УДК 004.056.5
DOI: 10.26102/2310-6018/2026.59.8.011

Использование системного анализа при проектировании интеллектуальных систем защиты от вредоносных программ

А.А. Абедалхуссайн[1]✉, Е.В. Ляпунцова[1,2]

[1]Национальный исследовательский технологический университет «МИСИС», Москва, Российская Федерация
[2]Московский государственный технический университет им. Н.Э. Баумана, Москва, Российская Федерация

Резюме. В работе рассматривается задача проектирования интеллектуальной системы защиты от вредоносных программ с учетом изменчивости входного потока, ограничений по вычислительным ресурсам и необходимости устойчивой работы на новых данных. Цель исследования состояла в том, чтобы показать, как методы системного анализа позволяют перейти от обычного сравнения моделей к обоснованному выбору архитектуры, схемы проверки и сценария обновления системы. Экспериментальная база включала 134435 объектов, из которых 57293 относились к вредоносным, а после исключения записей без распознанной даты и раннего архивного фрагмента было сформировано экспериментальное подмножество из 123704 объектов за период с августа 2019 года по сентябрь 2020 года. В исследовании сопоставлены перемешанная и хронологическая схемы проверки, базовые модели классификации, варианты сокращения признакового пространства, а также одноконтурная и двухконтурная архитектуры защиты. Показано, что использование 256 признаков в первом контуре обеспечивает почти то же качество, что и полное описание из 2381 признака, при заметно меньших вычислительных затратах. Установлено, что двухконтурная схема при передаче на второй контур 0,25 процента объектов сохраняет качество полной модели и почти не увеличивает среднее время обработки одного объекта. Дополнительно показано, что обновление первого контура повышает устойчивость системы во времени и снижает риск деградации качества на новых месяцах наблюдения.

Ключевые слова: системный анализ, вредоносные программы, интеллектуальная система защиты, хронологическая проверка, двухконтурная архитектура, сокращение признаков, адаптивное обновление, машинное обучение.

Для цитирования: Абедалхуссайн А.А., Ляпунцова Е.В. Использование системного анализа при проектировании интеллектуальных систем защиты от вредоносных программ. *Моделирование, оптимизация и информационные технологии.* 2026;14(8). URL: https://moitvivt.ru/ru/journal/article?id=2389 DOI: 10.26102/2310-6018/2026.59.8.011

Using systems analysis in designing intelligent protection systems against malicious software

A.A. Abedalhussain[1]✉, E.V. Lyapuntsova[1,2]

[1]National University of Science and Technology "MISIS", Moscow, the Russian Federation
[2]Bauman Moscow State Technical University, Moscow, the Russian Federation

Abstract. The paper addresses the problem of designing an intelligent protection system against malicious software under changing input flows, limited computational resources, and the need for stable operation on new data. The aim of the study was to show how systems analysis methods make it possible to move from a simple comparison of classification models to a justified choice of architecture, evaluation strategy, and update scenario. The experimental basis included 134435 objects, of which 57293 were malicious. After removing records without a recognized date and excluding the early

archival fragment, an experimental subset of 123704 objects covering the period from August 2019 to September 2020 was formed. The study compares randomized and chronological evaluation schemes, baseline classification models, several feature reduction variants, and both single-stage and two-stage protection architectures. The results show that using 256 features in the first stage provides almost the same recognition quality as the full 2381-feature representation while requiring substantially lower computational costs. It is also shown that the two-stage architecture preserves the quality of the full model while sending only 0.25 percent of objects to the second stage and almost not increasing the average processing time per object. In addition, first-stage updating improves temporal robustness and reduces the risk of quality degradation on later monthly data segments.

Keywords: systems analysis, malicious software, intelligent protection system, chronological evaluation, two-stage architecture, feature reduction, adaptive updating, machine learning.

For citation: Abedalhussain A.A., Lyapuntsova E.V. Using systems analysis in designing intelligent protection systems against malicious software. *Modeling, Optimization and Information Technology.* 2026;14(8). (In Russ.). URL: https://moitvivt.ru/ru/journal/article?id=2389 DOI: 10.26102/2310-6018/2026.59.8.011

Введение

Поток вредоносных объектов в рабочей среде меняется быстрее, чем успевает обновляться типовая модель детектирования. Речь идет не только о появлении новых образцов, но и о сдвиге в составе семейств, частоте ложных тревог, времени реакции аналитика и цене каждого дополнительного шага обработки. В рекомендациях NIST по реагированию на инциденты обнаружение уже встроено в общую логику управления рисками и непрерывного улучшения процедур[1], а в обзоре ENISA за 2024 год среди ведущих угроз отдельно выделены атаки на доступность, программы-вымогатели и угрозы для данных[2].

Под системным анализом в данной работе понимается разбор интеллектуальной защиты как связанной конструкции из входного потока, признакового описания, правил передачи объекта на углубленный анализ, критериев качества и сценариев обновления. В центре внимания оказывается вся цепочка принятия решения. К ней относится то, сколько признаков использует первый контур, какую долю объектов приходится эскалировать на второй, как быстро растет среднее время обработки и в какой точке прибавка качества перестает оправдывать усложнение архитектуры. Схема такого разбора особенно уместна для задач защиты от вредоносных программ, где лишний процент ложных срабатываний легко превращается в потерянное время аналитика, а тяжеловесная модель плохо переносится в потоковый режим.

Литература по машинному обучению для обнаружения вредоносных программ накопила множество работ с высокими метриками, однако уверенность в якобы решенной задаче давно выглядит завышенной. В обзоре [1] методические изъяны, ограничения наборов данных и слабая проверка на новых выборках названы среди причин, по которым хорошие результаты в эксперименте не гарантируют надежную эксплуатацию. Для анализа исполняемых файлов формата Portable Executable, далее PE, временной аспект вообще нельзя отодвигать на второй план: авторы BODMAS прямо связывали ценность набора с наличием временных меток и аккуратно выверенной семейственной разметки, без которых трудно изучать дрейф понятий и эволюцию

[1] Nelson A., Rekhi S., Souppaya M., et al. *Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile. NIST SP 800-61r3.* Gaithersburg: NIST; 2025. 40 p. https://doi.org/10.6028/NIST.SP.800-61r3

[2] European Union Agency for Cybersecurity. *ENISA Threat Landscape 2024.* Heraklion: ENISA; 2024. 129 p. https://doi.org/10.2824/0710888

семейств [2], а в наборе BenchMFC временная изменчивость уже заложена в саму конструкцию бенчмарка [3].

Материал настоящего исследования выбран по той же причине. BODMAS изначально разрабатывался как открытый набор для временного анализа PE-вредоносных программ [2], а в нашем эксперименте полный массив включает 134435 объектов и 2381 признак, тогда как после исключения записей без распознанной даты и раннего архивного фрагмента рабочее подмножество содержит 123704 объекта за период с августа 2019 года по сентябрь 2020 года, что отражено в Таблице 1.

Работы, посвященные дрейфу вредоносного поведения, подводят к той же инженерной проблеме с другой стороны. В статье [4] изменения в поведении программ-вымогателей описаны как фактор, сокращающий срок полезной работы классификатора между переобучениями; в недавней работе по адаптации к реально дрейфующим PE-данным временно согласованное разбиение уже используется как естественная схема проверки обновляемого детектора [5]. Практический смысл здесь заключается в том, что архитектура защиты должна допускать обновление без резкого роста стоимости обработки и без провала на новых месяцах наблюдения. Датированные записи полного массива охватывают 14 календарных месяцев: с августа 2019 года по сентябрь 2020 года. Экспериментальное подмножество сформировано в пределах того же временного интервала после исключения записей без распознанной даты и раннего архивного фрагмента, не включенного в основной эксперимент

Таблица 1 – Характеристика набора данных и экспериментального подмножества
Table 1 – Dataset characteristics and experimental subset

Показатель	**Полный набор данных**	**Экспериментальное подмножество**
Число объектов	134435	123704
Число признаков	2381	2381
Число безопасных объектов	77142	66411
Число вредоносных объектов	57293	57293
Число уникальных вредоносных семейств	582	582
Число месяцев наблюдения	14	14
Число записей без распознанной даты	3712	0
Границы основного временного интервала	2019-08–2020-09	2019-08–2020-09

Цель работы состоит в том, чтобы показать, как системный анализ меняет логику проектирования интеллектуальной системы защиты от вредоносных программ на материале BODMAS. Проверка сосредоточена на четырех связанных вопросах:

1) насколько искажается оценка качества при переходе от перемешанной схемы к хронологической;

2) где проходит разумная граница сокращения признакового пространства;

3) удается ли двухконтурной архитектуре удержать качество полной модели при компактном первом контуре;

4) как меняются качество и доля ложных срабатываний при разных сценариях обновления.

Материалы и методы

В качестве экспериментального материала использовался набор BODMAS, содержащий статическое признаковое описание исполняемых файлов формата Portable Executable и временные метки появления объектов. Исходный массив включал 134435 объектов и 2381 признак на объект; после исключения 3712 записей без распознанной даты и отсечения раннего архивного фрагмента, не содержащего рабочего потока вредоносных семейств, в основной эксперимент вошли 123704 объекта за период с августа 2019 года по сентябрь 2020 года. В исследование попадал тот интервал, где уже присутствует совместная динамика безопасных и вредоносных файлов и где можно проверять устойчивость системы на временном сдвиге потока.

Хронологическая организация выборки была принята как обязательная часть эксперимента, поскольку в задачах обнаружения вредоносных программ временной сдвиг меняет статистическую структуру данных и постепенно «старит» модель. Работы, ориентированные на реальную эксплуатацию, все чаще используют temporal split, то есть разбиение по времени, а не по случайно перемешанным строкам [6, 7]. Аналогичная логика проверки применяется и в исследованиях, где специально анализируется старение модели при изменении входного потока [8].

Состав обеих схем проверки приведен в Таблице 2. Перемешанная схема сохраняла стратификацию по классам и использовалась как формальный ориентир, хронологическая разбивала поток по месяцам и моделировала эксплуатационный сценарий, в котором обучение выполняется на более ранних данных, а итоговая оценка проводится на более поздних объектах.

Таблица 2 – Схемы разбиения данных для экспериментальной оценки
Table 2 – Data splitting schemes for experimental evaluation

Схема проверки	Часть	Число объектов	Доля вредоносных объектов	Временной интервал
Перемешанная	Обучающая	86592	0,4631	2019-08–2020-09
Перемешанная	Проверочная	18556	0,4631	2019-08–2020-09
Перемешанная	Итоговая	18556	0,4631	2019-08–2020-09
Хронологическая	Обучающая	63395	0,4668	2019-08–2020-03
Хронологическая	Проверочная	33810	0,4231	2020-04–2020-06
Хронологическая	Итоговая	26499	0,5055	2020-07–2020-09

Волна первых появлений новых вредоносных семейств, представленная на Рисунке 1, наглядно иллюстрирует поставленную проблему. Основной временной интервал начинается резким притоком новых семейств, затем интенсивность снижается, но полностью не исчезает.

Перемешанная схема строилась в пропорции 70/15/15 с сохранением доли вредоносных объектов в каждой части. В результате обучающая подвыборка содержала 86592 объекта, проверочная 18556 и итоговая 18556. Хронологическая схема формировалась иначе: в обучающую часть включались месяцы с 2019-08 по 2020-03, в проверочную с 2020-04 по 2020-06, в итоговую с 2020-07 по 2020-09. Число объектов составило соответственно 63395, 33810 и 26499, а доля вредоносных объектов заметно менялась от части к части, что и делало такую проверку более жесткой.

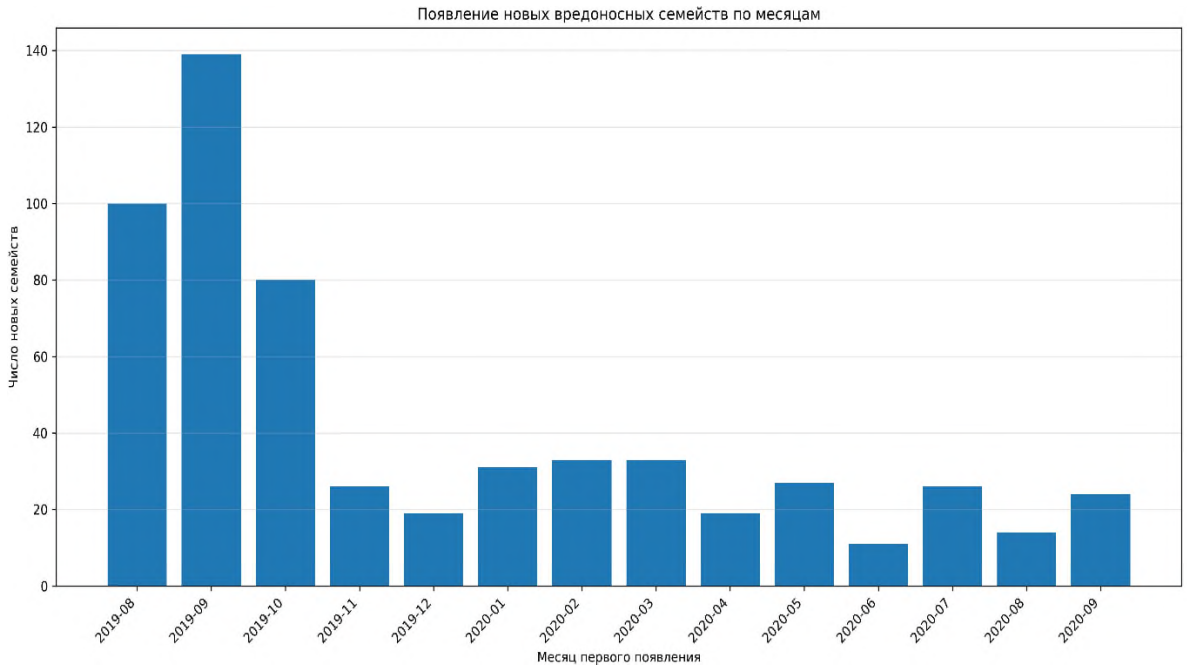

Рисунок 1 – Появление новых вредоносных семейств по месяцам
Figure 1 – Emergence of new malware families by month

Хронологическое разбиение разводит данные по месяцам и тем самым фиксирует сдвиг входного потока. Так, доля вредоносных объектов меняется с 0,4668 в обучающей части до 0,4231 в проверочной и 0,5055 в итоговой, а сами границы интервалов вынесены на Рисунок 2.

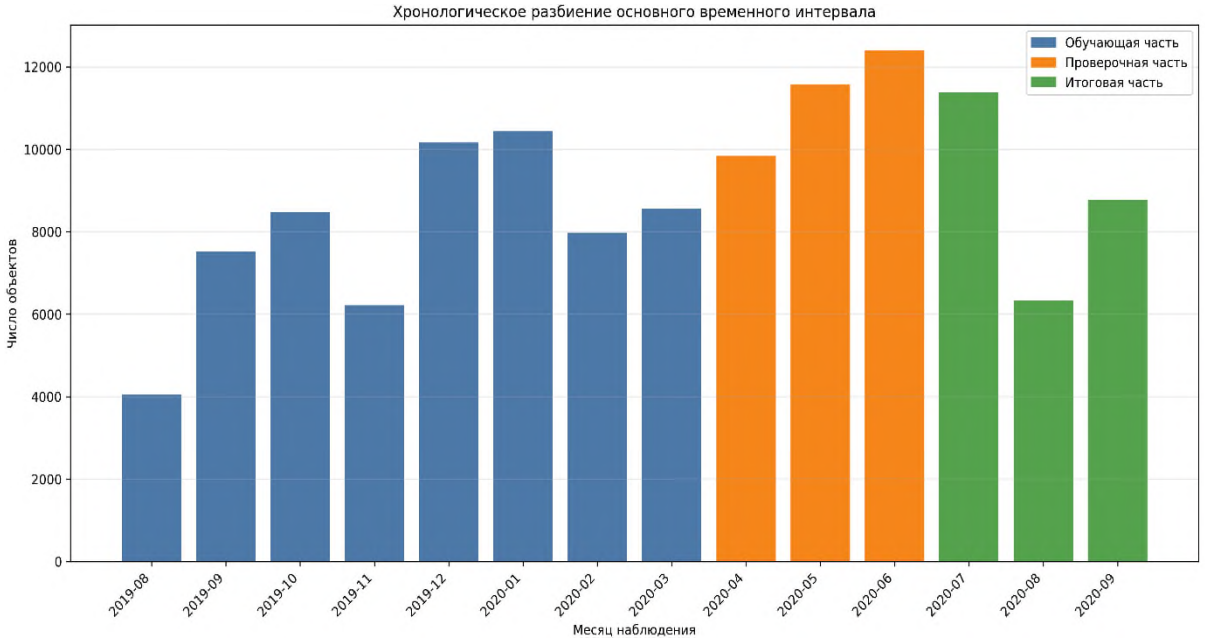

Рисунок 2 – Хронологическое разбиение основного временного интервала
Figure 2 – Chronological split of the main time interval

Базовая линейка моделей подбиралась так, чтобы сравнение не замыкалось внутри одного семейства алгоритмов. В первый набор вошла линейная логистическая модель с поэтапным обновлением весов, удобная как быстрый и прозрачный ориентир.

Второй вариант представлял ансамбль экстремально рандомизированных деревьев, где случайный выбор признаков и точек разбиения снижает коррелированность деревьев и хорошо работает на неоднородных признаковых пространствах [9]. Третьим опорным вариантом стал градиентный бустинг LightGBM, сохраняющий высокую точность при работе с большими табличными массивами за счет гистограммного построения разбиений и экономного обращения с признаками [10].

Для ранжирования использовались средние абсолютные вклады признаков, рассчитанные методом SHAP, где вклад каждой переменной интерпретируется через локальное изменение прогноза модели [11]. После построения упорядоченного списка признаков были сформированы сокращенные наборы размера 64, 128, 256 и 512, а затем проведено повторное обучение на каждом из них. Рациональным вариантом для первого контура был заранее принят набор из 256 признаков: при нем гармоническое среднее точности и полноты почти совпадало с полной моделью, а выигрыш по времени обучения и предсказания уже был ощутим.

Двухконтурная схема строилась вокруг управляемой передачи сомнительных объектов на углубленный анализ. Первый контур выдавал для *i*-го объекта вероятность принадлежности к вредоносному классу p_i, после чего вычислялась мера сомнительности решения:

$$u_i = 0{,}5 - |p_i - 0{,}5|. \quad (1)$$

При таком определении мера сомнительности принимает максимальное значение u_i = 0,5 при p_i = 0,5 и уменьшается по мере приближения вероятности к 0 или 1. Дополнительно уточнено, что на второй контур передаются объекты с наибольшими значениями u_i. На второй контур направлялась заранее заданная доля наиболее сомнительных объектов:

$$r = \frac{N_2}{N}, \quad (2)$$

где r – доля объектов, направляемых на углубленный анализ; N_2 – число объектов, переданных на второй контур; N – общее число объектов во входном потоке.

Похожая логика селективного углубления анализа встречается и в многоуровневых схемах детектирования [7, 8], где легкий первый уровень обслуживает основной поток, а более дорогой уровень активируется только для части экземпляров с повышенным риском ошибки.

Второй контур использовал полное описание объекта из 2381 признака и ту же модельную основу, что и лучший одноконтурный вариант. Подбор доли r проводился на проверочной части хронологической схемы. Среди допустимых вариантов выбирался тот, который удерживал гармоническое среднее точности и полноты на уровне, близком к полной одноконтурной модели, но не терял преимущества по времени обработки. По результатам такого отбора была зафиксирована доля r = 0,0025, то есть на второй контур передавалось около 0,25 % входного потока. Для итоговой части это соответствовало 67 объектам из 26499.

Проверка устойчивости первого контура во времени проводилась в трех режимах: без обновления, с периодическим обновлением и с адаптивным обновлением. Режим без обновления сохранял исходную модель на всем интервале итоговой оценки. Периодический сценарий предполагал переобучение через равные временные шаги. Адаптивный сценарий запускал обновление при накоплении признаков деградации на очередном фрагменте потока.

Оценка качества опиралась на набор метрик, разделяющих разные типы ошибок. Точность положительных срабатываний вычислялась по формуле:

$$P = \frac{TP}{TP+FP}, \quad (3)$$

где P – точность положительных срабатываний; TP – число истинно положительных решений, то есть правильно распознанных вредоносных объектов; FP – число ложно положительных решений, то есть безопасных объектов, ошибочно отнесенных к вредоносным.

Полнота обнаружения вредоносных объектов задавалась выражением:

$$R = \frac{TP}{TP+FN}, \quad (4)$$

где R – полнота обнаружения вредоносных объектов; TP – число истинно положительных решений; FN – число ложно отрицательных решений, то есть вредоносных объектов, ошибочно отнесенных к безопасным.

В качестве основной интегральной меры использовалось гармоническое среднее точности и полноты:

$$F_1 = \frac{2PR}{P+R}, \quad (5)$$

где F_1 – гармоническое среднее точности положительных срабатываний и полноты обнаружения; P – точность положительных срабатываний; R – полнота обнаружения вредоносных объектов.

Дополнительно для каждой модели рассчитывались общая точность классификации, доля ложных срабатываний и среднее время обработки одного объекта. Порог бинарного решения принимался равным 0,5, что позволяло напрямую сопоставлять одноконтурные и двухконтурные варианты в единой шкале. Все вычислительные эксперименты выполнялись на рабочей станции с процессором Intel Core i7-12700H, 14 ядрами, 16 ГБ оперативной памяти, без использования графического процессора. Использовалась операционная система Windows 11 Pro. Программная реализация выполнена на языке Python версии 3.10 с использованием библиотек scikit-learn версии 1.3.2, LightGBM версии 4.1.0 и SHAP версии 0.44.0. Среднее время обработки одного объекта рассчитывалось как отношение общего времени прогнозирования на соответствующей итоговой выборке к числу обработанных объектов. Для каждого варианта выполнялось 10 повторных измерений после 3 прогревочных запусков; в таблицах приведено среднее арифметическое значение. Вычисления выполнялись с использованием одного вычислительного потока.

Результаты

Разрыв между случайной и временной проверкой, который уже обсуждался в работах о моделировании детектирования как эволюционирующего потока и о задержках разметки [12, 13], в нашем эксперименте проявился довольно резко. Близкая проблема устойчивости при дрейфе входного потока рассматривается и в федеративных схемах мобильного детектирования [14]. По данным Таблицы 3 и Рисункам 3, 4 лучшим вариантом в обеих схемах остался градиентный бустинг по деревьям решений: при перемешанном разбиении его гармоническое среднее точности и полноты составило 0,9967, при хронологическом 0,9960, а доля ложных срабатываний снизилась с 0,0024 до 0,0009. Линейная логистическая модель потеряла меньше: 0,9890 против 0,9870 по F1. Ансамбль деревьев решений оказался заметно чувствительнее к смене временного интервала: его F1 снизилось с 0,9924 до 0,9654, хотя доля ложных срабатываний при этом уменьшилась до 0,0011. Картина получилась несимметричной, поскольку более строгая проверка ударила прежде всего по полноте обнаружения у промежуточной модели, а не по всем алгоритмам сразу.

Сокращение признакового пространства дало плавную, а не обвальную зависимость качества от размера описания. В Таблице 4 и на Рисунках 5, 6 видно, что переход от 64 признаков к полному набору из 2381 признака увеличил F1 лишь с 0,9955 до 0,9961, тогда как время обучения выросло с 10,5735 до 146,5926 секунды, а среднее время обработки одного объекта – с 0,025974 до 0,045175 мс. Вариант на 256 признаках занял здесь рабочую середину: F1 осталось на уровне 0,9958, доля ложных срабатываний составила 0,0009, время обучения – 36,5652 секунды, среднее время обработки – 0,031849 мс. Прирост качества после этой точки уже почти не читается, а вычислительная цена продолжает расти.

Сравнение архитектур показало, что переход к двум контурам дал выигрыш не за счет формального роста метрик, а за счет перераспределения вычислительной нагрузки. Согласно Таблице 5 и Рисункам 7, 8 компактная одноконтурная схема на 256 признаках показала F1 = 0,9958 при среднем времени обработки 0,031694 мс на объект, полная одноконтурная схема на 2381 признаке – F1 = 0,9960 при 0,065328 мс, а двухконтурная схема с тем же первым контуром и полным вторым достигла F1 = 0,9960 при 0,032210 мс. На второй контур передавалось 0,0025 входного потока; на итоговой части это соответствовало 67 объектам из 26499. По времени двухконтурная архитектура почти совпала с компактной, по качеству – с полной.

Таблица 3 – Сопоставление базовых моделей при двух схемах проверки
Table 3 – Comparison of baseline models under two evaluation schemes

Схема проверки	**Модель**	**Точность классификации**	**Полнота обнаружения**	**Гармоническое среднее точности и полноты**	**Доля ложных срабатываний**	**Среднее время на один объект, мс**
Перемешанная	Линейная логистическая модель	0,9898	0,9917	0,9890	0,0119	0,015312
Перемешанная	Ансамбль деревьев решений	0,9930	0,9923	0,9924	0,0064	0,026061
Перемешанная	Градиентный бустинг по деревьям решений	0,9970	0,9963	0,9967	0,0024	0,025359
Хронологическая	Линейная логистическая модель	0,9869	0,9847	0,9870	0,0109	0,007889
Хронологическая	Ансамбль деревьев решений	0,9662	0,9342	0,9654	0,0011	0,038578
Хронологическая	Градиентный бустинг по деревьям решений	0,9960	0,9929	0,9960	0,0009	0,028289

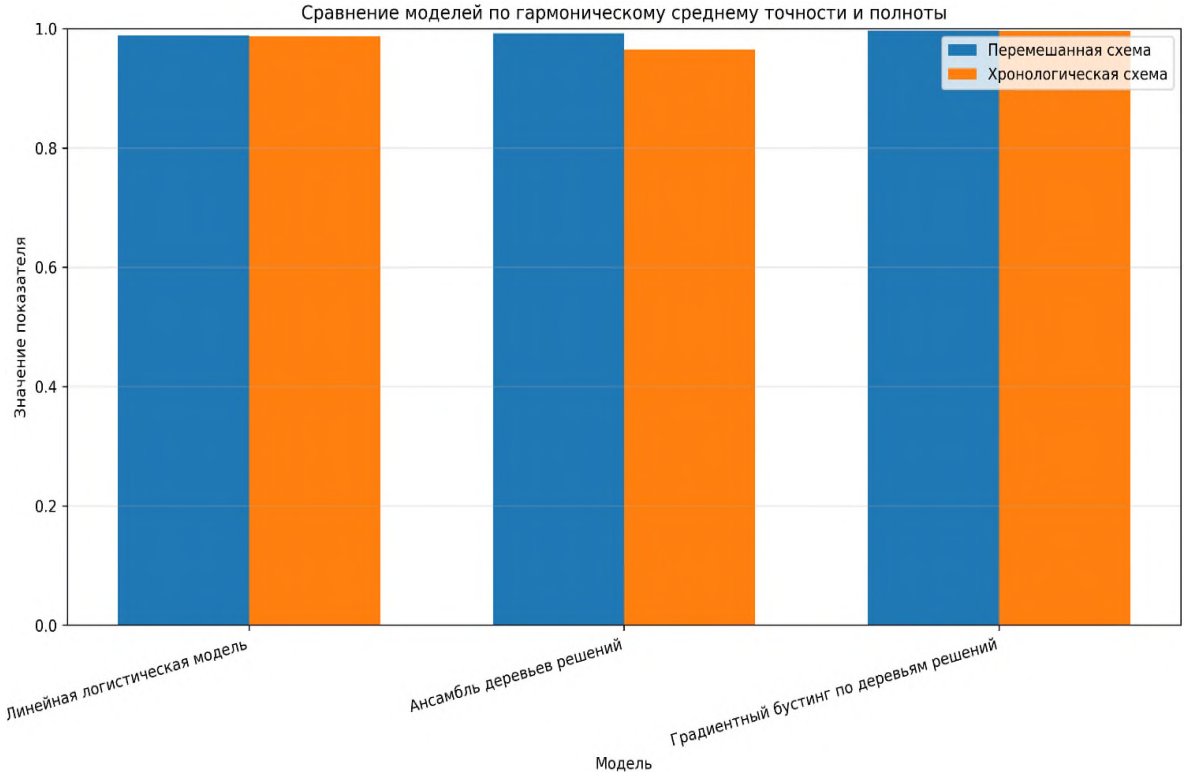

Рисунок 3 – Сравнение базовых моделей по гармоническому среднему точности и полноты
Figure 3 – Comparison of baseline models by f1 score

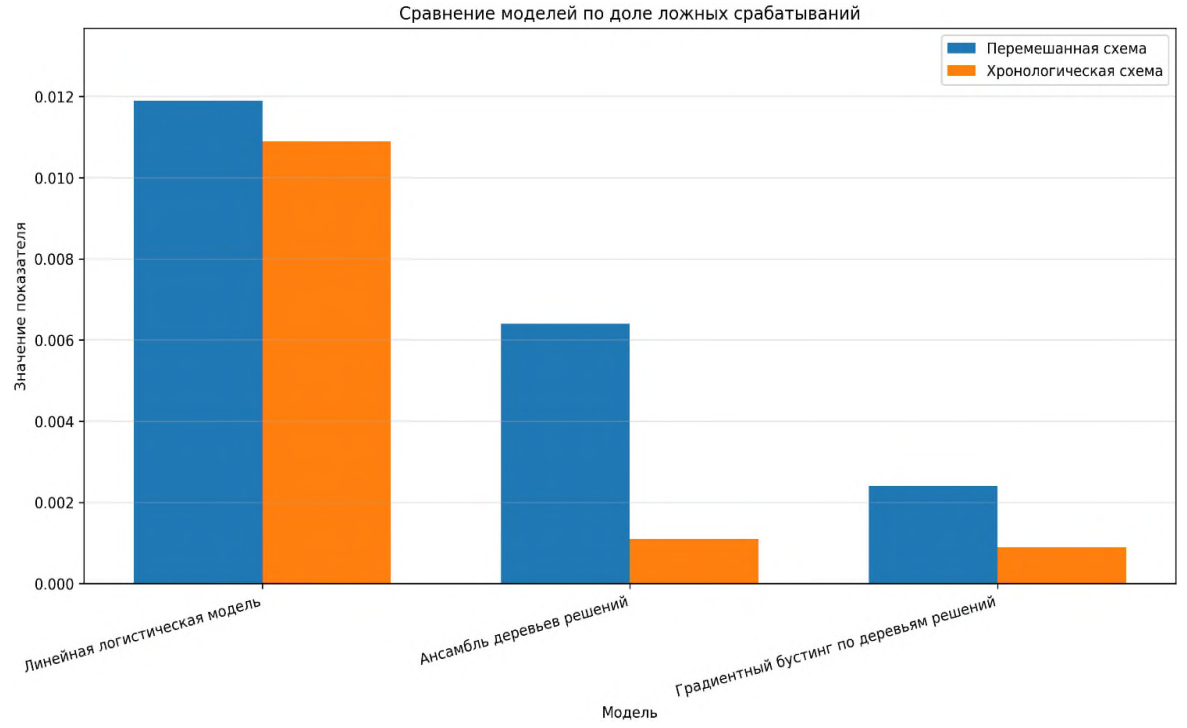

Рисунок 4 – Сравнение базовых моделей по доле ложных срабатываний
Figure 4 – Comparison of baseline models by false positive rate

Таблица 4 – Влияние числа признаков на качество и вычислительные затраты
Table 4 – Influence of feature count on quality and computational cost

Число признаков	Точность классификации	Полнота обнаружения	Гармоническое среднее точности и полноты	Доля ложных срабатываний	Время обучения, сек	Среднее время на один объект, мс
64	0,9954	0,9921	0,9955	0,0011	10,5735	0,025974
128	0,9956	0,9924	0,9956	0,0011	18,6544	0,029035
256	0,9958	0,9925	0,9958	0,0009	36,5652	0,031849
512	0,9959	0,9928	0,9960	0,0009	73,3312	0,039904
2381	0,9960	0,9931	0,9961	0,0010	146,5926	0,045175

Рисунок 5 – Зависимость качества распознавания от числа признаков
Figure 5 – Dependence of recognition quality on the number of features

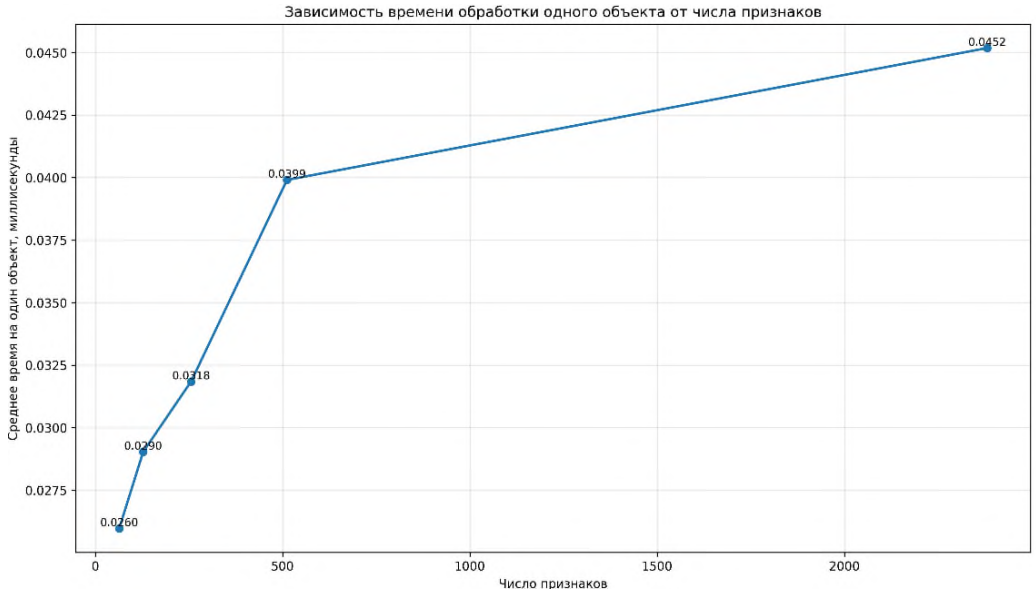

Рисунок 6 – Зависимость времени обработки одного объекта от числа признаков
Figure 6 – Dependence of object processing time on the number of features

Таблица 5 – Сравнение архитектур интеллектуальной системы защиты
Table 5 – Comparison of intelligent protection system architectures

Архитектура	Число признаков первого контура	Число признаков второго контура	Доля объектов, переданных на второй контур	Гармоническое среднее точности и полноты	Доля ложных срабатываний	Среднее время на один объект, мс
Одноконтурная компактная схема	256	0	0,0000	0,9958	0,0009	0,031694
Одноконтурная полная схема	2381	0	0,0000	0,9960	0,0009	0,065328
Двухконтурная схема	256	2381	0,0025	0,9960	0,0010	0,032210

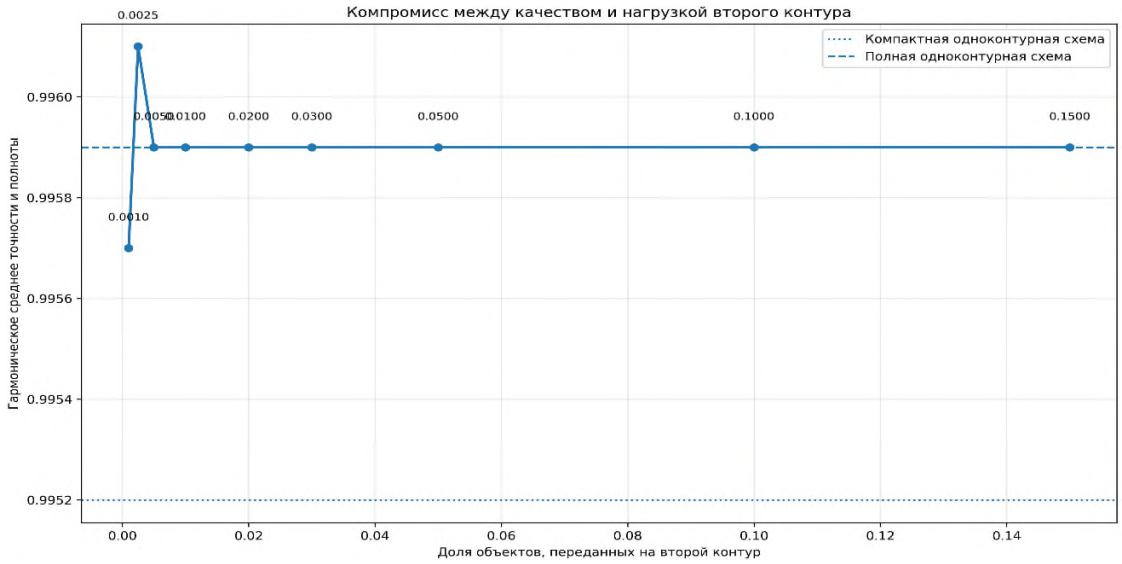

Рисунок 7 – Компромисс между качеством распознавания и нагрузкой второго контура
Figure 7 – Trade-off between recognition quality and second-stage load

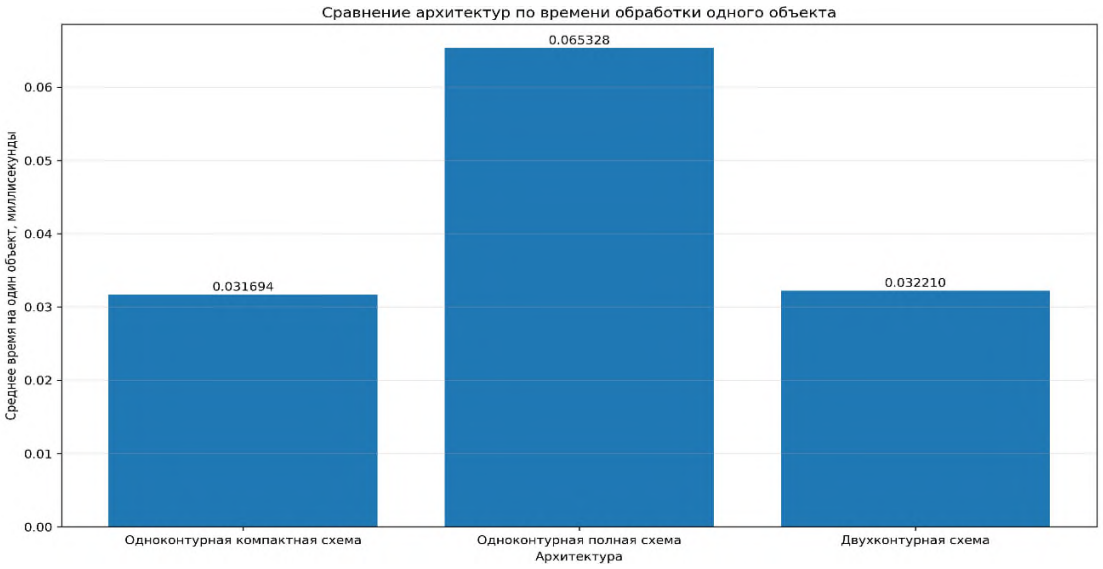

Рисунок 8 – Сравнение архитектур по времени обработки одного объекта
Figure 8 – Comparison of architectures by object processing time

Изменение режима обновления первого контура сработало, прежде всего, на устойчивость по месяцам. Рисунки 9, 10 и Таблица 6 показывают, что сценарий без обновления дал итоговое F1 = 0,9897 при доле ложных срабатываний 0,0119 и среднем времени обработки 0,029836 мс. Периодическое обновление после трех переобучений подняло F1 до 0,9911 и снизило долю ложных срабатываний до 0,0100, сохранив при этом наименьшее среднее время обработки – 0,025268 мс. Адаптивное обновление с пятью переобучениями вышло на то же значение F1 = 0,9911 при доле ложных срабатываний 0,0101 и времени 0,027837 мс. На помесячных кривых особенно хорошо заметен провал февраля и апреля у режима без обновления: после запуска обновлений линия качества выравнивается, а доля ложных срабатываний быстрее возвращается к низким значениям.

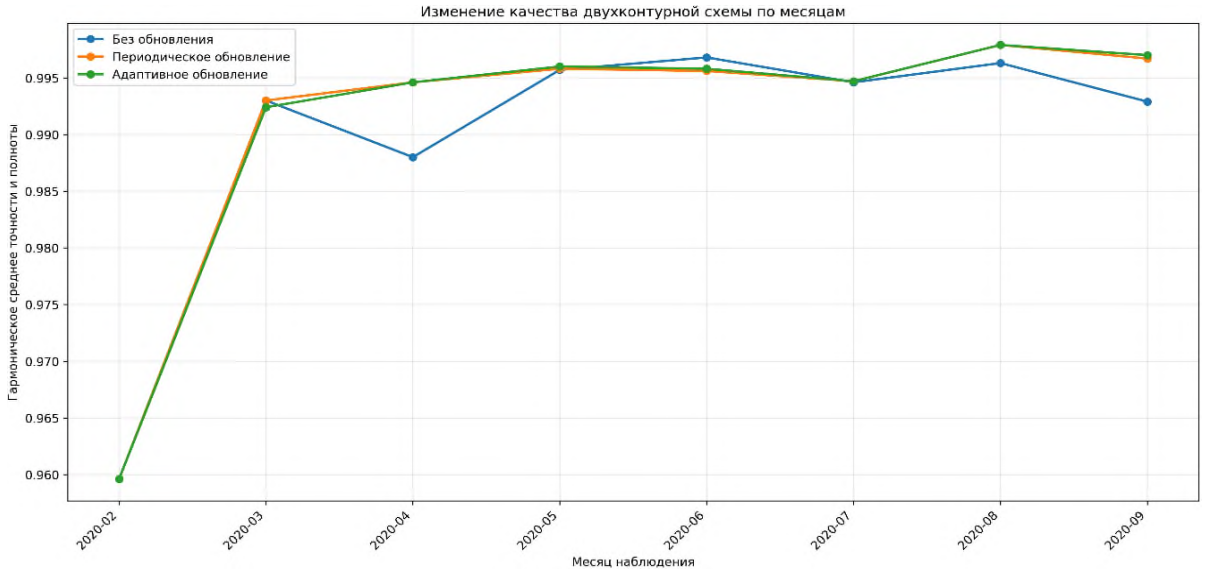

Рисунок 9 – Изменение качества двухконтурной схемы по месяцам при разных сценариях обновления

Figure 9 – Monthly quality change of the two-stage scheme under different update scenarios

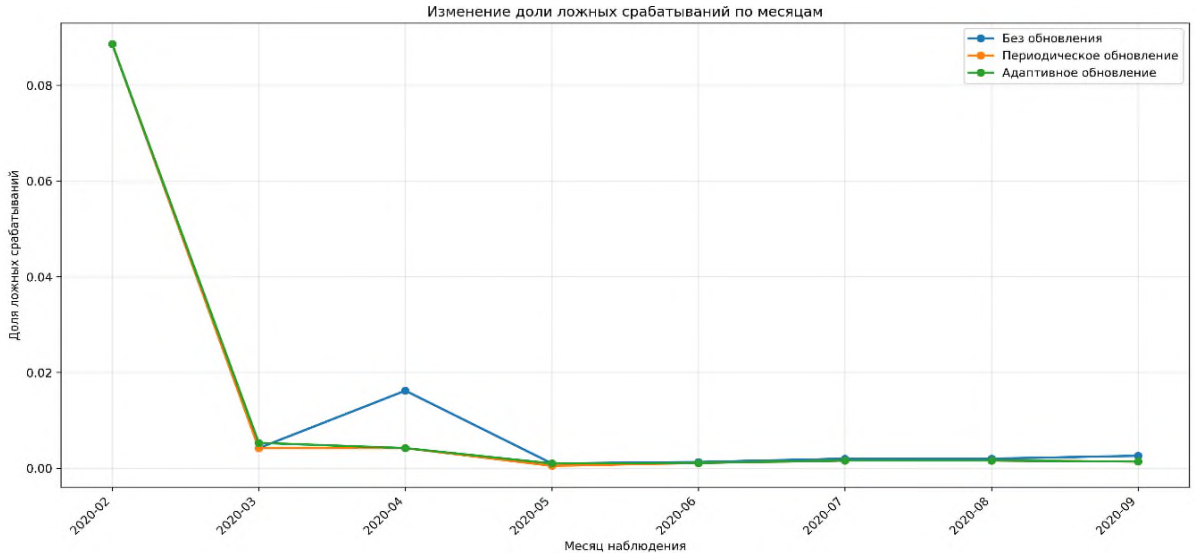

Рисунок 10 – Изменение доли ложных срабатываний по месяцам при разных сценариях обновления

Figure 10 – Monthly change in false positive rate under different update scenarios

Таблица 6 – Сопоставление сценариев обновления первого контура
Table 6 – Comparison of first-stage update scenarios

Сценарий обновления	Число переобучений первого контура	Гармоническое среднее точности и полноты	Доля ложных срабатываний	Среднее время на один объект, мс
Без обновления	0	0,9897	0,0119	0,029836
Периодическое обновление	3	0,9911	0,0100	0,025268
Адаптивное обновление	5	0,9911	0,0101	0,027837

Наибольший средний абсолютный вклад получили признаки с индексами 658, 637, 2359, 2360 и 613; их значения лежат в диапазоне от 1,215892 до 0,440974. Список сам по себе не дает семантической интерпретации, поскольку в текущем наборе индексы не сопровождаются предметными именами признаков, но хорошо показывает, что после сокращения пространства решение первого контура по-прежнему опирается на сравнительно узкое ядро наиболее информативных признаков.

Заключение

Проведенное исследование показало, что системный анализ в задаче защиты от вредоносных программ полезен как практический способ собрать архитектуру из нескольких взаимосвязанных решений. На материале BODMAS удалось проследить, как меняется итог работы системы при переходе от перемешанной проверки к хронологической, при сокращении признакового пространства, при введении второго контура и при смене режима обновления первого контура. Перемешанное разбиение давало более спокойную и формально удобную картину, тогда как разбиение по времени лучше отражало эксплуатационный режим, в котором модель обучается на ранних данных и затем сталкивается с более поздними объектами. На таком фоне особенно хорошо видно, какие решения действительно сохраняют устойчивость, а какие чувствительны к сдвигу состава вредоносных семейств и изменению распределения признаков.

Сокращение признакового описания до 256 признаков дало тот баланс, ради которого и вводился системный анализ проектных альтернатив. Потеря качества по сравнению с полной моделью оказалась минимальной, зато время обучения и средняя стоимость обработки одного объекта заметно уменьшились. На этой основе был сформирован первый контур двухконтурной архитектуры, а полный набор из 2381 признака остался только для углубленного анализа наиболее сомнительных объектов. При передаче на второй контур 0,25 % потока двухконтурная схема сохранила качество полной модели и почти не вышла за пределы времени компактной одноконтурной схемы.

Сценарии обновления первого контура подтвердили, что устойчивость системы во времени нельзя считать автоматическим свойством даже у сильной архитектуры. Периодическое и адаптивное обновление выровняли помесячную динамику качества и уменьшили долю ложных срабатываний по сравнению с режимом без обновления. Практический смысл полученных результатов состоит в том, что проектирование интеллектуальной системы защиты должно идти от связки качество – нагрузка – обновляемость вместо максимизации одной метрики на статической выборке.

Из ограничений стоит отметить, что исследование выполнено на одном наборе данных и в рамках статического признакового описания исполняемых файлов, поэтому дальнейшая работа может быть связана с проверкой архитектуры на других временных

срезах, на более выраженном дрейфе данных и на комбинированных признаках, включающих поведенческие характеристики. В качестве еще одного продолжения можно рассмотреть более точную настройку правила передачи объекта на второй контур, где можно варьировать не только долю передаваемых экземпляров, но и сам критерий сомнительности решения.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Guerra-Manzanares A. Machine Learning for Android Malware Detection: Mission Accomplished? A Comprehensive Review of Open Challenges and Future Perspectives. *Computers & Security.* 2024;138:103654. https://doi.org/10.1016/j.cose.2023.103654
2. Yang L., Ciptadi A., Laziuk I., et al. BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware. In: *2021 IEEE Security and Privacy Workshops, 27 May 2021, San Francisco, CA, USA*. IEEE; 2021. P. 78–84. https://doi.org/10.1109/SPW53761.2021.00020
3. Jiang Y., Li G., Li Sh., et al. BenchMFC: A benchmark dataset for trustworthy malware family classification under concept drift. *Computers & Security.* 2024;139:103706. https://doi.org/10.1016/j.cose.2024.103706
4. Fernando D.W., Komninos N. FeSAD ransomware detection framework with machine learning using adaption to concept drift. *Computers & Security.* 2024;137:103629. https://doi.org/10.1016/j.cose.2023.103629
5. Li A.Sh., Iyengar A., Kundu A., et al. Revisiting Concept Drift in Windows Malware Detection: Adaptation to Real Drifted Malware with Minimal Samples. In: *32nd Annual Network and Distributed System Security Symposium, 24–28 February 2025, San Diego, CA, USA*. The Internet Society; 2025. https://doi.org/10.14722/ndss.2025.240830
6. Maniriho P., Mahmood A.N., Chowdhury M.J.M. MeMalDet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations. *Computers & Security.* 2024;142:103864. https://doi.org/10.1016/j.cose.2024.103864
7. Augello A., De Paola A., Lo Re G. Hybrid multilevel detection of mobile devices malware under concept drift. *Journal of Network and Systems Management.* 2025;33(2):36. https://doi.org/10.1007/s10922-025-09906-3
8. Liu Zh., Wang R., Peng B., et al. LDCDroid: Learning data drift characteristics for handling the model aging problem in Android malware detection. *Computers & Security.* 2025;150:104294. https://doi.org/10.1016/j.cose.2024.104294
9. Geurts P., Ernst D., Wehenkel L. Extremely randomized trees. *Machine Learning.* 2006;63(1):3–42. https://doi.org/10.1007/s10994-006-6226-1
10. Ke G., Meng Q., Finley Th., et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, 04–09 December 2017, Long Beach, CA, USA*. 2017. P. 3146–3154.
11. Lundberg S.M., Lee S.-I. A Unified Approach to Interpreting Model Predictions. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, 04–09 December 2017, Long Beach, CA, USA*. 2017. P. 4765–4774.
12. Ceschin F., Botacin M., Gomes H.M., et al. Fast & Furious: On the modelling of malware detection as an evolving data stream. *Expert Systems with Applications.* 2023;212:118590. https://doi.org/10.1016/j.eswa.2022.118590

13. Botacin M., Gomes H. Towards more realistic evaluations: The impact of label delays in malware detection pipelines. *Computers & Security.* 2025;148:104122. https://doi.org/10.1016/j.cose.2024.104122
14. Augello A., De Paola A., Lo Re G. M2FD: Mobile malware federated detection under concept drift. *Computers & Security.* 2025;152:104361. https://doi.org/10.1016/j.cose.2025.104361

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Абедалхуссайн Ахмед Али, аспирант, Национальный исследовательский технологический университет «МИСИС», Москва, Российская Федерация.
e-mail: m2000009@edu.misis.ru
ORCID: 0009-0004-5065-398X

Ahmed A. Abedalhussain, Postgraduate, National University of Science and Technology "MISIS", Moscow, the Russian Federation.

Ляпунцова Елена Вячеславовна, доктор технических наук, профессор, Национальный исследовательский технологический университет «МИСИС», Московский государственный технический университет им. Н.Э. Баумана, Москва, Российская Федерация.
e-mail: liapuntsova.ev@misis.ru
ORCID: 0000-0002-3420-3805

Elena V. Lyapuntsova, Doctor of Engineering Sciences, Professor, National University of Science and Technology "MISIS", Bauman Moscow State Technical University, Moscow, the Russian Federation.

Статья поступила в редакцию 30.04.2026; одобрена после рецензирования 17.08.2026; принята к публикации 24.08.2025.

The article was submitted 30.04.2026; approved after reviewing 17.08.2026; accepted for publication 24.08.2026.