDA64 Business Edition, Kaspersky Systems Management. Приведенные библиотеки способны проводить сбор информации как удаленно, находясь на сервере, так и при помощи внедряемого в персональные компьютеры пользователей агента. Большая часть предоставляемой информации собирается посредством следующих встроенных технологий инвентаризации программно-аппаратного окружения и операционной системы: Windows Management Instrumentation, Active Directory, Domain Services (AD DS), SMS Provider, lshw, dpkg -l и других технологий.

Недостатки такого подхода очевидны. При недостаточном уровне квалифицированности со стороны администратора системы, наличии способностей в сфере компьютерных технологий у пользователей автоматизированных систем или при низкоуровневом подходе руководства к подбору кадров – появляется возможность внести изменения в конфигурационные данные устанавливаемого программного обеспечения. На рисунках 1-3 приведены примеры по изменению версии программ (ABBYY Lingvo и Htop) в операционных системах Windows (10) и Linux (Ubuntu) соответственно, из которых становятся очевидными пути обмана средств аудита программного обеспечения как для установленного ранее программного обеспечения, так и для устанавливаемого впервые.

Рис. 1. Изменение версии программы ABBYY Lingvo x5 15.0.511 на ABBYY Lingvo x5 15.0.510
(Fig. 1. Change the program version from ABBYY Lingvo x5 15.0.511 to ABBYY Lingvo x515.0.510)

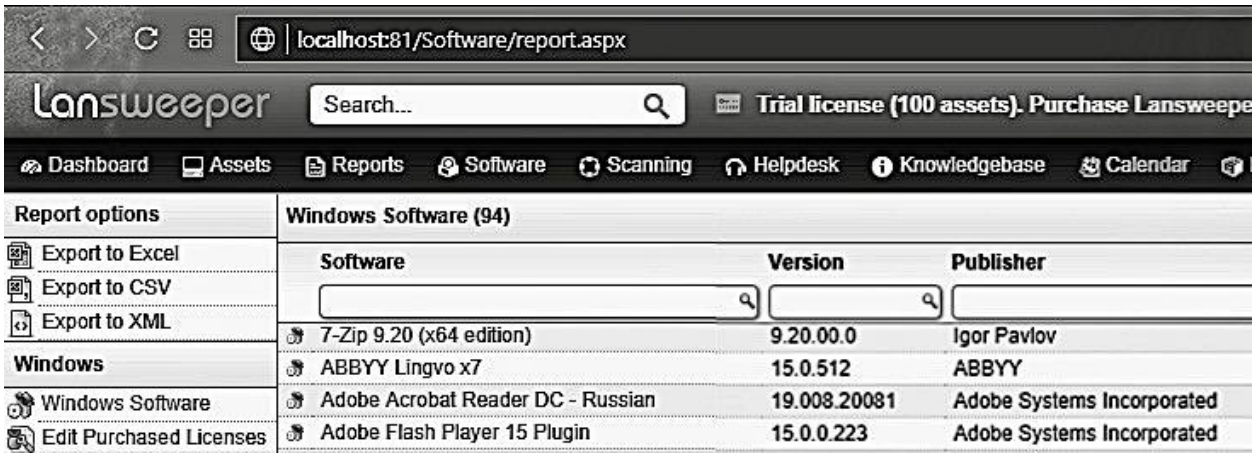

Рис. 2. Изменение версии программы ABBYY Lingvo x5 15.0.511 на ABBYY Lingvo x7 15.0.512
(Fig. 2. Change the program version from ABBYY Lingvo x5 15.0.511 to ABBYY Lingvo x7 15.0.512)

Из рис. 1 и 2 видно, что путем произведенных манипуляций в реестре операционной системы Windows версия программы ABBYY Lingvo была изменена дважды, целью данного вмешательства являлось намеренное влияние на результаты выдачи запроса по инвентаризации программного обеспечения с помощью средств Windows Management Instrumentation (WMI) (результат выдачи представлен на рис. 1) и распространенного программного средства Lansweeper (результат выдачи представлен на рис. 2).

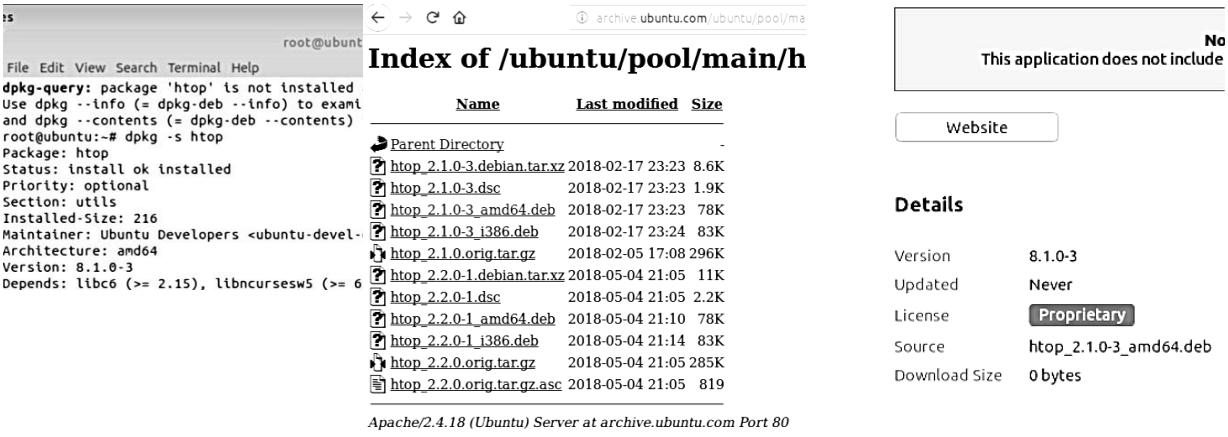

Рис. 3. Установка измененной версии программы htop 8.1.0-3 вместо htop 2.1.0-3
(Fig. 3. Installation a modified program version htop 8.1.0-3 instead htop 2.1.0-3)

На рис. 3 представлена установка программы Htop в операционной системе Ubuntu с заведомо измененной версией, исправленной путем вмешательства в конфигурационный файл устанавливаемого пакета .deb.

Ранее авторами в работах [2-4] был представлен подход к проведению аудита электронных носителей информации путем формирования частотных сигнатур побайтового или ассемблерного кода программ и были приведены различные методы идентификации программного обеспечения, т.е. определения схожести эталонных сигнатур санкционированных программ с сигнатурами идентифицируемых файлов. Так, в последней работе [5] был рассмотрен подход, основанный на машинном обучении, а именно алгоритме градиентного бустинга деревьев решений (gradient boosting trees) [6], реализованном в открытой библиотеке CatBoost от компании Яндекс [7].

В настоящей работе авторы ставят своей целью произвести сравнение нескольких средств реализаций градиентного бустинга применительно к задаче идентификации программного обеспечения и сравнить полученные результаты с подходами, рассматриваемыми другими исследователями.

2. Библиотеки градиентного бустинга деревьев решений

В работе рассмотрены следующие наиболее распространенные открытые библиотеки:

XGBoost – оптимизированная распределенная библиотека, является высоко эффективной, гибкой и портативной [8]. Обеспечивает параллельный бустинг деревьев (также известный как GBDT, GBM), способный разрешать множество научных задач быстро и точно [9]. Написанный код работает на большинстве существующих платформ, например, Hadoop, SGE, MPI, и способен решать миллиарды примеров. Работает со следующими языками программирования: Python, R, Julia,Scala.

LightGBM – платформа компании Microsoft [10], сконструированная таким образом, чтобы быть распределенной и эффективной со следующими преимуществами по сравнению с конкурентными библиотеками:

- более быстрая скорость тренировки и более высокая эффективность;
- более низкое использование памяти;
- большая точность;
- поддержка параллельного и GPU обучения;
- способна обрабатывать крупные массивы данных.

Работает со следующими языками программирования: Python, R.

CatBoost – отечественная библиотека, представленная в 2017 г. Особенностью алгоритма является построение симметричных деревьев, возможность работы с категориальными признаками, Кроме того, он позволяет обучаться на относительно небольшом количестве неоднородных данных. Работает со следующими языками программирования: Python, R.

3. Подбор параметров обучения и опытное тестирование

Эксперимент проведен на тех же выборках данных, что и в работе [5]. Обучающая выборка представлена 443 исполняемыми файлами операционной системы (ОС) Linux различных версий и разрядностей (32x и 64x), относящихся к 63 различным программам. В тестовую выборку вошло 123 файла, относящихся к тем же 63 программам, все они отличались от файлов, используемых в обучающей выборке, и имели разрядность 32x и 64x.

Эталонные сигнатуры программ, используемые в обучающей выборке, и сигнатуры идентифицируемых программ тестовой выборки имеют одинаковую структуру и представляют собой частотное распределение признака (одной из 10 ассемблерных команд: add, and, call, cmp, je, jmp, lea, mov, pop, push) в каждом из 30 получаемых интервалах разбиения ассемблерного кода программы.

В качестве параметров обучения для решения задачи мультиклассификации [11] при помощи выбранных библиотек градиентного бустинга были выбраны:

для XGBoost :

- booster – тип бустинга;
- eta – размер шага, используемый для предотвращения переобучения;
- max_depth – максимальная глубина дерева;
- num_round – число итераций бустинга;
- objective – метрика оценки, используемая в обучении модели;

для LightGBM :

- boosting – тип бустинга;
- learning_rate – размер шага, используемый для предотвращения переобучения;
- max_depth – максимальная глубина дерева;
- num_iterations – число итераций бустинга;
- objective – метрика оценки, используемая в обучении модели;

для CatBoost :

- boosting_type – схема бустинга;
- learning_rate – скорость обучения, используемая для уменьшения шага градиентного спуска;
- l2_leaf_reg – коэффициент регуляризации L2, используемый для расчета значения листов;

– depth – глубина дерева;
– iterations – максимальное число деревьев, которое будет построено при решении задачи машинного обучения;
– loss_function – метрика (функция потерь), используемая в обучении.

В табл. 1 представлены подобранные эмпирическим путем значения выбранных параметров для решения задачи идентификации.

Таблица 1. Параметры обучения моделей

Параметры обучения	**XGBoost**	**LightGBM**	**CatBoost**
booster/ boosting/ boosting_type	gbtree	gbdt	plain
eta/ learning_rate	0,3	0,3	0,7
-/-/ l2_leaf_reg	-	-	1
max_depth/ depth	2	7	2
num_round/ num_iterations/ iterations	1000	1000	1000
objective/ loss_function	Multi:SoftMax	MultiClass	MultiClass

Остальные значения параметров моделей были установлены по умолчанию.

4. Результаты

Результаты идентификации тестовой выборки по 10 ассемблерным командам приведены на рис. 4. Очевидно, что для почти всех ассемблерных команд число верно идентифицированных программ выше при использовании библиотеки XGBoost и достигает максимального значения в 95,93 % для команды jmp (118 правильно идентифицированных исполняемых файла из 123).

При этом из рисунка 5 видно значительное преимущество LightGBM во времени, затрачиваемого на обучение модели и идентификацию 123 исполняемых файлов тестовой выборки.

Анализ различных подходов сравнения исполняемых файлов был произведен при помощи бикубической меры качества кластеризации [12], выбор которой был обусловлен формой представления результатов проведенных экспериментов в работе [13] и представлен в табл. 2.

Так по результатам табл. 2 видно, что наиболее эффективным, с точки зрения расчета бикубической меры, является машинное обучение на основе метода градиентного бустинга решающих деревьев. В свою очередь из рассмотренных библиотек, XGBoost показывает наилучший результат. Однако стоит отметить, что при проведении эксперимента авторы обнаружили, что разные библиотеки градиентного бустинга бывают не в состоянии идентифицировать некоторое число программ тестовой выборки вне зависимости от выбранного признака, при этом среди трех рассмотренных алгоритмов имена таких программ были различными для каждого алгоритма. Таким образом, совокупность результатов библиотек LightGBM, CatBoost и XGBoost позволит достичь более эффективной идентификации исполняемых файлов.

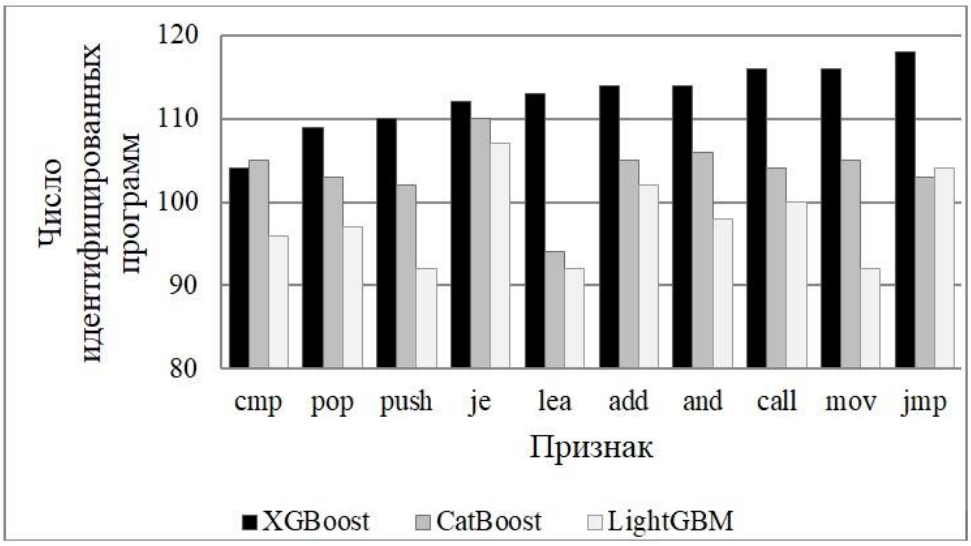

Рис. 4. Число верно идентифицированных исполняемых файлов с применением различных библиотек градиентного бустинга деревьев решений
(Fig. 4. The number of correctly identified executable files with the use of different libraries of gradient boosting decision trees)

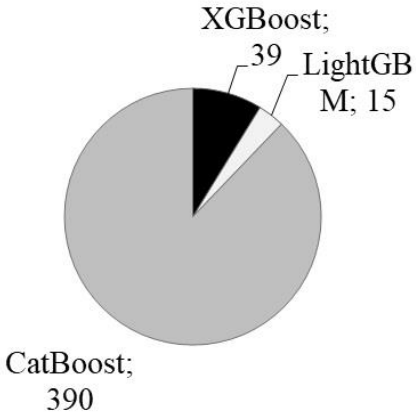

Рис. 5 Время обучения модели классификации и идентификации файлов тестовой выборки
(Fig. 5. Training time of classification model and identification of test sample files)

Таблица 2. Сравнение подходов к идентификации исполняемых файлов

Подход к идентификации файлов	**Максимум бикубической меры**	**Погрешность измерений**
На основе контекстно-зависимого кусочного хеширование [14]	0,29	0,059
На основе евклидова расстояния между векторами для блоков постоянного размера [15]	0,55	0,099
На основе редакционного расстояния между векторами для блоков постоянного размера	0,69	0,123
LightGBM	0,84	0,071
CatBoost	0,88	0,055
XGBoost	0,96	0,027

Заключение

Учитывая приведенные выше результаты, очевидно преимущество разработанного авторами подхода к идентификации программного обеспечения при помощи машинного обучения перед такими подходами сравнения файлов, использующих: контекстно-зависимое кусочное хеширование, евклидово расстояние между векторами для блоков постоянного размера и редакционное расстояние между векторами для блоков переменной длины.

Также стоит отметить, что наилучший результат библиотеки XGBoost перед другими средствами реализации градиентного бустинга деревьев решений применительно к задаче идентификации программного обеспечения.

СПИСОК ЛИТЕРАТУРЫ:

1. Williams S.P., Hardy J.A., Holgate C.A. Information security governance practices in critical infrastructure organizations: a socio-technical and institutional logic perspective // Electronic Markets. 2013. V. 23. N 4. P. 341–351.
2. Salakhutdinova K.I., Krivtsova I.E., Lebedev I.S., Sukhoparov M.E. An Approach to Selecting an Informative Feature in Software Identification // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2018, Vol. 11118. P. 318–327.
3. Krivtsova I.E., Lebedev I.S., Salakhutdinova K.I. Identification of Executable Files on the basis of Statistical Criteria // Proceedings of the 20th Conference of Open Innovations Association FRUCT. 2017. P. 202–208.
4. Салахутдинова К.И., Лебедев И.С., Кривцова И.Е., Сухопаров М.Е. Исследование влияния выбора признака и коэффициента (ratio) при формировании сигнатуры в задаче по идентификации программ // Проблемы информационной безопасности. Компьютерные системы. 2018. № 1. С. 136–141. URL: http://jisp.ru/article/issledovanie-vliyaniya-vybora-priznaka-i-koeffitsienta-ratio-pri-formirovanii-signatury-v-zadache-po-identifikatsii-programm/ (дата обращения: 29.01.2019).
5. Салахутдинова К.И., Лебедев И.С., Кривцова И.Е. Алгоритм градиентного бустинга деревьев решений в задаче идентификации программного обеспечения // Научно-технический вестник информационных технологий, механики и оптики. 2018. Т. 18. № 6(118). С. 1016–1022.doi: 10.17586/2226-1494-2018-18-6-1016-1022.
URL:https://ntv.ifmo.ru/en/article/18236/algoritm_gradientnogo_bustinga_derevev_resheniy__v_zadache_identifikacii_programmnogo_obespecheniya.htm (дата обращения: 29.01.2019).
6. Дружков П.Н., Золотых Н.Ю., Половинкин А.Н. Реализация параллельного алгоритма предсказания в методе градиентного бустинга деревьев решений // Вестник ЮурГУ. 2011. № 37 (254). С. 82–89.
7. CatBoost GitHub [Электронный ресурс]. URL: https://github.com/catboost (дата обращения: 29.01.2019).
8. XGBoost GitHub [Электронный ресурс]. URL: https://github.com/dmlc/xgboost (дата обращения: 09.02.2019).
9. Китов В.В. Исследование точности метода градиентного бустинга со случайными поворотами // Статистика и экономика. 2016. № 4. С. 22–26.
10. LightGBM GitHub [Электронный ресурс]. URL: https://github.com/Microsoft/LightGBM (дата обращения: 02.02.2019).
11. Кафтанников И.Л., Парасич А.В. Особенности применения деревьев решений в задачах классификации // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». 2015. № 3(15). С. 26–32.
12. Bagga A., Baldwin B. Cross-Document EventCoreference: Annotations, Experiments, and Observations // Proc. ACL-99 Workshop on Coreference and Its Applications. 1998. С. 1–8.
13. Антонов А. Е., Федулов А. С. Идентификация типа файла на основе структурного анализа//Прикладная информатика. 2013. № 2(44). С. 68–77.
14. Kornblum J. D. Identifying almost identical files using context triggered piecewise hashing // Digital Investigation. 2006. Vol. 3. P. 91–97.
15. Ebringer T., Sun L., Boztas S. A Fast Randomness Test that Preserves Local Detail // Proceedings of the 18th Virus Bulletin International Conference — United Kingdom: Virus Bulletin Ltd. 2008. P. 34–42.

REFERENCES:

[1] Williams S.P., Hardy J.A., Holgate C.A. Information security governance practices in critical infrastructure organizations: a socio-technical and institutional logic perspective. Electronic Markets. 2013. V. 23. N 4. P. 341–351.

[2] Salakhutdinova K.I., Krivtsova I.E., Lebedev I.S., Sukhoparov M.E. An Approach to Selecting an Informative Feature in Software Identification. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2018, Vol. 11118. P. 318-327.
[3] Krivtsova I.E., Lebedev I.S., Salakhutdinova K.I. Identification of Executable Files on the basis of Statistical Criteria. Proceedings of the 20th Conference of Open Innovations Association FRUCT. 2017. P. 202–208.
[4] Salakhutdinova K.I., Lebedev I.S., Krivtsova I.E., Sukhoparov M.E. Study of the effect of selection feature and coefficient (ratio) in the signature formation in the task of program identification. Information Security Problems. Computer Systems. 2018. № 1. P. 136–141. URL: http://jisp.ru/en/article/issledovanie-vliyaniya-vybora-priznaka-i-koeffitsienta-ratio-pri-formirovanii-signatury-v-zadache-po-identifikatsii-programm/ (accessed:29.01.2019).
[5] Salakhutdinova K.I., Lebedev I.S., Krivtsova I.E. Gradient boosting trees method in the task of software identification. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 2018, vol. 18, № 6. P. 1016–1022. doi: 10.17586/2226-1494-2018-18-6-1016-1022. URL:https://ntv.ifmo.ru/en/article/18236/algoritm_gradientnogo_bustinga_derevev_resheniy__v_zadache_identifikacii_programmnogo_obespecheniya.htm (accessed:29.01.2019).
[6] Druzhkov P.N., Zolotykh N.Yu., Polovinkin A.N. Parallel implementation of prediction algorithm in gradient boosting trees method. Bulletin SUSU, 2011, №. 37. P. 82–89.
[7] CatBoost GitHub [Elektronnyy resurs]. URL: https://github.com/catboost (accessed:29.01.2019).
[8] XGBoost GitHub [Elektronnyy resurs]. URL: https://github.com/dmlc/xgboost (accessed: 09.02.2019).
[9] Kitov V. V. Investigation of the accuracy of the gradient boosting method with random turns. Statistics and Economics. 2016. №. 4. P. 22–26.
[10] LightGBM GitHub [Elektronnyy resurs]. URL: https://github.com/Microsoft/LightGBM (accessed: 02.02.2019).
[11] Kaftannikov I.L., Parasich A.V. Decision tree's features of application in classification problem. Bulletin SUSU, Computer Technologies, Automatic Control & Radioelectronics, 2015, №. 3. P. 26–32.
[12] Bagga A., Baldwin B. Cross-Document Event Coreference: Annotations, Experiments, and Observations. Proc. ACL-99 Workshop on Coreference and Its Applications. 1998. P. 1–8.
[13] Antonov A.E., Fedulov A.S. File type identification based on structural analysis. Journal of Applied Informatics, 2013, №. 2. P. 68–77.
[14] Kornblum J. D. Identifying almost identical files using context triggered piecewise hashing. Digital Investigation — 2006. Vol. 3. P. 91–97.
[15] Ebringer T., Sun L., Boztas S. A Fast Randomness Test that Preserves Local Detail. Proceedings of the 18th Virus Bulletin International Conference — United Kingdom: Virus Bulletin Ltd. 2008. P. 34–42.

Поступила в редакцию – 5 марта 2019 г. Окончательный вариант – 31 мая 2019 г.
Received – March 5, 2019. The final version – March 31, 2019.