Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники (ТУСУР)

На правах рукописи

Павлычев Алексей Викторович

МЕТОДИКА ОБНАРУЖЕНИЯ КОМПЬЮТЕРНЫХ АТАК НА ОСНОВЕ АНАЛИЗА СИСТЕМНЫХ СОБЫТИЙ ОПЕРАЦИОННОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

2.3.6 – Методы и системы защиты информации,
информационная безопасность

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
член-корреспондент РАН,
доктор технических наук,
профессор, А.А. Шелупанов

Томск 2026

Содержание

Введение 4

1 Выявление компьютерных атак с использованием методов машинного обучения 11

1.1 Общая характеристика различных видов компьютерных атак 11

1.2 Обзор методов обнаружения компьютерных атак 22

1.3 Использование алгоритмов машинного обучения для обнаружения компьютерных атак 26

1.4 Методика выявления компьютерных атак с использованием алгоритмов машинного обучения 48

1.5 Выводы по главе 65

2 Анализ системных событий операционной системы 67

2.1 Исследование видов и содержания событий операционной системы Microsoft Windows 67

2.2 Анализ событий операционной системы на предмет выявления признаков компьютерных атак 81

2.3 Формальная модель компьютерной атаки 85

2.4 Перспективы адаптации под отечественные операционные системы в рамках импортозамещения 87

2.5 Выводы по главе 89

3 Формирование набора данных из записей системных событий 90

3.1 Алгоритм моделирования компьютерных атак и формирования набора данных 90

3.2 Реализация разработанного алгоритма 101

3.3. Описание полученного набора данных 104

3.4 Выводы по главе 112

4 Апробация разработанной методики выявления компьютерных атак 113

4.1 Преобразование набора данных и выбор метрик 113

4.2 Обучение классификаторов и оценка результатов 118

4.3 Интерпретация результатов и программная реализация 126

4.4 Выводы по главе 139

Заключение 140

Список используемой литературы 143

Приложение А Свидетельство о регистрации программы для ЭВМ 155
Приложение Б Акты внедрения и использования результатов диссертационной работы..... 156

Введение

Современные геополитические вызовы диктуют новые требования к киберустойчивости цифровых сервисов, предназначенных для полноценного обеспечения интересов личности, общества и государства.

Указом Президента Российской Федерации от 2 июля 2021 г. № 400 утверждена Стратегия национальной безопасности Российской Федерации[1]. Информационная безопасность впервые выделена в качестве одного из стратегических национальных приоритетов, направленных на обеспечение и защиту национальных интересов Российской Федерации. В стратегии целью обеспечения информационной безопасности определено укрепление суверенитета Российской Федерации в информационном пространстве.

В условиях фактически объявленной России кибервойны наибольшую опасность для цифрового суверенитета представляют злоумышленники с высоким потенциалом - профессиональные политически мотивированные хакеры из недружественных государств.

Согласно изученным аналитическим отчетам [1-6], в 2023-2025 годах наблюдается значительный рост инцидентов информационной безопасности. Сложившаяся политическая обстановка говорит о дальнейшем росте хакерских атак на российскую информационную инфраструктуру с использованием новых видов кибероружия.

Взрывной рост технологий искусственного интеллекта, а также наличие большого количества материалов в открытом доступе делает хакерские атаки все более изощренными, при этом зачастую не требуя от их организаторов глубоких познаний в сфере используемых технологий.

Своевременное и качественное выявление деструктивных действий злоумышленников требует внедрения новых интеллектуальных инструментов.

---

[1] Стратегия доступна по ссылке http://www.kremlin.ru/acts/bank/47046

Проблемам выявления компьютерных атак, обнаружения вторжений и применения методов машинного обучения в системах обеспечения информационной безопасности посвящено значительное количество исследований российских и зарубежных ученых.

Существенный вклад в развитие теоретических и практических аспектов защиты информации, обнаружения компьютерных атак и построения систем мониторинга безопасности внесли российские исследователи А.И. Гетьман, М.Н. Горюнов, А.С. Шабуров, Ш.Г. Магомедов, И.В. Котенко, О.И. Шелухин, Д.И. Раковский, С.А. Коноваленко, М.В. Буйневич, К.Е. Израилов и др.

Вопросы анализа сетевого трафика, обнаружения аномалий, применения машинного обучения в задачах кибербезопасности получили развитие в работах зарубежных исследователей R. Lippmann, S.K. Sahu, N. Moustafa, A. Khraisat, M. Ghurab, T. Mehmood, F. Jemili, S. J. Yang, R.U. Khan, W. L. Al-Yaseen, S. M. Kasongo, A.M. Chandrashekhar, D.M. Farid, C. Smiliotopoulos, G. Kambourakis и др.

Анализ научных публикаций показывает, что большинство существующих подходов ориентировано преимущественно на исследование сетевого трафика и использование готовых наборов данных, которые обладают рядом ограничений, связанных с устареванием сценариев атак, недостаточным учетом современных угроз и ограниченным использованием системных событий операционной системы.

Недостаточно исследованными остаются вопросы формирования наборов данных на основе системных событий операционной системы, построения формальных моделей компьютерных атак с учетом полного спектра системных событий.

Несмотря на значительное количество научных работ в области обнаружения компьютерных атак, задача разработки методики выявления компьютерных атак на основе анализа системных событий операционной

системы с использованием алгоритмов машинного обучения остается актуальной и требует дальнейшего исследования.

Целью исследования является создание методики выявления компьютерных атак с включением анализа системных событий операционной системы методами машинного обучения.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Разработка методики выявления компьютерных атак с использованием алгоритмов машинного обучения при обработке системных событий операционной системы;
2. Проведение анализа видов и содержания событий операционной системы с целью построения формальной модели компьютерной атаки;
3. Модернизация алгоритма моделирования компьютерных атак и формирования набора данных из записей системных событий операционной системы;
4. Проведение экспериментального исследования разработанной методики выявления компьютерных атак, подтверждение достоверности и оценка результатов работы методики.

Объектом исследования являются компьютерные системы, подвергшиеся несанкционированному воздействию (компьютерным атакам).

Предметом исследования выступает процесс обнаружения компьютерных атак с использованием методов анализа данных, основанных на записях системных событий операционной системы.

Основные методы исследования, примененные в диссертационной работе – это аналитические методы моделирования, системного анализа, теории защиты информации, методы машинного обучения.

Научная новизна результатов работы и проведенных исследований:

1. Предложена методика обнаружения компьютерных атак с проведением анализа записей системных событий операционной системы

методами машинного обучения, отличающаяся от известных методик, основанных на анализе сигнатур и сетевого трафика;

2. Разработана формальная модель компьютерных атак, основой которой является учет записей системных событий операционной системы. Модель отличается от известных полным охватом всех возможных системных событий;

3. Предложен алгоритм моделирования компьютерных атак и формирования набора данных из системных событий операционной системы, который в отличие от существующих подходов автоматизирует процесс сбора данных и обеспечивает полное покрытие тактик MITRE ATT&CK, а также учитывает валидацию выполнения скриптов и передачу метаданных.

Достоверность и обоснованность предлагаемых научных положений, результатов и выводов работы подкрепляется разносторонним изучением современного состояния предметной области, системным обоснованием предложенных моделей, не противоречащих известным положениям других авторов, проведением эксперимента, подтверждающего результаты теоретических изысканий, апробацией полученных результатов в научных публикациях и докладах на международных и российских научных и научно-практических конференциях, а также практикой внедрения результатов исследования.

Научная значимость работы состоит в развитии теории и методологии обеспечения информационной безопасности в части создания новых способов обнаружения признаков несанкционированного воздействия на информационную систему с использованием современных аналитических инструментов.

Практическая значимость результатов исследования состоит в расширении возможностей выявления признаков компьютерных атак с использованием анализа системных событий операционной системы.

Реализация результатов работы. Работа выполнена при поддержке Министерства образования и науки Российской Федерации в рамках гранта № 40469-23-2021-К («Грант ИБ МТУСИ»).

Положения, выносимые на защиту:

1. Применение предложенной методики обнаружения компьютерных атак с включением анализа системных событий операционной системы методами машинного обучения позволяет достичь точность обнаружения 0,9965, что на 8% выше точности сигнатурных методов и более чем на 6% выше точности классификаторов, обученных на сетевом трафике.

Паспорт специальности, пункт 6: методы, модели и средства мониторинга, предупреждения, обнаружения и противодействия нарушениям и компьютерным атакам в компьютерных сетях;

2. Разработанная формальная модель компьютерных атак, основанная на анализе системных событий операционной системы, обеспечивает универсальность и позволяет описывать признаки атак без использования дополнительных инструментов мониторинга.

Паспорт специальности, пункт 3: методы, модели и средства выявления, идентификации и классификации угроз нарушения информационной безопасности объектов различного вида и класса;

3. Алгоритм моделирования компьютерных атак и формирования наборов данных из системных событий операционной системы позволяет расширить спектр выявляемых тактик MITRE ATT&CK в 4 раза по сравнению с существующими подходами, обеспечить полное покрытие угроз, а также повысить достоверность данных за счет валидации выполнения сценариев и передачи метаданных.

Паспорт специальности, пункт 6: методы, модели и средства мониторинга, предупреждения, обнаружения и противодействия нарушениям и компьютерным атакам в компьютерных сетях.

Апробация работы. Основные и промежуточные результаты исследования докладывались и обсуждались на следующих конференциях:

1. II Всероссийская научная конференция (с приглашением зарубежных ученых) «Фундаментальные проблемы информационной безопасности в условиях цифровой трансформации» (Ставрополь, 2020);
2. Межвузовская научно-теоретическая конференция (в рамках Сибирского форума «Информационная безопасность – 2021») (Новосибирск, 2021);
3. Международная научно-методическая конференция «Интеграция образования, науки, бизнеса и власти» (НМК ТУСУР-2022) (Томск, 2022)
4. XVIII Всероссийская научно-практическая конференция (в рамках IX Пленума регионального отделения Федерального учебно-методического объединения в системе высшего образования по укрупненной группе специальностей и направлений подготовки 10.00.00 «Информационная безопасность» по Сибирскому и Дальневосточному федеральным округам (СибРОУМО))(Хабаровск, 2022);
5. II Всероссийская научная школа-семинар «Современные тенденции развития методов и технологий защиты информации» (Москва, 2022);
6. Международная научно-техническая конференция «Автоматизация» (RusAutoCon-2025) (Сириус, 2025).

Внедрение результатов. Результаты диссертационной работы внедрены в деятельность центра мониторинга ООО «Информационный центр», а также в учебный процесс Дальневосточного федерального университета.

Личный вклад. В диссертационной работе использованы результаты, в которых автору принадлежит определяющая роль. Постановка задачи исследования и верификация результатов в процессе выполнения работы осуществлялась научным руководителем член-корреспондентом РАН, д.т.н., профессором Шелупановым А.А.

Публикации по теме диссертации. По материалам исследования опубликовано 11 работ, в том числе 7 работ в изданиях, рекомендованных ВАК России. Получено 1 свидетельство о регистрации программы для ЭВМ.

Структура и объем работы. Диссертация содержит введение, 4 главы, заключение и список источников из 109 наименований. Объем работы: 157 страниц, в том числе 12 таблиц и 47 рисунков.

Благодарности. Автор выражает благодарность председателю ФУМО ВО ИБ Белову Е.Б. и председателю Сибирского регионального отделения УМО ВО ИБ Шелупанову А.А за поддержку и мотивацию для занятия научной деятельностью; научному коллективу Дальневосточного федерального университета и лично Артемьевой И.Л. и Гончаровой С.Н., которые терпеливо обсуждали ход работы и давали ценные советы.

1 Выявление компьютерных атак с использованием методов машинного обучения

1.1 Общая характеристика различных видов компьютерных атак

В условиях стремительной цифровизации всех сфер жизни общества проблема кибербезопасности становится одной из наиболее значимых. Компьютерные атаки представляют собой серьезную угрозу для государственных, корпоративных и частных информационных систем, что делает исследования в области их обнаружения и ликвидации крайне актуальными.

Традиционные системы защиты, такие как антивирусы и межсетевые экраны, зачастую не справляются с современными угрозами. Необходим поиск новых подходов к выявлению и ликвидации последствий современных кибератак.

Компьютерная атака – это целенаправленное несанкционированное воздействие на информацию, на ресурс автоматизированной информационной системы или получение несанкционированного доступа к ним с применением программных или программно-аппаратных средств. Сетевая атака является разновидностью компьютерной атаки, в которой используются протоколы межсетевого взаимодействия[2].

В Российской Федерации неправомерный доступ к охраняемой законом компьютерной информации, если это деяние повлекло уничтожение, блокирование, модификацию либо копирование компьютерной информации, является уголовно-наказуемым деянием[3].

Отдельно квалифицируется как более тяжкое преступление причинение вреда в результате неправомерного доступа к охраняемой компьютерной

---

[2] Национальный стандарт РФ ГОСТ Р 51275-2006 «Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положения» (утв. приказом Федерального агентства по техническому регулированию и метрологии от 27 декабря 2006 г. N 374-ст)

[3] Статья 272 Уголовного кодекса РФ от 13.06.1996 N 63-ФЗ (ред. от 28.02.2025)

информации, содержащейся в критической информационной инфраструктуре Российской Федерации[4].

Несмотря на нарушение закона Российской Федерации злоумышленники осуществляют различные виды компьютерных атак для достижения своих корыстных или политических целей.

Первые компьютеры появились в начале 1940-х годов. В то время не существовало Интернета и возможности использования компьютеров были ограничены. Поскольку обмена информацией между компьютерами не происходило, угрозы и атаки на них отсутствовали.

В 1950-х годах зародился «телефонный фрикинг» (англ. «phone phreaking») – взлом телефонных систем для совершения бесплатных звонков или снижения стоимости междугородней связи. Многие телефонные компании не могли противостоять этому явлению и несли значительные убытки. Позже аналогичные методы злоумышленников легли в основу взлома компьютерных систем.

Термин «хакерство» применительно к компьютерам возник в 1960-х. В 1965 году была обнаружена первая уязвимость в системе IBM 7094 CTSS. В 1967 году IBM привлекла студентов для тестирования нового компьютера. Они изучали исходный код системы и получили доступ ко всем исходным данным, что стало первым примером этичного хакинга и доказательством наличия уязвимостей в компьютерных системах.

Первые кибератаки зафиксированы в 1970-х с появлением ARPANET – первой сети с пакетной коммутацией. В 1971 году Боб Томас создал первый вирус Creeper, способный распространяться по ARPANET. В ответ Рэй Томлинсон разработал Reaper – первую антивирусную программу для удаления Creeper. В 1979 году хакер Кевин Митник был впервые арестован за киберпреступления [7].

---

[4] Статья 274.1 Уголовного кодекса РФ от 13.06.1996 N 63-ФЗ (ред. от 28.02.2025)

В 1980-х участились атаки с использованием компьютерных вирусов, а термин «кибершпионаж» вошёл в обиход в контексте угроз со стороны других государств. В 1985 году Минобороны США выпустило «Критерии определения безопасности компьютерных систем» (англ. Trusted Computer System Evaluation Criteria, TCSEC, «Оранжевая книга») - первое руководство по компьютерной безопасности. В 1987 году появилось первое коммерческое антивирусное программное обеспечение [8].

В 1990-х рост числа компьютеров и Интернета сопровождался распространением компьютерных вирусов. В 1996 году появились макровирусы, а в конце десятилетия вирусы Melissa и ILOVEYOU заразили миллионы компьютеров.

В 2000-х Интернет и компьютеры стали повсеместными, что привело и к росту киберпреступности. Появились первые хакерские группировки, а вредоносное программное обеспечение, такое как черви и трояны стали основными инструментами атак. Например, в 2004 году червь MyDoom осуществлял массированные DDoS-атаки, а в 2007 троян Zeus крал учётные данные банковских систем по всему миру через спам.

В 2010-х хакеры активно эксплуатировали уязвимости программного обеспечения. В 2016 году вирус Mirai атаковал IoT-устройства, а WannaCry (2017) и LockerGoga заразили системы в 150 странах. В 2020 году CovidLock шифровал данные на Android.

В 2020-х взлом компьютерных систем стал массовым: появилось такое понятие как «взлом как услуга» (англ. Hacking-as-a-Service, HaaS). Крупные компании и государства из-за высоких финансовых и репутационных рисков инвестируют в кибербезопасность [9].

Главная причина быстрого увеличения количества кибератак кроется в архитектуре компьютерных систем и сетей связи. Уязвимости, недостатки и ошибки в конфигурациях аппаратного и программного обеспечения, а также в компьютерных сетях делают их потенциальными целями хакерских атак.

Атаки, возникающие из-за аппаратных ошибок, сложнее предотвратить, поскольку применяемые программные инструменты недостаточно эффективны для их обнаружения и предотвращения. Зачастую причиной аппаратных атак являются троянские программы. Эти вредоносные программы вызывают чрезмерное использование ресурсов компьютера, снижают производительность и могут привести к отключению системы из-за перегрузки источника питания. Кроме того, незаконное копирование аппаратных компонентов и использование ненадежных деталей, приобретенных в Интернете, может привести к созданию бэкдоров в компьютерной системе [10].

Сложность взаимодействия аппаратных компонентов и интегральных схем затрудняет обнаружение уязвимостей, связанных с оборудованием. Изменение всего одной интегральной схемы может повлиять на множество компонентов и долго оставаться незамеченным. Поэтому обнаружение и реагирование на кибератаки, использующие аппаратные уязвимости, представляет собой сложную задачу. Для предотвращения таких атак были предложены различные методы, включая защищенное от взлома оборудование, аппаратные водяные знаки и обфускацию, которые могут играть важную роль в защите систем [11].

Большинство кибератак по-прежнему обусловлены ошибками, уязвимостями и недостатками в прикладном программном обеспечении. Количество таких уязвимостей и ошибок увеличивается с каждым днем [12]. Основные причины уязвимостей и ошибок в программном обеспечении включают:

Ошибки проверки ввода;

Проблемы с управлением доступом пользователей;

Ошибки при аутентификация;

Переполнение буфера;

Проблемы, связанные с запросами SQL (англ. Structured Query Language);

Межсайтовый скриптинг (XSS, англ. Cross-Site Scripting);

Использование компонентов с известными уязвимостями;

Проблемы с веб-сервисами и API (англ. Application Programming Interface);

Недостаточное тестирование безопасности программного обеспечения.

Программное обеспечение разрабатывается быстро, но тестирование безопасности часто оказывается недостаточным как на этапе разработки, так и на этапе внедрения. Использование приложений на различных платформах и устройствах создает дополнительные уязвимости, увеличивающие количество атак, связанных с программным обеспечением. Усугубляет проблему и недостаток знаний у команд разработчиков о принципах безопасной разработки.

Рекомендуемым способом исправления ошибок и недостатков является обновление программного обеспечения. Однако во время обновления ошибки и уязвимости не всегда устраняются, а в некоторых случаях новые обновления вызывают и дополнительные проблемы. Наиболее эффективный способ минимизировать атаки на основе программного обеспечения - создавать проект программы и требования по безопасности еще до написания кода. Поэтому крайне важно, чтобы разработчики программного обеспечения проходили обучение процессам безопасной разработки. Угрозы и атаки следует учитывать во время разработки, а все блоки кода необходимо тестировать вручную и автоматически на каждом этапе. Например, Microsoft сделала обязательным использование этапов Security Development Lifecycle (SDL), что значительно сократило количество ошибок и уязвимостей в процессе разработки [13].

Во время передачи данных через Интернет хакеры могут получить доступ к информации, изменить или полностью заменить данные. Основная причина таких угроз - использование устаревших сетевых протоколов и устройств без учета требований безопасности. Подавляющее большинство атак на компьютерные сети возникают из-за уязвимостей в таких протоколах,

как TCP (англ. Transmission Control Protocol), IP (англ. Internet Protocol), ARP (англ. Address Resolution Protocol), DHCP (англ. Dynamic Host Configuration Protocol) и DNS (англ. Domain Name System). Кроме того, из-за неправильной конфигурации сетевых устройств, включая коммутаторы, маршрутизаторы и точки беспроводного доступа, злоумышленники могут перехватывать информацию во время передачи данных [14].

С развитием технологий расширились возможности доступа к информации, однако одновременно возросла и сложность обеспечения информационной безопасности. Повсеместное внедрение информационных систем во все сферы жизни привело к расширению спектра кибератак. Сегодня кибератакам подвергаются различные сферы - от повседневной жизни до деятельности государственных учреждений, от экономики и коммерции до банковского сектора и здравоохранения.

Рассмотрим наиболее распространенные типы компьютерных атак:

1. Атаки с использованием социальной инженерии.

Атака заключается в манипулировании людьми с целью получения доступа к информации или системам [15]. Социальная инженерия, как и большинство кибератак, направлена на обход систем безопасности. Жертвами могут стать любые пользователи, однако наиболее уязвимы пожилые люди с ограниченными техническими знаниями, лица с минимальным социальным взаимодействием и склонные к импульсивному поведению.

2. Атаки на приложения.

Данный тип атак предполагает получение злоумышленниками несанкционированного доступа через уязвимости в коде приложений [16]. Несмотря на то, что некоторые языки программирования подвергаются атакам чаще других, уязвимости существуют как в проприетарном коде, так и в открытых фреймворках и библиотеках. Злоумышленники могут использовать различные уязвимости в коде, проблемы с устаревшими сертификатами и ошибки при аутентификации.

3. Атаки на криптографические системы.

Криптография обеспечивает защиту данных через процессы шифрования (преобразование открытого текста в шифрованный) и дешифрования (обратное преобразование). Криптографические атаки направлены на компрометацию криптосистем через выявление ошибок в коде, алгоритмах шифрования или системах управления ключами [17]. Они подразделяются на пассивные (несанкционированный доступ без изменения данных) и активные (модификация или несанкционированная передача информации).

4. Атаки с перехватом (англ. Hijacking).

Такие атаки предполагают захват контроля над компьютерными системами, программным обеспечением или сетевыми соединениями [18]. Основные виды включают: перехват информации в браузере, перехват сессии, DNS-спуфинг, перехват буфера обмена, IP-спуфинг.

5. Атаки на компьютерные сети.

Представляют собой манипуляции с информацией в компьютерных сетях с целью получения контроля над ними [19]. Позволяют останавливать системы, модифицировать данные, создавать ботнеты. Атака выполняется через передачу вредоносного кода или инструкций.

6. Фишинговые атаки.

Это один из наиболее распространенных видов кибератак [20]. Злоумышленники маскируются под доверенные источники (банки, госучреждения) с целью получения конфиденциальных данных. Ключевым методом защиты является обучение пользователей и проверка подлинности ресурсов.

7. Атаки с использованием вредоносного программного обеспечения.

Вредоносное программное обеспечение может выполнять различные функции: кражу информации, шпионаж, саботаж систем, вымогательство. Основные типы включают вирусы, червей, троянов и руткиты. Вредоносное программное обеспечение может долгое время оставаться незамеченным,

собирая информацию или выводя системы из строя, а также использоваться для шантажа [21].

8. Атаки с использованием ботов и ботнетов.

Боты – автоматизированные программы, способные выполнять как легитимные задачи, так и вредоносные действия [22]. Ботнеты представляют собой сети зараженных компьютеров под контролем злоумышленников. Такие сети часто используются для массовых атак (DDoS, рассылка спама).

9. Атаки на пароли.

Злоумышленники используют слабые пароли и публично доступную информацию [23]. Для противодействия таким кибератакам необходимо использовать сложные пароли и не публиковать личные данные в открытом доступе, а также использовать различные пароли для разных информационных систем.

10. Атака «человек посередине».

Один из старейших типов атак, при котором злоумышленник тайно вмешивается в коммуникацию между сторонами [24]. Особенно опасен в публичных Wi-Fi сетях из-за возможности перехвата незашифрованных данных. Основные меры безопасности включают: использование лицензионного программного обеспечения, установку антивирусных программ и межсетевых экранов, применение сложных паролей, осмотрительность при посещении веб-ресурсов, отказ от открытия вложений из неизвестных источников, использование виртуальных машин для сомнительных операций.

В зависимости от различных критериев можно привести различные подходы к классификации современных компьютерных атак, например:

1. Классификация по цели атаки:

- Атаки на конфиденциальность. Направлены на получение несанкционированного доступа к данным. Примерами таких атак являются фишинг, перехват трафика, атаки на криптографические системы;

- Атаки на целостность. Нацелены на изменение или повреждение данных. Сюда можно отнести SQL-инъекции, подмену данных, атаки типа «человек посередине»;
- Атаки на доступность: Целью является нарушение работы системы или сервиса. Основные виды атак данного вида: DDoS-атаки, атаки на переполнение буфера.

2. Классификация по методу реализации:

- Активные атаки. Предполагают прямое воздействие на систему. Например, внедрение вредоносного программного обеспечения, атаки на отказ в обслуживании;
- Пассивные атаки. Направлены на сбор информации без прямого воздействия на систему. В качестве примера можно привести прослушивание сети, анализ трафика.

3. Классификация по уровню воздействия:

- Атаки на уровне сети. Нацелены на сетевые протоколы и инфраструктуру. Сюда относятся ARP-спуфинг, ICMP-флуд, атаки на маршрутизацию;
- Атаки на уровне операционной системы. Данный вид атак направлен на уязвимости самой операционной системы, может осуществляться эксплуатация уязвимостей ядра, проводиться атаки на права доступа;
- Атаки на уровне приложений. Эксплуатируют уязвимости в программном обеспечении. Например, межсайтовый скриптинг, SQL-инъекции, эксплуатация уязвимостей в веб-приложениях.

4. Классификация по типу используемых уязвимостей:

- Атаки на аппаратное обеспечение. Нацелены на физические компоненты системы. Сюда относятся атаки через сторонние устройства (USB-устройства), атаки на микропроцессоры;
- Эксплуатация программных уязвимостей. Представляют собой использование ошибок в коде программ. К таким атакам можно отнести переполнение буфера, уязвимости нулевого дня (zero-day);

- Социальная инженерия. Вид атак, основанных на манипуляции пользователями для получения доступа.

5. Классификация по сложности обнаружения:

- Простые атаки, не требующие значительного уровня подготовки. Легко обнаруживаются стандартными средствами защиты, например, массовые DDoS-атаки, фишинг с низким уровнем подготовки;
- Сложные атаки, требующие продвинутых методов обнаружения. Это APT (англ. Advanced Persistent Threats) атаки с использованием так называемых стелс-технологий (технологий, скрывающих активность злоумышленника в сети жертвы).

6. Классификация по типу используемого вредоносного программного обеспечения:

- Вирусы и черви. Распространяются по системе и сети, нанося ущерб. Наиболее частый сценарий – использование вирусов-шифровальщиков, таких как WannaCry, а также троянов;
- Руткиты. Скрывают присутствие злоумышленника в системе, это атаки на уровне ядра операционной системы;
- Шпионское программное обеспечение. Собирает информацию без ведома пользователя. Сюда относятся различные кейлоггеры, программы для перехвата данных.

7. Классификация по уровню автоматизации:

- Ручные атаки. Требуют активного участия злоумышленника. Например, целевые атаки на конкретные системы;
- Автоматизированные атаки. Выполняются с использованием скриптов или ботов. Сюда можно отнести массовые сканирования сети, автоматические эксплойты.

8. Классификация по типу злоумышленника:

- Внешние атаки. Инициируются извне сети или системы. Обычно это различные хакерские группировки, киберпреступники;

- Внутренние атаки. Инициируются изнутри сети или системы. Сюда относятся действия недовольных сотрудников, утечки данных.

9. Классификация по типу технологий:

- Атаки на технологии искусственного интеллекта. Использование специально подготовленных запросов (промптов) для приведения системы в нештатное состояние или снятие заложенных автором ограничений;
- Атаки на блокчейн и криптовалюты. Данный вид атак нацелен на децентрализованные системы. В качестве примера - атака 51%, эксплуатация смарт-контрактов.

10. Классификация по времени воздействия:

- Мгновенные атаки. Проводятся быстро и наносят ущерб в короткий срок. Сюда относятся DDoS-атаки, эксплойты нулевого дня;
- Долгосрочные атаки. Направлены на длительное присутствие в системе. Такой вид атак характерен для различных APT-группировок.

Визуальное представление классификации компьютерных атак по различным критериям приведено на Рисунке 1.

Таким образом, современные компьютерные атаки представляют собой сложные и многоаспектные угрозы, которые сочетают в себе различные объекты воздействия и инструменты для достижения своих целей. Они могут быть направлены на нарушение конфиденциальности, целостности или доступности данных, а также использовать как технические уязвимости, так и методы социальной инженерии. Атаки часто включают в себя несколько этапов, таких как разведка, внедрение, эксплуатация и сокрытие следов, что делает их сложными для обнаружения и предотвращения.

Кроме того, злоумышленники активно применяют современные технологии, включая искусственный интеллект и машинное обучение, для автоматизации атак и повышения их эффективности. От специалистов по кибербезопасности требуется использование комплексных подходов, включающих не только традиционные методы защиты, но и передовые технологии анализа данных, мониторинга и прогнозирования угроз.

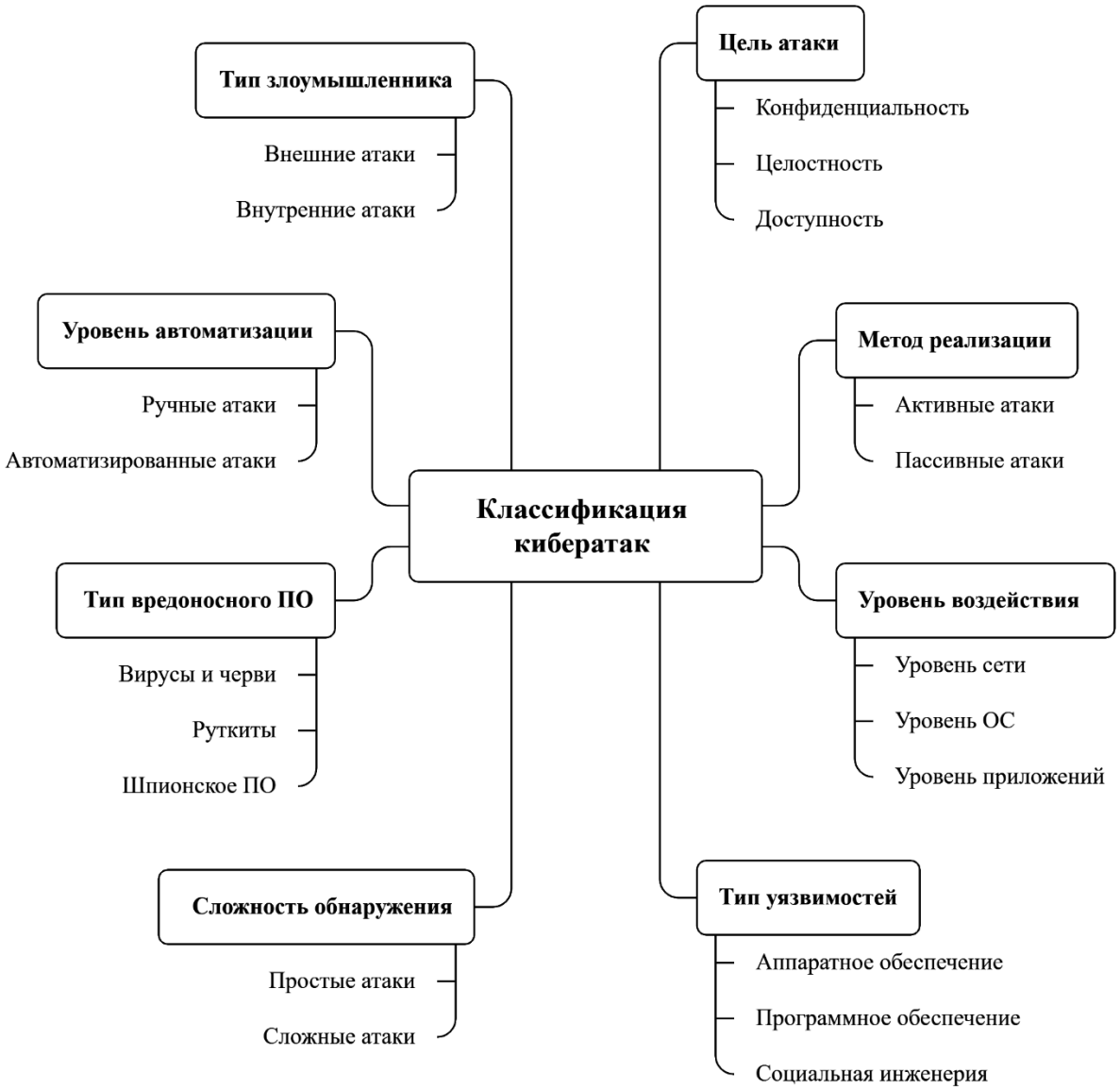

Рис. 1. Классификация компьютерных атак

Для успешного противодействия современным компьютерным атакам необходимо постоянно совершенствовать методы обнаружения и нейтрализации, а также учитывать их многообразие и сложность.

1.2 Обзор методов обнаружения компьютерных атак

Для противодействия различным компьютерным атакам специалисты по обеспечению информационной безопасности используют следующие основные методы и инструменты [25-26]:

1. Анализ сетевого трафика.

Анализ сетевого трафика – это процесс мониторинга и исследования данных, передаваемых по сети, с целью выявления подозрительной активности. Этот способ широко используется в системах обнаружения вторжений (IDS, англ. Intrusion Detection Systems) и предотвращения вторжений (IPS, англ. Intrusion Prevention Systems).

В рамках анализа сетевого трафика могут использоваться следующие подходы:

1.1. Сигнатурный анализ.

Сигнатурный анализ основан на сравнении сетевого трафика с заранее определенными шаблонами (сигнатурами), характерными для известных атак. Сигнатура может включать определенную последовательность байтов в пакете или специфический заголовок. Данный подход используется в таких системах как Snort или Suricata, для обнаружения атак, например, SQL-инъекции или сканирование портов.

1.2. Анализ аномалий.

Подход основан на создании модели «нормального» поведения сети и выявлении отклонений от нее. Например, резкое увеличение количества запросов к серверу может указывать на DDoS-атаку. Для анализа трафика и выявления аномалий широкое распространение получили методы машинного обучения.

1.3. Анализ протоколов.

Представляет собой проверку соответствия сетевого трафика стандартам протоколов (например, TCP/IP, HTTP). Атаки, эксплуатирующие ошибки в реализации протоколов, могут быть обнаружены через анализ нарушений стандартов. В качестве примера можно привести обнаружение атак, таких как TCP SYN Flood, через анализ заголовков пакетов.

1.4. Статистический анализ.

Использование статистических методов для выявления аномалий в трафике. Например, анализ частоты запросов или объема передаваемых данных.

2. Анализ системных событий (лог-файлов).

Системные события содержат данные об изменениях, происходящих в операционной системе и приложениях. Анализ лог-файлов позволяет выявить подозрительные действия, такие как несанкционированный доступ или изменения конфигураций.

Выделяют следующие основные типы событий:

События безопасности - содержат информацию о попытках входа, изменении прав доступа и других событиях, связанных с безопасностью (журналы Windows event log или Linux syslog);

События операционной системы - содержат информацию о запуске, обновлении, ошибках и другие данные о работе операционной системы;

События приложений – представляют собой записи о работе программного обеспечения, ошибках и предупреждениях (например журналы веб-серверов Apache, Nginx).

При работе с системными журналами используются следующие методы анализа [27]:

2.1. Ключевые слова и фильтры.

Представляет собой поиск в лог-файлах ключевых слов, связанных с атаками (например, «failed login», «root access»). Часто для анализа лог-файлов может использоваться утилита grep.

2.2. Корреляция событий.

Анализ взаимосвязей между событиями в разных журналах для выявления сложных атак с использованием SIEM-систем.

2.3. Машинное обучение для анализа лог-файлов.

Использование алгоритмов машинного обучения для автоматического анализа журналов и выявления аномалий. Могут применяться методы классификации для обнаружения атак.

3. Поведенческий анализ.

Обнаружение отклонений от нормального поведения системы или пользователя, что может указывать на атаку. Например, выявление аномальной активности пользователя, такой как несанкционированный доступ к данным. Для выявления отклонений предварительно создаются различные профили пользователей.

4. Анализ уязвимостей.

Поиск слабых мест в системе, которые могут быть использованы злоумышленниками. При данном подходе используется ручной поиск уязвимостей или применяются автоматизированные средства – сканеры уязвимостей.

Классификация методов выявления компьютерных атак приведена на Рисунке 2.

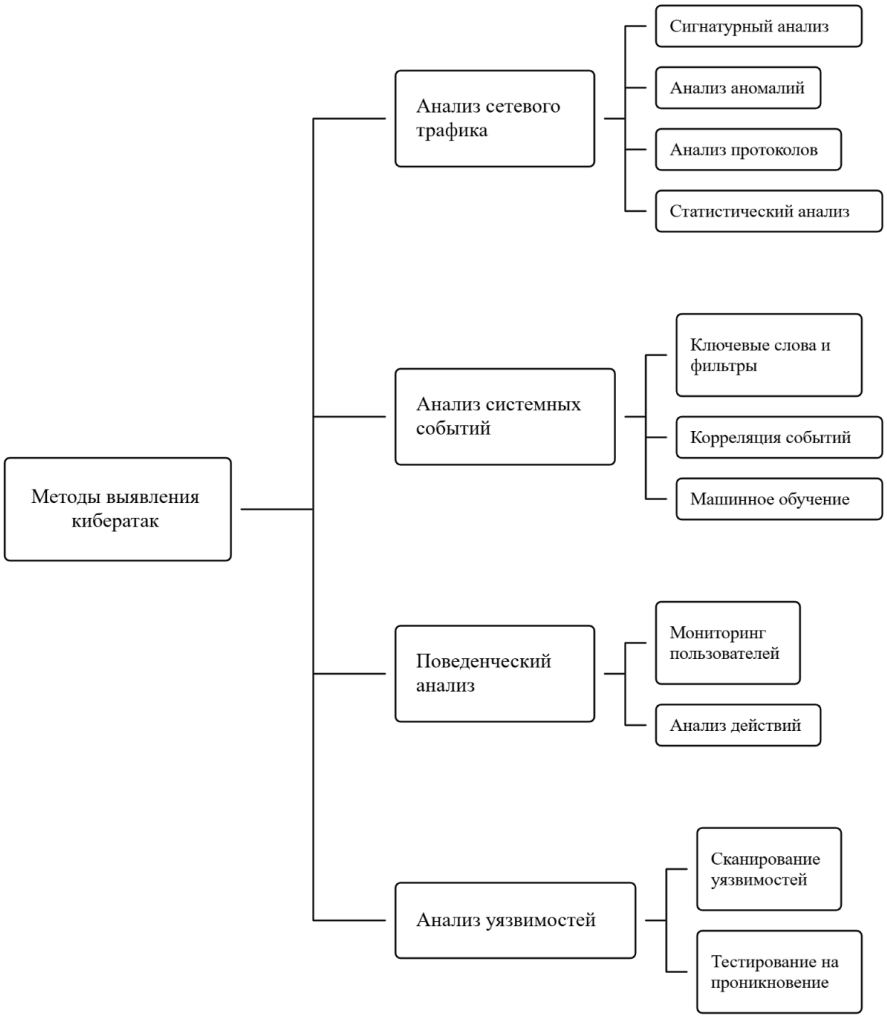

Рис. 2. Методы обнаружения компьютерных атак

В рамках настоящего исследования предлагается методика обнаружения компьютерных атак, включающая в себя автоматизированный сбор и анализ записей системных событий операционной системы, в отношении которой осуществлялись компьютерные атаки. Полученные результаты обрабатываются с использованием методов машинного обучения.

Рассмотрим различные методы машинного обучения в решении задач обеспечения кибербезопасности.

1.3 Использование алгоритмов машинного обучения для обнаружения компьютерных атак

Машинное обучение представляет собой метод анализа данных, который автоматизирует построение аналитических моделей. Это ветвь искусственного интеллекта, основанная на идее, что системы могут учиться на данных, выявлять закономерности и принимать решения с минимальным вмешательством человека [28].

Машинное обучение доказало свою эффективность в решении различных аналитических задач и все чаще используется для обнаружения угроз и автоматического устранения их, прежде чем они смогут нанести ущерб. Разработанные алгоритмы быстро сканирует большие объемы данных и анализирует их с помощью статистики.

Рассмотрим подробнее методы машинного обучения в задаче обнаружения компьютерных атак.

В машинном обучении первостепенное значение имеет набор данных. Прежде чем проводить какой-либо анализ, исследователь должен сам тщательно изучить и обработать набор имеющихся данных. В методах машинного обучения нельзя напрямую использовать необработанные данные, такие как захват пакетов (pcap), NetFlow и другие сетевые данные.

Данные должны быть предварительно обработаны, чтобы их можно было использовать в популярных инструментах машинного обучения, таких как R, RapidMiner, специальные библиотеки в Python, например Scikit-Learn. Исследователи, использующие анализ машинного обучения, должны понимать методологию сбора данных и методы, используемые при предварительной обработке данных.

Рассмотрим примеры наиболее распространенных наборов данных (датасеты, англ. datasets), используемых в машинном обучении при решении задач в области кибербезопасности.

The DARPA (Defense Advanced Research Project Agency – Агентство перспективных оборонных исследований) располагает двумя наборами данных, имеющих большую ценность для исследователей кибербезопасности. Набор данных DARPA 1998 и 1999 годов был разработан группой Cyber Systems and Technology лаборатории Линкольна Массачусетского технологического института (MIT/LL) [29].

Датасет DARPA 1998 построен на моделировании сетевых данных TCP/IP, данные собраны за 9 недель: 7 недель использовались для обучения системы, а оставшиеся 2 недели использовались для проверки системы [30]. В этом наборе данных определены четыре различных типа атак: отказ в обслуживании (DOS), повышение привилегий (U2R), получение удаленного доступа к системе (R2L) и сканирование сети и узлов (scan). Аналогичный набор данных был снова подготовлен с более смоделированными атаками, DARPA 1999 [31].

KDD 1999 – еще один известный набор данных, который в основном используется исследователями кибербезопасности (набор был специально подготовлен в рамках соревнований по анализу данных – The Third International Knowledge Discovery and Data Mining Tools Competition) [32]. Набор KDD 1999 имеет 41 подробный атрибут и очень похож на данные

NetFlow. Авторы статьи «A detailed analysis of the KDD Cup 1999 data set» [33] всесторонне изучили набор данных KDD 1999. Статистический анализ показал, что в наборе данных было огромное количество избыточных записей. Обнаружено, что 78 % обучающей выборки и 75 % тестовой выборки одинаковы. Эти врожденные проблемы делают KDD 1999 непригодным для использования, и поэтому предложен новый набор данных, названный NSL-KDD [34].

Исследователи методов обнаружения компьютерных атак часто используют одни и те же наборы данных, для оценки своих алгоритмов. Но эти наборы данных устарели и могут не содержать актуальных признаков современных видов атак. Поэтому в настоящий момент остро стоит проблема формирования наборов данных для использования в машинном обучении для решения актуальных задач кибербезопасности [35-38].

Рассмотрим некоторые популярные методы машинного обучения, применяемых при обнаружении компьютерных атак [39].

1.3.1. Байесовская сеть

Байесовская сеть – это направленный ациклический граф $G = \langle V, E \rangle$, в котором каждой вершине $v \in V$ поставлена в соответствие случайная величина X_v и каждое ребро $(u, v) \in E$ представляет прямую зависимость X_v от X_u. Пусть $parents(v) = u|(u, v) \in E$, тогда в Байесовской сети каждой вершине $v \in V$ графа должно быть сопоставлено распределение условных вероятностей от вершин из $parents(v)$, где *parents* – родительский узел [40].

Наивные методы Байеса – это набор алгоритмов обучения с учителем, основанных на применении теоремы Байеса с «наивным» предположением об условной независимости между каждой парой характеристик при заданном значении переменной класса.

Несмотря на свои чрезмерно упрощенные предположения, наивные Байесовские классификаторы довольно хорошо работают во многих реальных

ситуациях. Им требуется небольшой объем обучающих данных для оценки необходимых параметров.

Сеть разрабатывается как случайный набор переменных и их условных зависимостей через направленный ациклический граф. Узлы, представляющие потомка, зависят от родительских узлов, и каждый узел поддерживает состояния формы условной вероятности и случайной величины. На Рисунке 3 приведен пример обнаружения сигнатур атаки с использованием Байесовской сети. Каждое состояние может быть входом в другое состояние с определенным набором значений состояния. Каждое значение состояния, которое может перейти из одного состояния в другое, имеет соответствующую вероятность, и сумма этих вероятностей равна 1, представляющей весь набор значений состояния.

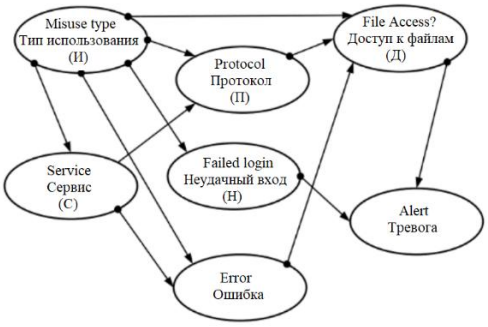

Рис. 3. Обнаружение сигнатур атак с помощью Байесовской сети

Байесовскую сеть можно использовать для обнаружения аномалий. В статье «A framework for an adaptive intrusion detection system using Bayesian network» [41] авторы описывают создание системы обнаружения вторжений с использованием Байесовской сети. Для моделирования системы использовался набор данных KDD 1999 с девятью его атрибутами. Точность 88% и 89% была достигнута в обычном сценарии и сценарии атаки при обучении. На тестовой выборке модель показала точность обнаружения для зондирования (probe) - 99%, сканирования (scan) - 21%, атаки отказ в обслуживании (DOS) - 89% и получения удаленного доступа к системе (R2L)

7%. Поскольку количество тренировочных случаев было очень низким в случае R2L, точность модели существенно пострадала.

1.3.2. Деревья решений

Дерево решений – логический алгоритм, решающий задачи классификации и регрессии. Представляет собой объединение логических условий в структуру дерева [42].

Цель алгоритма состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной, изучая простые правила принятия решений, выведенные из характеристик данных.

Иными словами, дерево решений – это алгоритм классификации $a(x) = (V_{внутр}, v_0, V_{лист}, S_v, \beta_v)$, задающийся деревом (связным ациклическим графом), где:

- $V = V_{\text{внутр}} \cup V_{\text{лист}}$ – множество вершин, $v_0 \in V$ – корень дерева;
- $S_v: D_v \to V_v$ – функция перехода по значению предиката в множество детей вершины v;
- $\beta_v: X \to D_v$ – предикат ветвления, $v \in V_{\text{внутр}}$ и $|D_v| < \infty$;

Для листьев $v \in V_{\text{лист}}$ определена метка класса $y_v \in Y$.

Дерево решений очень похоже на дерево. У деревьев есть листья, которые представляют различные классификации, а ветви являются ссылками или функциями, которые, в свою очередь, обеспечивают путь к классификациям. Дерево решений – это метод машинного обучения, основанный на рекурсивной древовидной структуре. Дерево решений состоит из элементов: корневого или промежуточного узла, пути и конечного узла, как показано на Рисунке 4. Корневой и промежуточный узел дерева представляет объект или атрибут. Каждый путь расхождения дерева представляет возможные значения родительского узла (объекта). Конечный узел соответствует прогнозируемой категории или классифицированному атрибуту. Результирующее дерево представляется в виде правил «ЕСЛИ-ТО».

Во время построения дерева меры энтропии и получения информации используются для дальнейшего выбора наилучшего промежуточного узла.

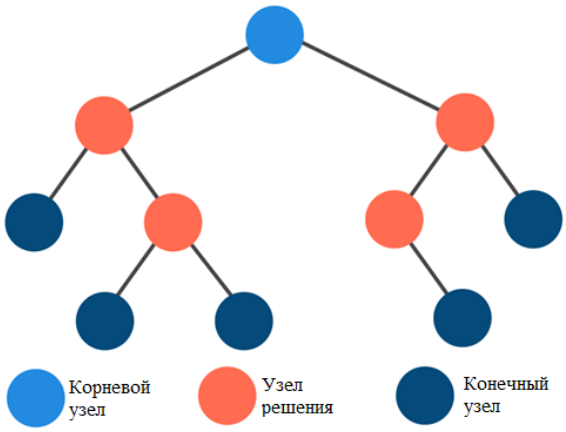

Рис. 4. Дерево решений

К преимуществам данного метода можно отнести:

- Простоту в понимании и интерпретации;
- Возможность визуализации;
- Небольшая предварительная подготовка данных;
- Возможна проверка модели с помощью статистических тестов.

К недостаткам деревьев решений можно отнести:

- Обучающиеся деревья решений могут создавать слишком сложные деревья, которые плохо обобщают данные (данное явление называется переобучением);
- Деревья решений могут быть нестабильными, поскольку небольшие изменения в данных могут привести к созданию совершенно другого дерева;
- Обучающиеся деревья решений создают предвзятые деревья, если некоторые классы доминируют.

Алгоритмы CART (Classification and Regression Trees), ID3 (Iterative Dichotomiser 3) и C4.5 – популярные алгоритмы для автоматического создания деревьев решений [43-45]. ID3 работает на основе жадного подхода, однако он не может обрабатывать числовые атрибуты. C4.5 является улучшенной версией ID3 и преодолевает ограничения ID3, решая проблему переобучения

с использованием методов обрезки деревьев. CART поддерживает как числовые, так и категориальные атрибуты и обрабатывает отсутствующие значения, которые не могут быть обработаны ID3. В современных библиотеках машинного обучения, таких как Scikit-learn, реализованы улучшенные версии CART, которые поддерживают многопоточность и оптимизированы для работы с большими объемами данных.

Данный метод используется в системе обнаружения и предотвращения вторжений SNORT. Процесс сравнения правил обработки входящего трафика идет медленно из-за большого количества сигнатур. Авторы статьи «Using decision trees to improve signature - based intrusion detection» [46] в своем эксперименте заменили 150 правил SNORT, используя вариант алгоритма ID3. Их цель состояла в том, чтобы заменить эти алгоритмы моделью дерева решений для увеличения скорости обработки. Метод кластеризации применяется для улучшения процесса сопоставления. Имея набор сигнатур (каждая из которых диктует ряд ограничений, которым входные данные должны соответствовать, чтобы активировать ее), алгоритм генерирует дерево решений, которое используется для поиска вредоносных событий с использованием минимального количества избыточных сравнений. Данный подход применялся к набору данных DARPA 1999. Скорость обработки и эффективность модели разработки сравнивали с анализом SNORT. Модель достигла максимального ускорения 105%. Для дальнейших экспериментов количество заменяемых правил было увеличено со 150 до 1581. Хотя авторы не приводят никаких количественных показателей, исследование выявило существенное ускорение с использованием алгоритма дерева решений, а время обработки правил резко сократилось.

1.3.3. Случайный лес

Случайный лес (англ. Random Forest) – один из примеров объединения классификаторов в ансамбль [47].

Итоговый классификатор может быть представлен в виде формулы $a(x) = \frac{1}{N}\sum_{i=1}^{N} t_i(x)$, где N – количество деревьев, $t_i(x)$ – решающее дерево. Для задачи классификации мы выбираем решение по большинству результатов, выданных классификаторами, а в задаче регрессии — по их среднему значению.

Таким образом, случайный лес – усреднение над решающими деревьями, при обучении которых для каждого разбиения признаки выбираются из некоторого случайного подмножества признаков.

Случайный лес в кибербезопасности применяется, например, для измерения объема спама и для обнаружения вторжений [48, 49]. Данный метод требует меньших вычислительных затрат на этапе обучения модели. Однако, поскольку случайный лес объединяет прогноз нескольких деревьев решений, необходимо выбрать деревья решений, которые следует учитывать в процессе прогнозирования.

Для тестирования метода для обнаружения вторжений использовался набор данных NSL-KDD. На данном наборе метод показал точность обнаружения вторжений 92,79 % [50].

1.3.4. K - ближайших соседей

K-ближайших соседей – это алгоритм обучения без учителя. Он основан на функции расстояния, которая измеряет различие двух экземпляров данных.

Принцип, лежащий в основе методов ближайших соседей, заключается в том, чтобы найти предопределенное количество обучающих выборок, ближайших по расстоянию до новой точки, и предсказать метку на их основе. Количество выборок может быть заданной пользователем константой (обучение k-ближайших соседей) или варьироваться в зависимости от локальной плотности точек (обучение соседей на основе радиуса). В общем случае расстояние может быть любой метрической мерой: наиболее распространенным выбором является стандартное евклидово расстояние.

Методы на основе соседей известны как не обобщающие методы машинного обучения, поскольку они просто «запоминают» все свои обучающие данные.

Несмотря на свою простоту, метод ближайших соседей успешно справился с большим количеством задач классификации и регрессии [51].

Пусть задана обучающая выборка пар «объект-ответ» $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$.

На множестве объектов задана функция расстояния $\rho(x, x')$, которая должна быть достаточно адекватной моделью сходства объектов. Чем больше значение этой функции, тем менее схожими являются два объекта x, x'.

Для произвольного объекта u расположим объекты обучающей выборки x_i в порядке возрастания расстояний до u: $\rho(u, x_{1;u}) \leq \rho(u, x_{2;u}) \leq \ldots \leq \rho(u, x_{m;u})$, где через $x_{i;u}$ обозначается тот объект обучающей выборки, который является i -м соседом объекта u. Аналогичное обозначение введём и для ответа на i -м соседе: $y_{i;u}$. Таким образом, произвольный объект u порождает свою перенумерацию выборки. В наиболее общем виде алгоритм ближайших соседей есть: $a(u) = argmax_{y \in Y} \sum_{i=1}^{m} [y_{i;u} = y] w(i, u)$, где $w(i, u)$ - заданная весовая функция, которая оценивает степень важности i -го соседа для классификации объекта u. Естественно полагать, что эта функция не отрицательна и не возрастает по i (поскольку чем дальше объект, тем меньший вклад он должен вносить в пользу своего класса).

По-разному задавая весовую функцию, можно получать различные варианты метода ближайших соседей.

$w(i, u) = [i = 1]$ – простейший метод ближайшего соседа;

$w(i, u) = [i < k]$ – метод k-ближайших соседей.

На обучение данного алгоритма уходит меньше времени, чем на другие. Однако время работы КБС является довольно затратным. На Рисунке 5 показана работа КБС. Этот классификатор работает на предположении, что

сходные точки данных в пространстве будут ближе друг к другу, чем непохожие. Значение k -й точки данных влияет на общую производительность алгоритма [52].

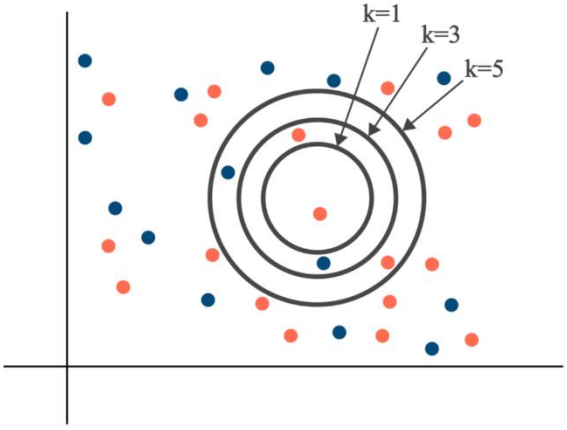

Рис. 5. K - ближайший сосед

Авторы статьи «A novel SVM-kNN-PSO ensemble method for intrusion detection system» [53] приводят результаты обнаружения аномалий в системе обнаружения вторжений методом k-ближайших соседей, которые составили от 75,7% до 99,8% при k =11.

Классификатор чувствителен к зашумленным данным и выбору функции расстояния для нахождения расстояния или разницы между точками данных. КБС требует достаточного объема памяти для манипуляций и требует значительных вычислительных ресурсов. Евклидово расстояние $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ – один из вариантов функций, используемых в качестве функции расстояния для вычисления расстояния между точками данных x и y.

1.3.5. Кластеризация

Кластеризация - это метод обучения без учителя, в котором мера сходства используется для группировки данных, поэтому исследователю не требуется явное описание различных классов атак.

Пусть X – множество объектов, Y – множество идентификаторов (меток) кластеров. На множестве X задана функция расстояния между объектами

$\rho(x, x')$. Дана конечная обучающая выборка объектов $X_m = \{x_1, \ldots, x_m\} \subset X$. Необходимо разбить выборку на подмножества (кластеры), то есть каждому объекту $x_i \in X_m$ сопоставить метку $y_i \in Y$, таким образом чтобы объекты внутри каждого кластера были близки относительно метрики ρ, а объекты из разных кластеров значительно различались.

Алгоритм кластеризации – функция $a: X \to Y$, которая любому объекту $x \in X$ ставит в соответствие идентификатор кластера $y \in Y$ [54].

Современные системы обнаружения вторжений не всегда могут распознать новую атаку методом сигнатурного анализа, так как сигнатуры нового вида атак (zero-day) могут быть еще не созданы и не загружены в базу решающих правил системы. Авторы статьи «Intrusion signature creation via clustering anomalies» [55] демонстрируют применение алгоритма кластеризации для создания сигнатуры атаки нулевого дня в реальном времени. Для проверки работоспособности предложенного алгоритма использовался набор данных KDD. Точность обнаружения неизвестных атак была достигнута уровня 70-80 %. Учитывая неизвестный характер атак, этот уровень точности является высоким.

1.3.6. Нейронные сети

Нейронная сеть строится по аналогии работы человеческого мозга. Сеть имеет структуру слоев, ввод данных активирует нейрон второго слоя сети, что, в свою очередь, выводит на следующий уровень иерархии и так далее. Вывод производится последним слоем сети. Функция, которая преобразует несколько входных параметров в один выходной называется искусственным нейроном.

Искусственный нейрон – это такая функция $R_n \to R$, которая преобразует несколько входных параметров в один выходной.

Как видно на Рисунке 6, у нейрона есть n входов x_i, у каждого из которого есть вес w_i, на который умножается сигнал, проходящий по связи. После этого взвешенные сигналы $x_i w_i$ направляются в сумматор, который

агрегирует все сигналы во взвешенную сумму. Эту сумму также называют *net*. Таким образом, $net = \sum_{i=1}^{n} w_i x_i = w^T x$.

Просто так передавать взвешенную сумму *net* на выход достаточно бессмысленно – нейрон должен ее как-то обработать и сформировать адекватный выходной сигнал. Для этих целей используют функцию активации, которая преобразует взвешенную сумму в какое-то число, которое и будет являться выходом нейрона. Функция активации обозначается $\varphi(net)$. Таким образом, выходом искусственного нейрона является $\varphi(net)$.

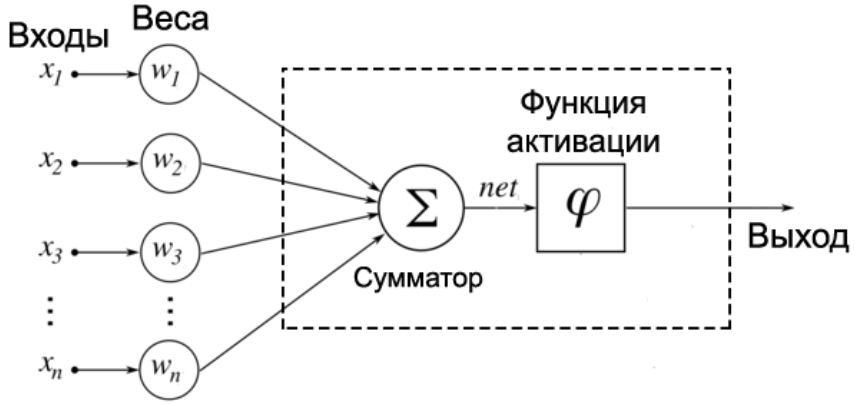

Рис. 6. Схема искусственного нейрона

Для разных типов нейронов используют самые разные функции активации [56].

Одним из основных недостатков нейронной сети является огромное время обучения из-за наличия локальных минимумов. Этот подход был распространен в середине девяностых годов, но из-за появления метода опорных векторов использование нейронный сетей стало уменьшаться [57]. С разработкой сверточных нейронных сетей популярность нейронных сетей вновь возросла.

Автор статьи «Artificial neural networks for misuse detection» [58] описывает модель нейронной сети, которая использует многокатегориальный классификатор для обнаружения аномалий. Данные классифицируются либо как обычный трафик, либо как вредоносный трафик. Для генерации данных использовался сетевой монитор RealSecure со встроенными сценариями атак.

Предварительная обработка данных выполнялась с использованием девяти выбранных параметров: код ICMP, тип ICMP, адрес источника, адрес назначения, номер протокола, порт источника, порт назначения, длина необработанных данных и тип необработанных данных. Точность алгоритма составила 93% на этапе тестирования.

Рекуррентные нейронные сети (РНС) – это вид нейронных сетей, в которых связи между элементами образуют направленную последовательность. РНС содержит скрытые состояния [59]. Каждое состояние использует выходные данные предыдущего состояния в качестве входных данных, как показано на Рисунке 7(a). Таким образом, информация циркулирует между состояниями в РНС. Основной целью РНС является обработка данных временных рядов и анализ потоков данных. РНС обладает памятью, что означает, что она хранит информацию из предыдущего опыта и позже использует ее в качестве входных данных для следующих состояний [60].

Сверточные нейронные сети (СНС) – специализированный тип многослойных нейронных сетей, которые являются расширением традиционных искусственных нейронных сетей с прямой связью [61]. Они состоят из трех видов слоев, в том числе одного или нескольких сверточных слоев, одного или нескольких полносвязных слоев и объединенных слоев, как показано на рисунке 7(b). ZFNet, GoogLeNet, ResNet, EfficientNet являются архитектурами СНС широко используемыми в соревнованиях по машинному зрению и распознаванию образов. СНС широко используется для распознавания изображений, поиска лекарств, обнаружения аномалий и многих других [62].

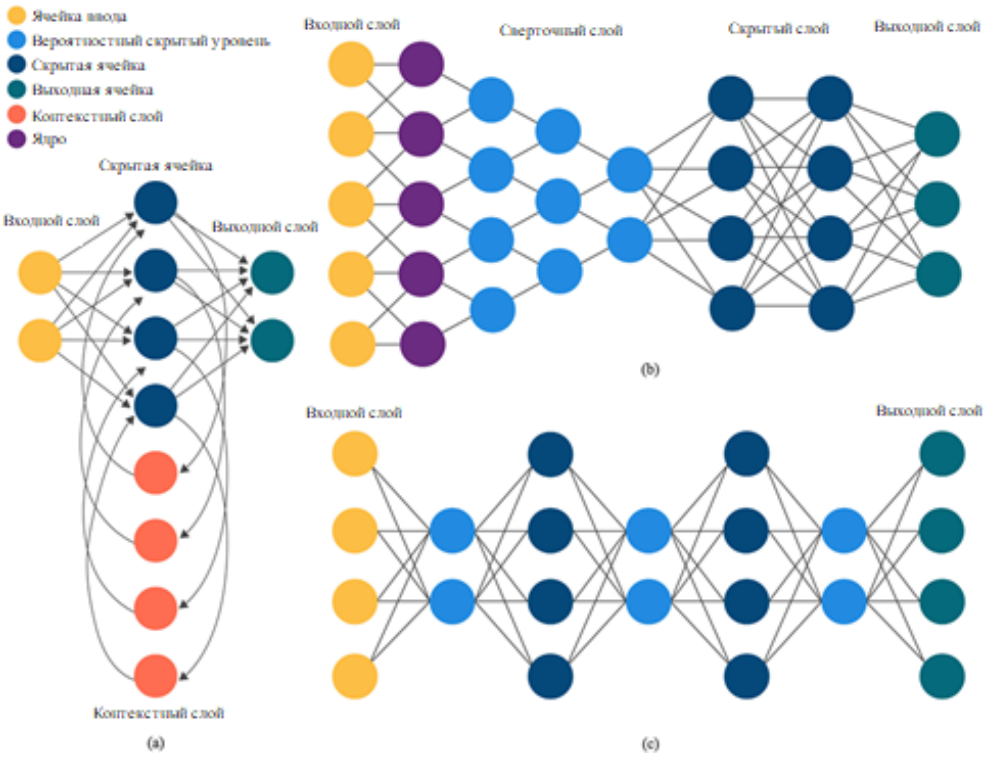

Рис. 7. Графическое представление различных нейронных архитектур:
(a) - Рекуррентная нейронная сеть (РНС),
(b) - Сверточная нейронная сеть (СНС),

Рассмотрим алгоритм свертки. Свертка – операция над парой матриц A (размера $n_x \times n_y$) и B (размера $m_x \times m_y$), результатом которой является матрица $C = A \times B$ размера $(n_x - m_x + 1) \times (n_y - m_y + 1)$. Каждый элемент результата вычисляется как скалярное произведение матрицы B и некоторой подматрицы A такого же размера (подматрица определяется положением элемента в результате). То есть, $C_{i,j} = \sum_{u=1}^{m_x-1} \sum_{v=0}^{m_y-1} (A_{i+u,j+v} B_{u,v})$.

Логический смысл свертки такой – чем больше величина элемента свертки, тем больше эта часть матрицы A была похожа на матрицу B (похожа в смысле скалярного произведения). Поэтому матрицу A называют изображением, а матрицу B – фильтром или образцом.

Авторы статьи «An improved convolutional neural network model for intrusion detection in networks» [63] предложил улучшенную версию СНС для обнаружения вторжений с точностью 99,23% с использованием набора данных

KDD99. CHC также широко используется для классификации вредоносного трафика [64].

1.3.7. Автоэнкодер

Автоэнкодер – это тип нейронной сети, который используется для неконтролируемого обучения. Основная задача автоэнкодера заключается в снижении размерности данных путем их сжатия и устранения шума, с последующим восстановлением исходной структуры данных. Автоэнкодер работает на основе принципа, согласно которому выходные данные должны максимально соответствовать входным данным, как представлено на Рисунке 8. Автоэнкодер состоит из двух частей:

- Энкодер: отвечает за сжатие входа в скрытом слое. Представлен функцией кодирования $h = g(x)$;
- Декодер: предназначен для восстановления ввода из скрытого слоя. Представлен функцией декодирования $x = f(h)$.

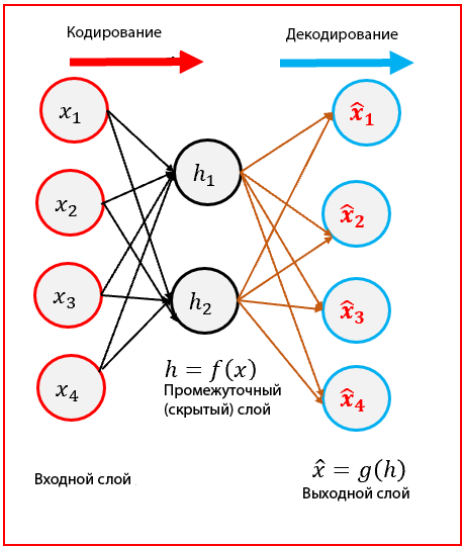

Рис. 8. Схема автоэнкодера

Таким образом, автоэнкодер описывают функцией $r = f(g(x))$, где r совпадает с изначальным x на входе.

Цель автоэнкодера – минимизировать разницу между входными данными *x* и восстановленными данными *r*, что обычно достигается за счет минимизации функции потерь, например, среднеквадратичной ошибки (MSE).

Многослойный автоэнкодер состоит из нескольких частей. Во-первых, кодировщик используется, чтобы научиться сжимать данные. Во-вторых – слой, который используется для хранения полностью сжатых данных. Более того, с помощью декодера модель учится выполнять реконструкцию данных. Наконец, в четвертой части потери при реконструкции измеряют, насколько результат близок к целевым значениям выхода [65].

Авторы статьи «A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework» [66] в своей работе приводят результаты работы нейронной сети в задаче обнаружения вторжений, которые составили от 70% до 100 % правильно классифицированных вторжений, в зависимости от типа проводимой атаки и рассматриваемого набора данных.

1.3.8. Метод опорных векторов

Метод опорных векторов (МОВ, SVM, англ. Support vector machine) представляет собой набор алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит к семейству линейных классификаторов.

Метод опорных векторов считается наиболее часто используемым и успешным методом машинного обучения для задач кибербезопасности, особенно для систем обнаружения вторжений. Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Рисунок 9 дает пиктографическое объяснение МОВ. Данный алгоритм работает в предположении, что чем больше расстояние между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора. Точки данных, лежащие на

границе гиперплоскости называются опорными векторными точками. МОВ подразделяются на две основные категории. Он может быть линейным и нелинейным в зависимости от функции ядра, а также может быть одноклассовым и многоклассовым [67, 68]. МОВ требует много памяти для обработки и времени для обучения.

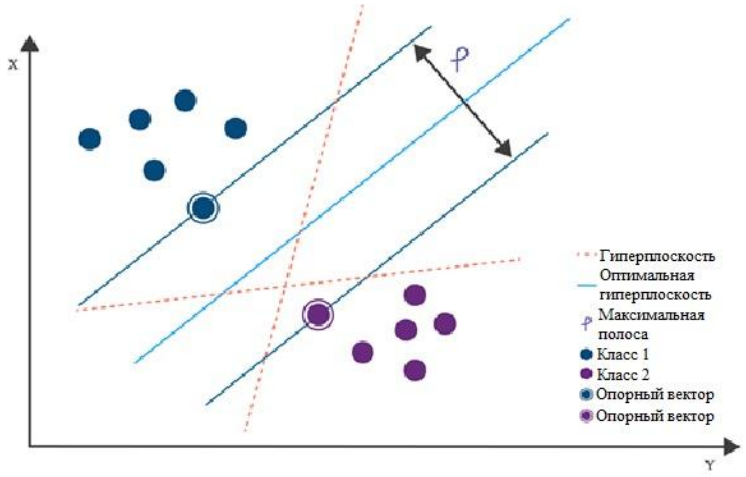

Рис. 9. Метод опорных векторов

К преимуществам метода можно отнести следующие:

- Хорошо работает с пространством признаков большого размера;
- Хорошо работает с данными небольшого объема;
- Алгоритм максимизирует разделяющую полосу, что позволяет уменьшить количество ошибок классификации.

Недостатки у метода следующие:

- Долгое время обучения (для больших наборов данных);
- Неустойчивость к «шуму»: выбросы в обучающих данных становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости.

Функции и параметры ядра также влияют на производительность классификатора.

Метод опорных векторов часто используется в задаче классификации. Рассмотрим задачу бинарной классификации, в которой объектам из $X = R^n$ соответствует один из двух классов $Y = \{-1; 1\}$.

Пусть задана обучающая выборка пар «объект-ответ»: $T^l = (\vec{x}_i, y_i)_{i=1}^l$. Необходимо построить алгоритм классификации $a(\vec{x}): X \to Y$.

В пространстве R^n уравнение $\langle \vec{w}, \vec{x} \rangle - b = 0$ при заданных $\vec{w}$ и b определяет гиперплоскость – множество векторов $\vec{x} = (x_1, \ldots, x_n)$, принадлежащих пространству меньшей размерности R^{n-1}. Например, для R^1 гиперплоскостью является точка, для R^2 – прямая, для R^3 – плоскость и т.д. Параметр $\vec{w}$ определяет вектор нормали к гиперплоскости, а через $\frac{b}{\|\vec{w}\|}$ выражается расстояние от гиперплоскости до начала координат.

Гиперплоскость делит R^n на два полупространства: $\langle \vec{w}, \vec{x} \rangle - b > 0$ и $\langle \vec{w}, \vec{x} \rangle - b < 0$.

Говорят, что гиперплоскость разделяет два класса C_1 и C_2, если выполнено условие:

$$\begin{cases} \langle \vec{w}, \vec{x} \rangle - b > 0, \forall x \in C_1 \\ \langle \vec{w}, \vec{x} \rangle - b < 0, \forall x \in C_2 \end{cases} \quad (1)$$

либо

$$\begin{cases} \langle \vec{w}, \vec{x} \rangle - b < 0, \forall x \in C_1 \\ \langle \vec{w}, \vec{x} \rangle - b > 0, \forall x \in C_2 \end{cases} \quad (2)$$

При выполнении одного такого условия объекты этих классов лежат по разные стороны от гиперплоскости.

Авторы статьи «Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs» [69] продемонстрировали точность работы данного метода для обнаружения вторжений на уровне 60-80% для разного вида атак.

1.3.9. Генетический алгоритм и генетическое программирование

Два наиболее популярных метода вычислений, основанных на принципе выживания сильнейших – это генетический алгоритм (ГА) и генетическое программирование (ГП). Эти алгоритмы работают с популяцией хромосом, которые развиваются на основе определенных операторов. Используются три основных оператора: отбор, скрещивание и мутация. Алгоритм запускается со случайно сгенерированной «популяцией», значение пригодности вычисляется для каждого объекта. Данный алгоритм определяет способность каждого объекта решать текущую проблему, и наборы объектов с более высокой способностью имеют более высокие шансы быть выбранными в пуле спаривания. Два выбранных объекта выполнят следующий шаг, называемый кроссовером, и, наконец, каждый из них подвергнется мутации. Среди двух мутировавших объектов самая подходящая хромосома будет передана следующему поколению.

Авторы статьи «Evolutionary design of intrusion detection programs» [70] использовали простую модель ГП для разработки классификатора атак. В этом анализе использовались три популярные модели ГП: линейное генетическое программирование, программирование экспрессии генов и программирование множественной экспрессии. В модели использовались различные математические операторы в качестве наборов функций. Набор данных DARPA 1998 использовался в качестве основного набора данных для проверки сгенерированной модели. В наборе данных было 4 основных типа атак (U2R, R2L, DoS и probe) с 24 различными сценариями атак. Доля ложных срабатываний вышеуказанной модели составляла от 0% до 5% в зависимости от типа исследуемой атаки.

1.3.10. Скрытые Марковские модели (CMM)

Модель представляет из себя Марковскую цепь, для которой известны начальная вероятность и матрица вероятностей переходов. Скрытой она

называется потому, что мы не имеем информации о ее текущем состоянии. Мы получаем информацию на основе некоторого наблюдения, в данном алгоритме мы будем использовать натуральное число от 1 до N, как индекс наблюдаемого события [71]. Для каждого состояния скрытой Марковской модели задан вектор вероятности эмиссии, который характеризует вероятность наблюдения каждого события, когда модель находится в этом состоянии. Совокупность таких векторов образует матрицу эмиссии.

Марковская модель λ задается как $\lambda = \{S, \Omega, \Pi, A, B\}$, где $S = \{s_1 \dots s_n\}$ — состояния, $\Omega = \{\omega_1 \dots \omega_m\}$ — возможные события, $\Pi = \{\pi_1 \dots \pi_n\}$ — начальные вероятности, $A = \{a_{ij}\}$ — матрица переходов, а $B = \{b_{i\omega_k}\}$— вероятность наблюдения события ω_k после перехода в состояние s_i [72].

Другими словами, СММ – это статистическая Марковская модель с набором состояний, которые связаны между собой с помощью переходных вероятностей, определяющих топологию модели. Предполагается, что система представляет собой Марковский процесс с ненаблюдаемыми параметрами. Эта модель обеспечивает прямо-обратную корреляцию, которую можно использовать для определения скрытых параметров по наблюдаемым параметрам. Поскольку распределение вероятностей в каждом состоянии различно, система может изменять состояния с течением времени и способна представлять нестационарные последовательности.

Авторы статьи «Investigating hidden Markov models capabilities in anomaly detection» [73] использовали СММ для разработки системы обнаружения вторжений. Для обучения алгоритма использовалось 5 из 41 признака аномального поведения сетевого трафика, присутствующих в наборе данных KDD 1999. Взаимосвязь между состояниями развита таким образом, что любое состояние может переходить в любое другое состояние. Доля корректных предсказаний модели составила 79%. Результат работы данного алгоритма показывает, что система способна классифицировать сетевой трафик

пропорционально количеству признаков, используемых для обучения CMM, поэтому если в анализе используется более 5 функций, точность модели может повыситься.

1.3.11. Индуктивное обучение

Вывод определенной информации из набора данных известен как дедукция. С другой стороны, переход от конкретного наблюдения к разработке теорий и моделей известен как индуктивное обучение. Это два основных метода, используемых для вывода информации из данных. Индуктивный анализ выявляет некоторые общие закономерности, которые используются для выработки некоторых гипотетических выводов.

Авторы статьи «Using artificial anomalies to detect unknown and known network intrusions» [74] разработали искусственный генератор аномалий для генерации случайных событий и аномального трафика. Для создания этих случайных аномалий использовались два основных подхода: генерация аномалий на основе распределения и отфильтрованные искусственные аномалии. Сгенерированные данные были случайным образом объединены с набором данных DARPA 1998 года. Эти данные были использованы авторами для изучения эффективности разработанной модели индуктивного обучения. Модель показала успешный уровень обнаружения 90 %, и был достигнут низкий уровень ложных срабатываний - 2%. В исследовании приведена методология разработки набора данных, который можно использовать для обнаружения аномалий, и показано применение модели индуктивного обучения к обработанному набору данных [75].

В рамках исследования проведен сравнительный анализ рассмотренных методов машинного обучения в задаче обнаружения компьютерных атак на примере датасета NSL-KDD. Результаты приведены в Таблице 1.

Модели машинного обучения, основанные на анализе сетевого трафика, демонстрируют максимальную точность на уровне 0,9279. Приведенные в

таблице значения позволяют ввести критерий эффективности для методики обнаружения компьютерных атак с использованием алгоритмов машинного обучения. Предлагаемая методика должна обеспечивать точность $\theta \geq 0,9$.

Таблица 1. Сравнительный анализ методов машинного обучения в задаче обнаружения компьютерных атак

Метод машинного обучения	**Преимущества**	**Недостатки**	**Примеры применения**	**Результаты работы алгоритмов на наборе данных NSL-KDD (F1-Score)**	**Время обучения (сек)**
Случайный лес	Высокая точность, устойчивость к переобучению	Требует больше вычислительных ресурсов, чем одиночные деревья	Классификация вредоносного ПО, обнаружение аномалий	0.9279	12.5551
k-ближайших соседей	Простота реализации, не требует обучения	Чувствитель-ность к данным, зависимость от выбора метрики расстояния и параметра k	Классификация сетевого трафика, обнаружение аномалий	0.9231	0.0068
Генетический алгоритм	Глобальный поиск решений, устойчивость к локальным оптимумам	Высокая вычислительная сложность, медленная сходимость	Оптимизация параметров моделей, обнаружение аномалий	0.9135	267.8857
Нейронные сети	Способность моделировать сложные нелинейные зависимости	Требует большого объема данных для обучения, сложность интерпретации	Обнаружение аномалий, классификация вредоносного ПО	0.9000	275.8265
Деревья решений	Простота интерпретации, устойчивость к шуму в данных	Склонность к переобучению, снижение точности на сложных данных с большим количеством признаков	Обнаружение аномалий в сетевом трафике, признаков компьютерных атак	0.8972	0.8664
Индуктивное обучение	Простота интерпретации, применение для задач с четкими правилами	Низкая точность на сложных данных, требует ручного создания правил	Обнаружение аномалий в сетевом трафике, классификация вредоносного ПО	0.8841	0.2138

Метод опорных векторов	Использование для задач классификации с малым объемом данных	Требует большого объема вычислительных ресурсов для больших данных	Обнаружение вторжений, классификация сетевого трафика	0.8194	7.1415
Скрытые Марковские модели	Применяются для анализа временных последовательностей	Требуют большого объема данных для обучения, сложность настройки	Анализ журналов событий, обнаружение аномалий в временных данных	0.7900	8.8711
Автоэнкодеры	Применяются для обнаружения аномалий	Требуют большого объема данных и вычислительных ресурсов	Обнаружение аномалий в сетевом трафике, анализ лог-файлов	0.6975	1055.4080
Кластеризация	Возможность поиска скрытых структур в данных	Требует выбора числа кластеров, чувствительность к начальным условиям	Сегментация сетевого трафика, обнаружение аномалий	0.6973	0.9686
Байесовская сеть	Учет вероятностных зависимостей, интерпретируемость	Требует точного задания априорных вероятностей, сложность для больших данных	Обнаружение аномалий в сетевом трафике, анализ рисков	0.6952	0.0265

1.4 Методика выявления компьютерных атак с использованием алгоритмов машинного обучения

В Российской Федерации создана и функционирует государственная система обнаружения, предупреждения и ликвидации последствий компьютерных атак на информационные ресурсы Российской Федерации (ГосСОПКА).

Деятельность данной системы строится на основе национальных стандартов России:

- ГОСТ Р 59547-2021 Защита информации. Мониторинг информационной безопасности. Общие положения[5];
- ГОСТ Р 59709-2022 Защита информации. Управление компьютерными инцидентами. Термины и определения[6];
- ГОСТ Р 59710-2022 Защита информации. Управление компьютерными инцидентами. Общие положения[7];
- ГОСТ Р 59711-2022 Защита информации. Управление компьютерными инцидентами. Организация деятельности по управлению компьютерными инцидентами[8];
- ГОСТ Р 59712-2022 Защита информации. Управление компьютерными инцидентами. Руководство по реагированию на компьютерные инциденты[9].

Указанные стандарты описывают структурированный подход к организации и ведению деятельности по управлению компьютерными инцидентами, который включает в себя четыре стадии:

1. Организация деятельности по управлению компьютерными инцидентами;

2. Обнаружение и регистрация компьютерных инцидентов;

---

[5] Национальный стандарт РФ ГОСТ Р 59547-2021 Защита информации. Мониторинг информационной безопасности. Общие положения (утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 27 июля 2021 г. N 656-ст

[6] Национальный стандарт РФ ГОСТ Р 59709-2022 Защита информации. Управление компьютерными инцидентами. Термины и определения (утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 29 ноября 2022 г. N 1375-ст)

[7] Национальный стандарт РФ ГОСТ Р 59710-2022 Защита информации. Управление компьютерными инцидентами. Общие положения (утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 29 ноября 2022 г. N 1376-ст)

[8] Национальный стандарт РФ ГОСТ Р 59711-2022 Защита информации. Управление компьютерными инцидентами. Организация деятельности по управлению компьютерными инцидентами (утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 29 ноября 2022 г. N 1377-ст)

[9] Национальный стандарт РФ ГОСТ Р 59712-2022 Защита информации. Управление компьютерными инцидентами. Руководство по реагированию на компьютерные инциденты (утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 29 ноября 2022 г. N 1378-ст)

3. Реагирование на компьютерные инциденты (включая фиксацию материалов, связанных с возникновением компьютерных инцидентов и установление причин и условий возникновения компьютерных инцидентов);

4. Анализ результатов деятельности по управлению компьютерными инцидентами.

Компьютерным инцидентом стандартами определен факт нарушения и (или) прекращения функционирования информационного ресурса, сети электросвязи, используемой для организации взаимодействия информационных ресурсов, и (или) нарушения безопасности обрабатываемой в информационном ресурсе информации, в том числе произошедший в результате компьютерной атаки.

Для описания методики выявления компьютерных инцидентов (компьютерных атак) используем нотацию IDEF0.

Нотация IDEF0 (англ. Integration Definition for Function Modeling) представляет собой методологию функционального моделирования и графического описания процессов, предназначенную для формализации и описания бизнес-процессов. Данная нотация была разработана на основе методологии структурного анализа и проектирования SADT (англ. Structured Analysis and Design Technique).

IDEF0 является одной из наиболее распространенных нотаций для моделирования организационных и функциональных систем, позволяющей описывать процессы на различных уровнях детализации. Особенностью данной нотации является ее способность эффективно отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах. При этом в основе методологии находятся функциональные отношения между работами, а не их временная последовательность.

Нотация IDEF0 использует следующие основные типы элементов для построения диаграмм:

1. Функциональные блоки (англ. Activity Boxes). Функциональные блоки представляют собой прямоугольники, которые обозначают функции, процессы или работы, выполняемые в рамках моделируемой системы. Каждый функциональный блок имеет уникальное имя, описывающее действие, которое выполняется в рамках данной функции. Внутри прямоугольника указывается название функции и ее идентификатор (обычно в правом нижнем углу). Функциональные блоки на диаграмме IDEF0 располагаются по степени важности (доминирования) - наиболее важные размещаются в верхнем левом углу диаграммы, а наименее важные - в правом нижнем. Такое расположение называется диагональным доминированием.

2. Интерфейсные стрелки (англ. Arrows). Интерфейсные стрелки представляют собой направленные линии со стрелками на концах, которые отображают элементы системы, обрабатываемые функциональными блоками или оказывающие иное влияние на функции. Стрелки описывают взаимодействие функций между собой и с внешней средой. В зависимости от того, к какой стороне прямоугольника подходит стрелка, она имеет различное назначение:

- Входные стрелки (англ. Input) – подходят к левой стороне функционального блока и изображают данные или материальные объекты, которые преобразуются или расходуются функцией для получения результата;
- Выходные стрелки (англ. Output) – выходят из правой стороны функционального блока и представляют данные или материальные объекты, производимые или создаваемые функцией;
- Управляющие стрелки (англ. Control) – входят в верхнюю часть функционального блока и представляют условия, правила, стратегии, процедуры или стандарты, которыми руководствуется функция;
- Механизмы (англ. Mechanism) – входят в нижнюю часть функционального блока и показывают ресурсы, которые выполняют функцию, но не изменяются ею (например, персонал, оборудование, программное обеспечение).

В IDEF0 используется специальный вид контекстной диаграммы (A-0), состоящей из одного блока, описывающего функцию верхнего уровня, ее входы, выходы, управления и механизмы. На ней формулируется цель модели и точка зрения, с которой строится модель. Точка зрения указывает на должностное лицо или подразделение, с позиции которого разрабатывается модель. На контекстной диаграмме содержится контекстная функция A0, которая указывает на основное действие, выполняемое системой.

Нотация IDEF0 обладает рядом преимуществ, которые делают ее эффективным инструментом для моделирования сложных систем:

- Полнота описания – методология обеспечивает полное описание моделируемой системы и ее окружения;
- Комплексность декомпозиции – позволяет постепенно детализировать процессы до необходимого уровня;
- Выразительность – наглядно отображает все основные элементы процессов и взаимосвязи между ними;
- Строгость и точность – благодаря строгим правилам построения диаграмм обеспечивается точность и непротиворечивость моделей;
- Поддержка итеративного моделирования – возможность постепенного уточнения и совершенствования модели.

В контексте информационной безопасности и, в частности, управления компьютерными инцидентами, нотация IDEF0 является эффективным инструментом для моделирования процессов обнаружения, регистрации и реагирования на инциденты. Она позволяет четко определить: входные данные для процессов (события информационной безопасности, журналы, дампы трафика и т.д.), выходные результаты (зарегистрированные инциденты, отчеты, рекомендации), управляющие воздействия (политики безопасности, стандарты, методики), механизмы реализации (персонал, программное обеспечение, технические средства).

Нотация IDEF0 представляет собой мощный инструмент для формализации и визуализации процессов управления компьютерными

инцидентами, обеспечивая структурированный подход к их обнаружению, регистрации и реагированию в соответствии с требованиями ГОСТов серии 59709-59712.

Методику выявления компьютерных инцидентов в нотации IDEF0 можно представить следующим образом [76].

Функция A0: Выявлять компьютерные инциденты

Данная функция является родительской и указывает на основные элементы, которые включает в себя процесс выявления компьютерных инцидентов.

Входные данные (I1): Источники данных мониторинга – программные или программно-технические средства, с которых осуществляется сбор данных мониторинга. Основными источниками выступают системы обнаружения и предотвращения компьютерных атак, а также системы управления событиями информационной безопасности. В качестве входных данных (I2) также могут использоваться события операционных систем, лог-файлы приложений и сетевого оборудования, опросы персонала, индикаторы компрометации и другие сведения, предназначенные для подтверждения факта возникновения компьютерного инцидента.

Управление (C1): Функция регулируется требованиями законодательства и стандартами:

Нормативными правовыми актами и стандартами в области информационной безопасности, включая ГОСТы серии 59709-59712, которые определяют требования к процессам управления компьютерными инцидентами;

Политиками безопасности – внутренними документами организации, определяющие принципы, правила и требования по обеспечению информационной безопасности.

Возможные ограничения – факторы, ограничивающие возможности мониторинга (технические, организационные, правовые).

В качестве элементов управления (C2) выступают правила регистрации информации о компьютерном инциденте. Должны быть описаны требования к оформлению зарегистрированных инцидентов, а сам процесс подтверждения проходит по утвержденной методике.

Механизмы: Для реализации функции используются два основных элемента:

Технические средства (M1) – технические средства и программное обеспечение для сбора и анализа данных о событиях безопасности;

Персонал (M2) – сотрудники центра мониторинга, осуществляющие анализ собранных данных и выявление потенциальных угроз, методисты и технические эксперты.

Выходные данные (O1): Зарегистрированные компьютерные инциденты, по которым в дальнейшем осуществляется реагирование (следующая стадия управления компьютерными инцидентами).

Общая схема выявления компьютерных инцидентов в нотации IDEF0 представлена на Рисунке 10.

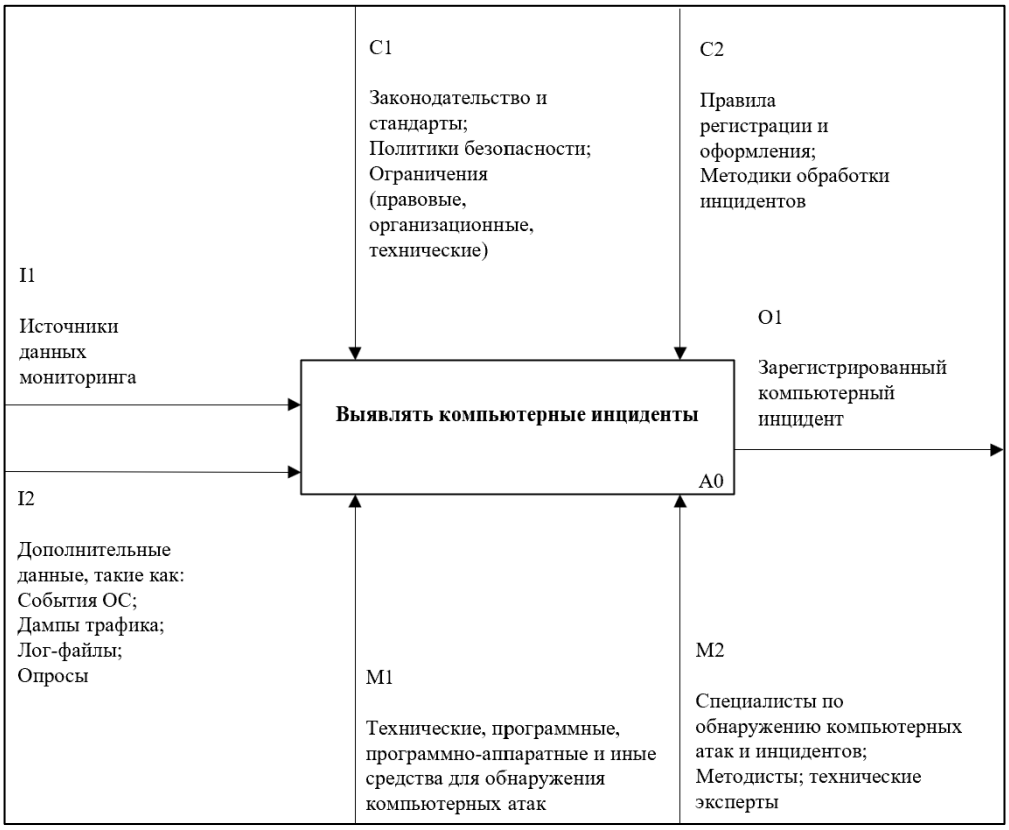

Рис. 10. Процесс выявления компьютерных инцидентов в нотации IDEF0

В ходе декомпозиции родительской функции можно выделить следующие функции, входящие в процесс выявления, подтверждения и регистрации компьютерных инцидентов.

Функция A1: Осуществлять мониторинг информационной безопасности

Функция является первым и основополагающим этапом в процессе выявления компьютерных инцидентов. Направлена на непрерывное наблюдение за состоянием информационной безопасности в организации и сбор данных о событиях безопасности из различных источников. В соответствии с ГОСТ Р 59547-2021, мониторинг информационной безопасности должен обеспечивать своевременное обнаружение событий безопасности, которые могут свидетельствовать о возникновении компьютерных инцидентов. Данная функция является критически важной для эффективного функционирования всей системы управления компьютерными инцидентами, поскольку от качества и полноты собираемых данных зависит возможность своевременного выявления и реагирования на инциденты.

Входные данные (I1): Источники данных мониторинга - включают в себя системы обнаружения и предотвращения вторжений, антивирусные средства, системы контроля доступа и другие компоненты информационной инфраструктуры. Эти источники предоставляют первичные данные о событиях, происходящих в информационной системе.

Управление (C1): Функция регулируется требованиями законодательства и стандартами, которые были описаны ранее у родительской функции.

Механизмы (M1): Для реализации функции используются:

Технические средства мониторинга – программно-аппаратные средства и программное обеспечение для сбора и анализа данных о событиях безопасности;

Специалисты по управлению инцидентами – сотрудники центра мониторинга, осуществляющие первичный анализ собранных данных и выявление потенциальных угроз.

Выходные данные (O1): События информационной безопасности, которые могут указывать на нарушения безопасности или потенциальные угрозы. Эти события передаются для дальнейшего анализа и обработки в рамках функции «Зарегистрировать признаки возможного возникновения компьютерных инцидентов».

Функция A2: Зарегистрировать признаки возможного возникновения компьютерных инцидентов

Функция представляет собой второй этап в процессе выявления компьютерных инцидентов и направлена на анализ событий информационной безопасности с целью выявления признаков, которые могут свидетельствовать о возникновении компьютерных инцидентов. В соответствии с ГОСТ Р 59712-2022 «Защита информации. Управление компьютерными инцидентами. Руководство по реагированию на компьютерные инциденты», регистрация признаков возможного возникновения компьютерных инцидентов может осуществляться как автоматизированным способом (с использованием средств управления событиями информационной безопасности) на основе правил регистрации, так и неавтоматизированным способом (специалистами при самостоятельном анализе событий безопасности или при получении соответствующей информации от работников организации). Правила регистрации признаков возможного возникновения компьютерных инцидентов должны позволять реализовать один или совокупность методов анализа, включая сигнатурные методы (основанные на сопоставлении конкретных признаков и условий взаимосвязей событий) и бессигнатурные методы (основанные на выявлении статистической и иной зависимости между событиями и формировании профилей функционирования информационных ресурсов).

Входные данные (O1 от A1): События информационной безопасности – зафиксированные в ходе мониторинга события, которые требуют анализа для выявления потенциальных инцидентов. Эти события поступают от функции «Осуществлять мониторинг информационной безопасности» и содержат информацию о различных действиях и состояниях, зафиксированных в информационной системе.

Управление (C2): Правила регистрации компьютерных инцидентов определяют порядок и критерии регистрации событий информационной безопасности.

Механизмы (M2): Для реализации функции используются:

Средства управления событиями информационной безопасности - специализированные программные средства (SIEM-системы), которые обеспечивают сбор, нормализацию, корреляцию и анализ событий безопасности;

Специалисты по управлению инцидентами - персонал, обладающий необходимыми компетенциями для анализа событий безопасности и выявления признаков инцидентов;

Методисты – сотрудники центра мониторинга, которые определяют критерии для событий информационной безопасности.

Выходные данные (O2): Карточки признаков компьютерных инцидентов – структурированные записи, содержащие информацию о выявленных признаках, которые могут свидетельствовать о возникновении компьютерных инцидентов. Эти карточки включают описание признаков, их классификацию, источник информации, время обнаружения и другие атрибуты, необходимые для дальнейшего анализа.

Функция A3: Подтвердить компьютерные инциденты

Функция является третьим этапом в процессе выявления компьютерных инцидентов и направлена на проверку и подтверждение того, что зарегистрированные признаки действительно свидетельствуют о возникновении компьютерного инцидента. В соответствии с ГОСТ Р 59712-

2022, подтверждение компьютерных инцидентов включает в себя проверку достоверности информации о признаках возможного возникновения компьютерных инцидентов и установление факта возникновения компьютерного инцидента. Этот процесс может включать дополнительный сбор и анализ информации, проведение опросов пользователей и администраторов, анализ системных событий, лог-файлов и дампов трафика, а также использование специализированных инструментов для исследования потенциальных инцидентов. Подтверждение компьютерных инцидентов является критически важным этапом, поскольку позволяет отфильтровать ложные срабатывания и сосредоточить ресурсы на реагировании на реальные инциденты. Результатом этого этапа является формирование карточек компьютерных инцидентов, которые содержат всю необходимую информацию для последующей регистрации и реагирования.

Входные данные (O2 от A2): Карточки признаков компьютерных инцидентов структурированные записи о выявленных признаках, требующих проверки.

I2: Дополнительные источники данных, включающие:

События операционной системы – детальные записи о событиях в операционных системах;

Дампы трафика – записи сетевого трафика, которые могут содержать следы вредоносной активности;

Лог-файлы – журналы различных приложений и сервисов;

Опросы – информация, полученная от пользователей и администраторов систем.

Управление (C3): Функция основывается на методиках обработки инцидентов – документах, регламентирующих процедуру обработки.

Механизмы (M3): Для реализации функции используются:

Специализированное программное обеспечение, скрипты – программные средства для анализа системных событий, дампов трафика и других данных;

Аналитики – персонал, обладающий необходимыми компетенциями для проведения расследований и подтверждения инцидентов;

Технические эксперты – сотрудники центра мониторинга, обладающие глубокими компетенциями в конкретной сфере информационной безопасности. Могут привлекаться для принятия решения по отдельным сложным инцидентам.

Выходные данные (О3): Карточки компьютерных инцидентов – структурированные записи, содержащие подтвержденную информацию о компьютерных инцидентах, включая их описание, классификацию, источник, время обнаружения, затронутые ресурсы и другие атрибуты, необходимые для дальнейшего реагирования.

Функция А4: Зарегистрировать компьютерный инцидент

Функция представляет собой заключительный четвертый этап в процессе управления компьютерными инцидентами и направлена на регистрацию подтвержденных компьютерных инцидентов в системе управления инцидентами. В соответствии с ГОСТ Р 59712-2022, регистрация компьютерного инцидента включает в себя присвоение уникального идентификатора инциденту, определение его категории и приоритета, а также фиксацию всей доступной информации об инциденте в системе управления инцидентами. Категория инцидента определяется на основе его типа, масштаба и потенциального воздействия на информационные ресурсы организации. Приоритет инцидента устанавливается с учетом его критичности и срочности реагирования. Регистрация компьютерного инцидента обеспечивает формализацию процесса управления инцидентами и создает основу для последующего реагирования, расследования и анализа. Зарегистрированные инциденты становятся частью базы знаний организации и могут использоваться для улучшения процессов обеспечения информационной безопасности и предотвращения подобных инцидентов в будущем.

Входные данные (O3 от A3): Карточки компьютерных инцидентов – структурированные записи, содержащие подтвержденную информацию о компьютерных инцидентах, полученные от функции «Подтвердить компьютерные инциденты».

Управление (C4): Функция регулируется правилами оформления, которые устанавливают требования к документированию процессов мониторинга.

Механизмы (M4): Для реализации функции используются системы управления инцидентами – специализированное программное обеспечение, обеспечивающее учет, обработку и аналитику по всем зарегистрированным компьютерным инцидентам;

Регистрацией компьютерных инцидентов в системе занимаются аналитики;

Методисты обеспечивают разработку правил регистрации и учета инцидентов.

Выходные данные (O4): Зарегистрированный компьютерный инцидент – зарегистрированная запись о компьютерном инциденте в системе управления инцидентами, которая содержит всю необходимую информацию для последующего реагирования и расследования.

Описанная методика выявления компьютерных инцидентов, построенная в соответствии с серией ГОСТов серии 59709-59712 в нотации IDEF0 приведена на Рисунке 11.

Модификация методики

Описанная выше методика выявления компьютерных инцидентов предусматривает использование журналов операционных систем в качестве одного из источников данных для подтверждения компьютерных инцидентов (функция A3). Однако, в условиях постоянно растущего объема данных и усложнения компьютерных атак, традиционные методы анализа журналов становятся недостаточно эффективными.

Применение для обработки журналов операционной системы методов машинного обучения позволяет:

Автоматизировать процесс анализа больших объемов данных журналов операционной системы;

Выявлять неочевидные закономерности и аномалии, которые могут указывать на компьютерные атаки;

Повысить точность обнаружения компьютерных инцидентов и снизить количество ложных срабатываний;

Обнаруживать новые, ранее неизвестные типы атак на основе поведенческого анализа.

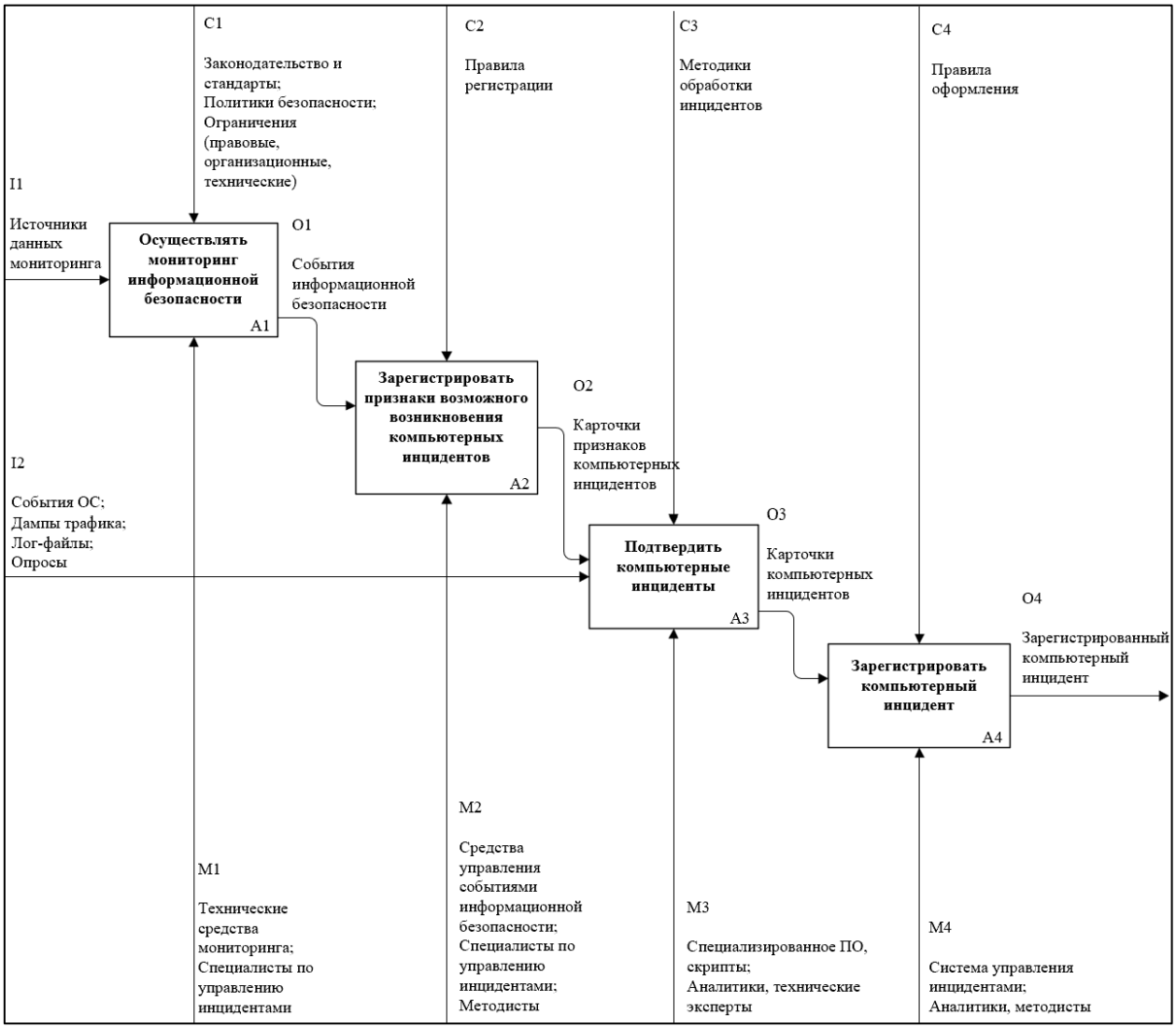

Рис. 11. Методика выявление компьютерных инцидентов в нотации IDEF0

Согласно ГОСТ Р 59712-2022, для регистрации признаков возможного возникновения компьютерных инцидентов могут применяться как сигнатурные, так и бессигнатурные методы анализа. Бессигнатурные методы, основанные на выявлении статистической и иной зависимости между событиями безопасности и формировании профилей функционирования информационных ресурсов, могут быть эффективно реализованы с использованием методов машинного обучения.

Модификация методики предполагает добавление нового функционального блока и изменение существующих связей в диаграмме IDEF0. Новый функциональный блок:

Функция А2: Провести анализ событий операционной системы методами машинного обучения

Функция направлена на использование методов машинного обучения для анализа событий операционной системы с целью выявления признаков компьютерных атак и включает в себя следующие этапы:

- Подготовку исходных данных для последующего анализа методами машинного обучения. Включает в себя сбор, фильтрацию, нормализацию и структурирование данных;
- Создание и обучение моделей машинного обучения для анализа журналов операционной системы с целью выявления признаков компьютерных атак. Включает в себя выбор алгоритмов, обучение моделей на исторических данных и их валидацию;
- Применение обученных моделей машинного обучения для анализа текущих источников с целью выявления признаков компьютерных атак. Включает в себя обработку данных с помощью моделей, интерпретацию результатов и формирование выводов.

Полученная в результате работы функции оценка модели машинного обучения служит источником дополнительной информации при регистрации

признаков компьютерных инцидентов для дальнейшего принятия решения о подтверждении факта возникновения инцидента.

Входные данные (I2): записи системных событий операционных систем, а также исторические данные о компьютерных инцидентах – информация о ранее зафиксированных инцидентах, используемая для обучения моделей.

Управление (C2): правила предварительной обработки данных – определяют порядок и критерии фильтрации, нормализации и структурирования данных;

Рекомендации по выбору методов машинного обучения – определяющие подходы к выбору алгоритмов и обучению моделей;

Критерии для определения инцидентов – параметры, определяющие значения для идентификации компьютерных атак.

Механизмы: (M2) Специализированное программные средства для обработки событий операционной системы – программные средства для сбора, фильтрации и нормализации данных;

Платформы и библиотеки машинного обучения – программные средства для создания и обучения моделей;

Системы анализа данных – программные средства для применения моделей машинного обучения и интерпретации результатов;

Аналитики данных – персонал, обладающий компетенциями в области анализа данных и машинного обучения.

Выходные данные: (O2) – оценка модели машинного обучения вероятности того, что в источниках данных содержатся признаки компьютерного инцидента.

Модификация исходных функциональных блоков

A3 (ранее A2): Зарегистрировать признаки возможного возникновения компьютерных инцидентов (модифицированный)

Дополнительные входные данные: Оценка модели машинного обучения (от A2);

Также изменена нумерация функциональных блоков, входов, выходов, механизмов и контролей в исходной методике в соответствии с нумерацией добавленных элементов.

Модифицированная методика выявления компьютерных инцидентов, включающая элементы машинного обучения, построенная в нотации IDEF0 приведена на Рисунке 12.

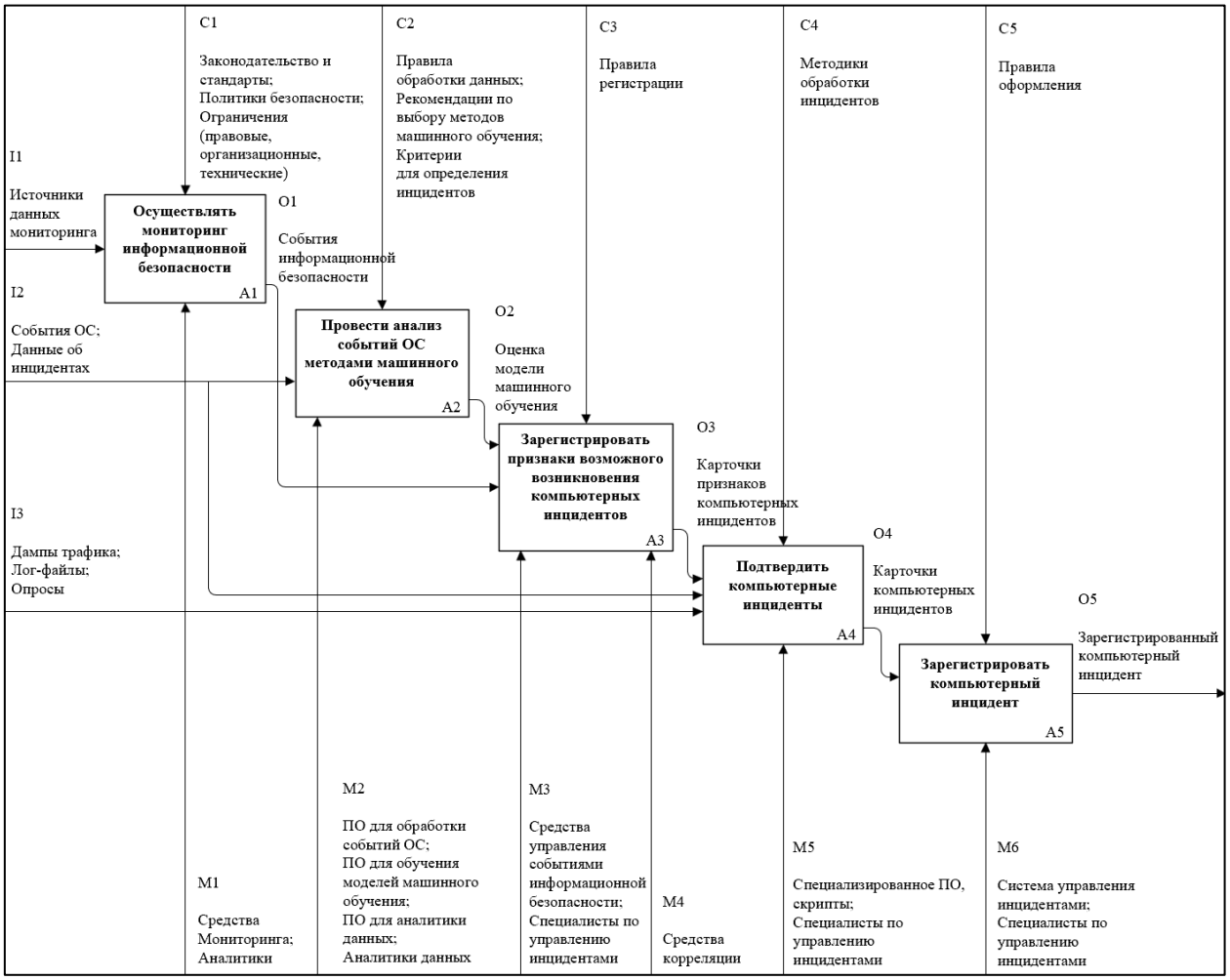

Рис. 12. Предлагаемая методика выявления компьютерных инцидентов

Преимущества предлагаемой методики

1. Повышение точности обнаружения – применение методов машинного обучения позволяет выявлять неочевидные закономерности и аномалии, которые могут быть пропущены при традиционном анализе;

2. Обнаружение новых типов атак – бессигнатурные методы, реализованные с использованием машинного обучения, способны выявлять ранее неизвестные типы атак на основе поведенческого анализа;

3. Автоматизация анализа больших объемов данных – методы машинного обучения эффективно справляются с обработкой больших объемов данных журналов ОС;

4. Адаптивность к изменениям – модели машинного обучения могут быть переобучены с учетом новых данных, что обеспечивает адаптацию методики к изменяющимся условиям и новым угрозам.

Предлагаемая методика соответствует требованиям стандартов, регламентирующих систему функционирования ГосСОПКА, дополняя и расширяя ее. Методика согласуется с положениями ГОСТ Р 59712-2022 в части применения бессигнатурных методов анализа, основанных на выявлении статистической и иной зависимости между событиями безопасности и формировании профилей функционирования информационных ресурсов. Алгоритм машинного обучения Random Forest, внедренный в методику, реализует данный подход, обеспечивая более эффективное выявление признаков компьютерных атак на основе анализа системных событий операционной системы.

1.5 Выводы по главе

В рамках первой главы исследования проведен обзор основных видов компьютерных атак и методов их обнаружения. Подробно рассмотрена практика применения методов машинного обучения в области кибербезопасности. Проведен сравнительный анализ указанных методов, а также результаты их работы на датасете NSL-KDD. Максимальная точность классификаторов, обученных на сетевом трафике, составила 0,9279. Введен

критерий эффективности: предлагаемая методика должна обеспечивать точность обнаружения не менее 0,9.

В соответствии с серией ГОСТов 59712-2022 составлена методика выявления компьютерных атак в нотации IDEF0, включающая элементы, связанные с анализом системных событий операционной системы с использованием алгоритмов машинного обучения. Предложенная методика обладает рядом преимуществ, повышая точность выявления атак.

2 Анализ системных событий операционной системы

2.1 Исследование видов и содержания событий операционной системы Microsoft Windows

На сегодняшний момент существует множество различных операционных систем, относящихся к различным семействам. По данным статистического агентства Statcounter GlobalStats в 2025 году более 70% рынка всех операционных систем для персональных компьютеров занимают операционные системы семейства Windows[10].

Еще в 1985 году вышла первая операционная система семейства Windows, которая давала возможность пользователю выходить из текстового режима и просматривать несколько приложений одновременно. Данный аспект стал ключом к популяризации данной операционной системы. С выходом Windows 95 она стала распространяться огромными темпами, с тех пор это занимая первое место по распространенности в мире. Благодаря более удобному и понятному интерфейсу, не требующего больших усилий в освоении, операционные системы семейства Windows еще долго будут держать лидирующие позиции. Однако, из-за глобальной популярности данного семейства операционных систем, она стала приоритетным выбором и для проведения компьютерных атак со стороны злоумышленников. Чем больше людей используют операционную систему, тем больше устройств потенциально могут стать объектом несанкционированного воздействия.

[10] Подробную информацию можно найти на сайте: https://gs.statcounter.com/os-market-share/desktop/worldwide

Для профилактики кибератак необходимо регулярно устанавливать обновления операционной системы и пользоваться средствами антивирусной защиты информации.

Если же атака на устройство успешно произведена, специалистам по защите информации приходится работать с системами записи происходящих событий в операционной системе. Для этого они используют, в том числе, встроенное в операционную систему Windows средство просмотра событий, которое несмотря на всю свою сложность помогает разобраться в причинах заражения [77].

Для ликвидации последствий и установления причин проведенной на компьютер атаки специалистам по информационной безопасности или криминалистам необходимо отслеживать не только текущее состояние операционной системы, но и состояния, которые предшествовали атаке. Необходимо видеть всю динамику происходящих в операционной системе процессов. Системные события Windows хранят в себе информацию обо всех произошедших изменениях в операционной системе.

Тот вид, в котором можно наблюдать системные события на данный момент, был не всегда. До выхода операционной системы Windows Vista отсутствовала унификация событий Windows и журналов событий данной операционной системы [78].

Большинство программного обеспечения, созданного для операционных систем семейства Windows, как и сама операционная система, регистрируют ошибки, сбои, события в собственные журналы событий, и все они имеют отличный друг от друга формат. Ведение же централизованного журнала событий, в который записываются события работы той или иной программы, значительно облегчает работу с журналами, так как не требуется работать одновременно с различными типами событий. Абсолютно все программные продукты для операционной системы Windows, записывают как минимум

информацию о своем запуске в журналы событий Windows [79], а некоторое программное обеспечение записывает также и выход из него.

Системные события – это инструмент, позволяющий фиксировать все изменения, происходящие в операционной системе, чтобы в случае необходимости специалист по защите информации или администратор мог их просмотреть, проанализировать и выявить причины нештатных ситуаций, возникающих в системе [80]. Системные события облегчают отслеживание текущего состояние системы, позволяют проанализировать производительность системы и историю изменений. Они могут рассказать об ошибках и сбоях, произошедших в системе, определить потенциально опасную деятельность пользователя. Анализ событий помогает не только определить нештатное поведение программного обеспечения, но и сделать выводы о времени и причинах такого поведения [81].

События операционной системы Windows могут содержать в себе множество различных типов изменений, от которых будет зависеть запись в тот или иной журнал событий.

В программном интерфейсе приложения просмотра событий отображается множество различных журналов, в том числе и журналы некоторых приложений и служб.

Журналы событий – это файлы определенного формата, в которые записываются все события, происходящие в системе. В эти файлы записываются события получения доступа, создания, изменения или удаления файлов, изменения системных параметров, таких как, например, дата и время, изменения конфигураций системы и другие [82]. В них записываются миллионы событий, которые попадают в различные журналы Windows.

Рассмотрим журналы системных событий Windows подробнее.

1. Журнал безопасности (Security). Любые события, имеющие значение для безопасности системы, записываются именно в этот журнал, например, удаление файлов, вход в систему, выход из системы и другие события.

2. Журнал системы (System). В журнале системы регистрируются события, зарегистрированные операционной системой, такие как ошибки или сбои при запуске жесткого диска.

3. Журнал приложений (Application). В данный журнал записываются события, определенные разработчиками того или иного программного обеспечения. Например, ошибки при запуске приложений.

4. Журнал событий установки (Setup Events). Содержит события, связанные с установкой, обновлением и настройкой Windows. Используется для диагностики проблем во время установки операционной системы, обновлений или миграции данных.

5. Журнал пересланных событий (Forwarded Events). Журнал для хранения событий, пересланных с других компьютеров в сети (через подписки на события). Используется в корпоративных средах для централизованного мониторинга событий с нескольких рабочих станций.

Каждый файл журнала представляет из себя бинарный файл формата EVTX схожий по формату с файлом базы данных. Сам журнал состоит из определенных категорий, разделенных по столбцам, как представлено на Рисунке 13.

Каждая запись события состоит из следующих категорий:

1. Критичность события (уровень).

1.1.Ошибка – некое событие, указывающее на проблему, например, такую как проблема ведения реестра операционной системы.

1.2.Предупреждение – необязательно значимое событие, но указывающее на возможную проблему в будущем. Если приложение смогло

восстановиться самостоятельно, то событие будет классифицироваться именно как предупреждение.

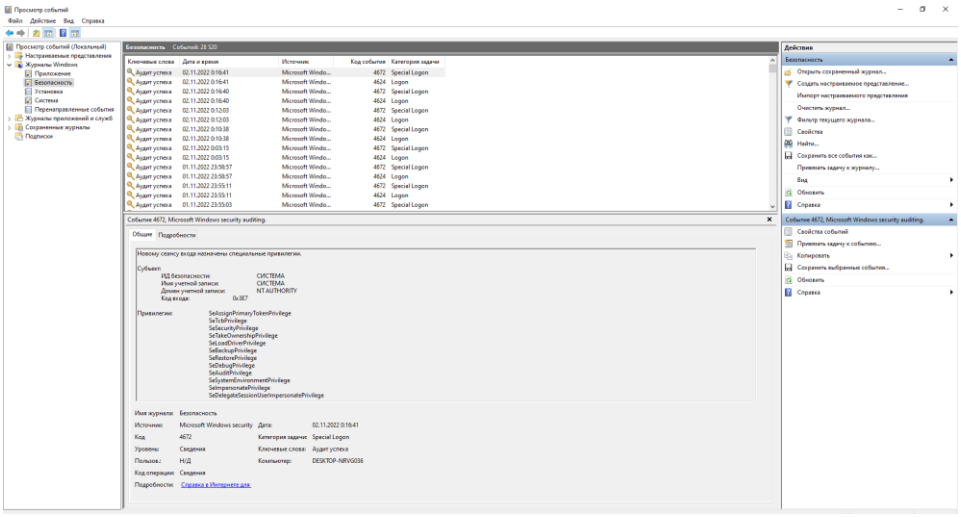

Рис. 13. Внешний вид приложения Просмотр событий

1.3.Сведения – события, которые описывают успешную работу драйвера, службы или приложения. К примеру, если какая-либо системная служба операционной системы запустилась без ошибок, то запись в журнале будет иметь вид «сведения».

1.4.Аудит отказа – данная запись делается при неудачной попытке получения доступа к функционалу безопасности. Например, попытки пользователя попасть в скрытую системную папку, не имея прав администратора, запишутся в аудит отказа.

1.5.Аудит успеха – противоположность аудиту отказа. Записываются удачные случаи получения доступа, например, вход в систему.

2. Дата и время регистрации того или иного события.

3. Источник события – это название программного обеспечения, которое зарегистрировало данное событие. Часто это имя приложения или имя подкомпонента приложения, если оно большое. В реестр можно добавить не более 16 384 источников событий. Журнал безопасности предназначен только

для системного использования. Драйверы устройств добавляют свои имена в системный журнал. Приложения и службы добавляют свои имена в журнал приложений или создают собственный журнал.

4. Категория задачи – помогает объединять события для дальнейшей их фильтрации. Каждый источник события может определять свои собственные категории, в которых они отображаются. Категории пронумерованы последовательно, начиная с номера 1. Категории могут храниться в отдельном файле сообщений или в файле, который содержит сообщения других типов.

5. Идентификатор события (код события) – однозначно идентифицирует событие. Разные источники могут идентифицировать события по-разному, зависит от замысла разработчиков программного обеспечения.

6. Пользователь – содержит имя пользователя, от имени которого были выполнены процессы в системе. Данное поле имеет высокую важность в системе безопасности с целью отслеживания событий каждого пользователя в отдельности.

7. Компьютер – указывается имя автоматизированного рабочего места, на котором регистрируется событие.

Для простоты чтения данных событий каждый журнал можно выгрузить в удобном для дальнейшей обработки формате:

1. EVTX – основной формат журналов событий операционной системы Windows.

2. XML – так называемый язык разметки, который используется для хранения и передачи данных.

3. CSV – формат текстового вида, предназначенный для вывода табличных данных.

4. TXT – формат для хранения текста, информация в который записывается в виде строк.

2.1.2. Формат файла журнала событий

Каждый журнал событий содержит заголовок, представленный структурой ELF_LOGFILE_HEADER, используемый в начале журнала событий для определения информации о журнале, имеющий фиксированный размер.

Служба регистрации событий должна добавить ELF_LOGFILE_HEADER в журнал событий.

Структура ELF_LOGFILE_HEADER:

```
EVENTLOGHEADER {
        ULONG HeaderSize;
        ULONG Signature;
        ULONG MajorVersion;
        ULONG MinorVersion;
        ULONG StartOffset;
        ULONG EndOffset;
        ULONG CurrentRecordNumber;
        ULONG OldestRecordNumber;
        ULONG MaxSize;
        ULONG Flags;
        ULONG Retention;
        ULONG EndHeaderSize;
},
```

где:

1. HeaderSize – размер структуры заголовка. Размер всегда 0x30.

2. Signature – подпись всегда 0x654c664c, что является ASCII для журнала событий.

3. MajorVersion – основной номер версии журнала событий, всегда установлен на 1.

4. MinorVersion – дополнительный номер версии журнала событий, всегда установлен на 1.

5. StartOffset – смещение к самой старой записи в журнале событий.

6. EndOffset – смещение к ELF_EOF_RECORD в журнале событий.

7. CurrentRecordNumber – номер следующей записи, которая будет добавлена в журнал событий.

8. OldestRecordNumber – номер самой старой записи в журнале событий. Для пустого файла самый старый номер записи установлен 0.

9. MaxSize – максимальный размер в байтах журнала событий. Максимальный размер определяется при создании журнала событий. Служба регистрации событий обычно не обновляет это значение, она опирается на конфигурацию реестра. Считыватель журнала событий может использовать обычные файловые API для определения размера файла.

10. Flags – статус журнала событий. Этот элемент может иметь одно из следующих значений, приведенных в Таблице 3.

Таблица 3. Статусы журнала событий

Значение	**Пояснение**
ELF_LOGFILE_HEADER_DIRTY 0x0001	Указывает, что записи были записаны в журнал событий, но файл журнала событий не был правильно закрыт
ELF_LOGFILE_HEADER_WRAP 0x0002	Указывает, что записи в журнале событий завернуты
ELF_LOGFILE_LOGFULL_WRITTEN 0x0004	Указывает, что последняя попытка записи не удалась из-за недостатка места.
ELF_LOGFILE_ARCHIVE_SET 0x0008	Указывает, что атрибут архива был установлен для файла

11. Retention – значение хранения файла при его создании. Служба регистрации событий обычно не обновляет это значение, она опирается на конфигурацию реестра.

12. EndHeaderSize – конечный размер структуры заголовка. Размер всегда 0x30.

За заголовком следует переменное число записей событий,

представленных структурами EVENTLOGRECORD, содержащих информацию о записи события.

Структура EVENTLOGRECORD:

```
EVENTLOGRECORD {
        DWORD Length;
        DWORD Reserved;
        DWORD RecordNumber;
        DWORD TimeGenerated;
        DWORD TimeWritten;
        DWORD EventID;
        WORD  EventType;
        WORD  NumStrings;
        WORD  EventCategory;
        WORD  ReservedFlags;
        DWORD ClosingRecordNumber;
        DWORD StringOffset;
        DWORD UserSidLength;
        DWORD UserSidOffset;
        DWORD DataLength;
        DWORD DataOffset;
},
```

где:

1. Length – размер этой записи события в байтах.

2. Reserved – значение DWORD, которое всегда установлено в ELF_LOG_SIGNATURE.

3. RecordNumber – номер записи. Это значение можно использовать с флагом EVENTLOG_SEEK_READ в функции ReadEventLog, чтобы начать чтение с указанной записи.

4. TimeGenerated – время, когда эта запись была представлена. Это время измеряется в количестве секунд, прошедших с 00:00:00 1 января 1970 года, всемирное координированное время.

5. TimeWritten – время, когда эта запись была получена службой для записи в журнал. Это время измеряется в количестве секунд, прошедших с 00:00:00 1 января 1970 года, всемирное координированное время.

6. EventID – идентификатор события. Это значение зависит от источника события и используется с именем источника, чтобы найти строку описания в файле сообщений для источника события.

7. EventType – тип события. Этот элемент может иметь одно из следующих значений, приведенный в Таблице 4.

Таблица 4. Тип событий

Значение	**Пояснение**
EVENTLOG_ERROR_TYPE 0x0001	Событие ошибки
EVENTLOG_AUDIT_FAILURE 0x0010	Событие аудита отказа
EVENTLOG_AUDIT_SUCCESS 0x0008	Событие аудита успеха
EVENTLOG_INFORMATION_TYPE 0x0004	Информационное событие
EVENTLOG_WARNING_TYPE 0x0002	Событие предупреждения

8. NumStrings – Количество строк, присутствующих в журнале. Эти строки объединяются в сообщение перед его отображением для пользователя.

9. EventCategory – Категория для этого события. Значение этого значения зависит от источника события.

10. UserSidOffset – Смещение идентификатора безопасности (SID) в этой записи журнала событий. Чтобы получить имя пользователя для этого SID, используется функция LookupAccountSid.

11. DataLength – Размер специфичных для события данных в байтах.

12. DataOffset – Смещение специфичной для события информации в этой записи журнала событий в байтах.

Заключительной структурой является ELF_EOF_RECORD, содержащая

информацию, которая включается после последней записи в журнале событий и используется в журнале событий, чтобы позволить службе регистрации событий реконструировать ELF_LOGFILE_HEADER.

Структура ELF_EOF_HEADER:

```
EVENTLOGEOF {
        ULONG RecordSizeBeginning;
        ULONG One;
        ULONG Two;
        ULONG Three;
        ULONG Four;
        ULONG BeginRecord;
        ULONG EndRecord;
        ULONG CurrentRecordNumber;
        ULONG OldestRecordNumber;
        ULONG RecordSizeEnd;
},
```

где:

1. RecordSizeBeginning – начальный размер ELF_EOF_RECORD. Начальный размер всегда 0x28.

2. One – идентификатор, который помогает отличить эту запись от других записей в журнале событий. Значение всегда установлено в 0x11111111.

3. Two – идентификатор, который помогает отличить эту запись от других записей в журнале событий. Значение всегда установлено в 0x22222222.

4. Three – идентификатор, который помогает отличить эту запись от других записей в журнале событий. Значение всегда установлено в 0x33333333.

5. Four – идентификатор, который помогает отличить эту запись от других записей в журнале событий. Значение всегда установлено на

0x44444444.

6. BeginRecord – смещение к самой старой записи. Если журнал событий пуст, это устанавливается в начало этой структуры.

7. EndRecord – смещение к началу этой структуры.

8. CurrentRecordNumber – номер записи следующего события, которое будет записано в журнал событий.

9. OldestRecordNumber – номер записи самой старой записи в журнале событий. Номер записи будет 0, если журнал событий пуст.

10. RecordSizeEnd – конечный размер ELF_EOF_RECORD всегда 0x28.

11. Если журнал событий пуст, ELF_EOF_RECORD записывается сразу после ELF_LOGFILE_HEADER.

Структура ELF_LOGFILE_HEADER и структура ELF_EOF_RECORD записываются в журнал событий при создании журнала событий и обновляются каждый раз, когда событие записывается в журнал.

Запись в журнал осуществляется с помощью функции ReportEvent, которая делает запись в конец указанного журнала событий, которая передает параметры службе регистрации событий. Служба регистрации событий использует эту информацию для записи структуры EVENTLOGRECORD в журнал событий.

Синтаксис:

```
BOOL ReportEvent (
        HANDLE hEventLog,
        WORD   wType,
        WORD   wCategory,
        DWORD  dwEventID,
        PSID   lpUserSid,
        WORD   wNumStrings,
        DWORD  dwDataSize,
        LPCSTR lpStrings,
```

LPVOID lpRawData

),

где:

1. hEventLog – дескриптор журнала событий. Функция RegisterEventSource возвращает этот дескриптор.

2. wType – тип события для регистрации. Этот параметр может принимать одно из следующих значений, представленных в Таблице 5.

Таблица 5. Типы событий для регистрации

Значение	**Пояснение**
EVENTLOG_SUCCESS 0x0000	Информационное событие
EVENTLOG_AUDIT_FAILURE 0x0010	Событие аудита отказа
EVENTLOG_AUDIT_SUCCESS 0x0008	Событие аудита успеха
EVENTLOG_ERROR_TYPE 0x0001	Событие ошибки
EVENTLOG_INFORMATION_TYPE 0x0004	Информационное событие
EVENTLOG_WARNING_TYPE 0x0002	Событие предупреждения

3. wCategory – категория события. Это информация об источнике; категория может иметь любое значение.

4. dwEventID – идентификатор события. Идентификатор события указывает запись в файле сообщений, связанную с источником события.

5. lpUserSid – указатель на идентификатор безопасности текущего пользователя. Этот параметр может быть недействителен, если идентификатор безопасности не требуется.

6. wNumStrings – количество строк вставки в массиве, на которое указывает параметр lpStrings. Нулевое значение указывает на отсутствие строк.

7. dwDataSize – количество байтов необработанных (двоичных)

данных, специфичных для события, для записи в журнал. Если этот параметр равен нулю, данные для конкретного события отсутствуют.

8. lpStrings – указатель на буфер, содержащий массив строк с нулевым символом в конце, которые объединяются в сообщение до того, как средство просмотра событий отобразит эту строку для пользователя. Этот параметр должен быть допустимым указателем (или NULL), даже если wNumStrings равен нулю. Каждая строка ограничена 31 839 символами.

9. lpRawData – указатель на буфер, содержащий двоичные данные. Этот параметр должен быть допустимым указателем (или NULL), даже если параметр dwDataSize равен нулю.

Запись событий в журналы организована одним из следующих способов:

Non-wrapping. Самая старая запись находится сразу после заголовка журнала событий, а новые записи добавляются после последней добавленной записи (до ELF_EOF_RECORD). Журнал событий не переносится, пока не будет достигнут предел размера журнала событий. Размер журнала событий ограничен либо значением конфигурации MaxSize, либо объемом системных ресурсов;

Wrapping. Записи организованы в виде кольцевого буфера. Самые старые записи заменяются новыми по мере их добавления. Расположение самых старых и новых записей будет различаться. Между ELF_EOF_RECORD и самой старой записью есть некоторое пространство, потому что система сотрет целое число записей, чтобы освободить место для самой новой записи. Например, если самая новая запись имеет длину 100 байт, а две самые старые записи имеют длину 75 байт, система удалит две самые старые записи. Дополнительные 50 байтов будут использованы позже при записи новых значений.

Файл журнала событий имеет фиксированный размер, и когда для записи новых событий в файле не хватает места, то запись разделяется на две

записи. Например, если позиция для следующей записи составляет 100 байтов от конца файла, а размер записи составляет 300 байтов, первые 100 байтов будут записаны в конце файла, а следующие 200 байтов будут записаны в начале файла сразу после ELF_LOGFILE_HEADER.

Наибольший интерес для исследователей информационной безопасности представляют следующие журналы операционной системы Windows: Security (Безопасность), System (Система) и Application (Приложение) [83].

Для расширенного аудита событий безопасности специалистами по информационной безопасности может использоваться утилита Sysmon (System Monitor)[11]. С помощью данного инструмента можно получить детальную информацию о процессах, сетевых подключениях, изменениях файлов, драйверов и других событий. В связи с этим Sysmon используется для анализа безопасности, обнаружения атак и расследования инцидентов. События Sysmon записываются в отдельный журнал (Microsoft-Windows-Sysmon/Operational). Утилита Sysmon – это дополнение к стандартному журналу событий для глубокого анализа активности в системе, однако по умолчанию она не используется и требуется произвести дополнительные настройки для начала сбора расширенных данных.

2.2 Анализ событий операционной системы на предмет выявления признаков компьютерных атак

Системные события содержат большое количество информации, которую зачастую сложно понять. Более того, большая часть этой информации не несет в себе полезной нагрузки для специалиста по защите информации, так как соответствует нормальному исправному поведению операционной

[11] https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

системы. Поскольку события отражают изменения внутреннего состояния системы, ошибочные и неожиданные действия внутри системы отражаются как аномальные.

Специалистами по информационной безопасности уделяется большое внимание контролю за отдельными событиями, которые могут свидетельствовать о признаках компьютерных атак. Перечень таких событий приведен в Таблице 6. Однако ручная обработка большого количества данных неэффективна. А контроль отдельных событий не может дать однозначный ответ является ли данная активность легитимной, либо принадлежит злоумышленнику.

Таблица 6. События безопасности и признаки компьютерных атак

Event ID	**Название**	**Описание**	**Признаки возможной компьютерной атаки**
1102	Очистка журнала безопасности	Указывает на удаление записей в журнале безопасности, что может свидетельствовать о попытке скрыть следы взлома	Удаление логов для маскировки действий злоумышленника
4624	Успешный вход в систему	Фиксирует успешные аутентификации, включая подозрительные входы (например, с необычных IP-адресов или в нерабочее время)	Использование легитимных учетных данных для несанкционированного доступа
4635	Неудачный вход в систему	Множественные неудачные попытки входа могут указывать на подбор паролей или атаку методом перебора	Попытки подбора паролей или использование украденных учетных данных
4646	Вход с явными учетными данными	Может свидетельствовать о краже или неправомерном использовании учетных данных (например, создание токенов после компрометации)	Кража и использование учетных данных для доступа к системе
4662	Операция с объектом	Отслеживает действия с критическими объектами (например, репликация данных в Active Directory), что может указывать на атаку	Несанкционированный доступ или копирование конфиденциальных данных
4663	Запрос доступа к объекту	Фиксирует попытки доступа к защищенным файлам или ключам реестра без соответствующих прав	Попытки чтения или изменения системных файлов или настроек
4670	Изменение разрешений объекта	Указывает на изменение прав доступа к файлам или другим объектам, что может быть использовано для повышения привилегий	Изменение разрешений для получения несанкционированного доступа
4672	Назначение прав администратора	Фиксирует случаи предоставления административных прав новым пользователям, что может указывать на взлом	Повышение привилегий для получения контроля над системой
4698	Создание запланированной задачи	Подозрительные задачи могут использоваться для запуска вредоносного кода или поддержания доступа к системе	Создание скрытых процессов для выполнения вредоносных действий

4720	Создание новой учетной записи	Несанкционированное создание учетных записей может свидетельствовать о подготовке к атаке или внутренней угрозе	Добавление новых пользователей для обхода контроля доступа
4724	Сброс пароля учетной записи	Попытки сброса паролей могут указывать на попытку захвата учетной записи	Изменение паролей для получения контроля над учетными записями
4725	Добавление пользователя в группу	Изменение членства в привилегированных группах (например, Domain Admins) может указывать на попытку повышения привилегий	Добавление пользователей в группы с повышенными правами для расширения доступа
4726	Запрос билета Kerberos	Необычные запросы билетов Kerberos могут свидетельствовать о попытке кражи или подделки учетных данных	Кража или подделка аутентификационных токенов для несанкционированного доступа
4768	Запрос сервисного билета Kerberos	Аномальные запросы сервисных билетов могут указывать на атаки, связанные с компрометацией Kerberos	Использование уязвимостей Kerberos для получения доступа к ресурсам
4769	Проверка учетных данных контроллером домена	Множественные неудачные попытки проверки могут указывать на атаку перебора или подмену учетных данных	Попытки подбора или подмены учетных данных для аутентификации
4776	Проверка учетных данных контроллером домена	Фиксирует попытки аутентификации через контроллер домена. Аномальное количество неудачных попыток может указывать на перебор паролей.	Множественные неудачные попытки входа с одного источника, что характерно для атак перебора
7045	Установка новой службы	Установка подозрительных служб может свидетельствовать о внедрении вредоносного ПО или создании механизма для сохранения доступа.	Добавление служб для скрытого выполнения кода или поддержания контроля над системой

Важным аспектом метода обнаружения аномалий является характер желаемой аномалии. Аномалии можно разделить на следующие три категории:

- Точечные аномалии. Если отдельный экземпляр данных можно рассматривать как аномальный по отношению к остальным данным, то этот экземпляр называется точечной аномалией. Это самый простой тип аномалии, и он находится в центре внимания большинства исследований по обнаружению аномалий.

- Контекстные аномалии. Если экземпляр данных является аномальным в определенном контексте, но не в другом, то он называется контекстной аномалией (также называемой условной аномалией). Понятие контекста определяется структурой набора данных и должно быть определено как часть формулировки проблемы.

- Коллективные аномалии. Если набор связанных экземпляров данных является аномальным по отношению ко всему набору данных, это называется коллективной аномалией. Отдельные экземпляры данных в коллективной аномалии сами по себе могут не быть аномалиями, но их совместное появление в виде набора является аномальным.

Можно выделить следующие аномалии системных событий:

1. Аномалия отдельных событий – аномалия, которую можно идентифицировать с помощью одной отдельной записи события. Аномалия может выделяться двумя способами:

- Идентификатор критичности.
- Необычная для данной сети запись в оставшихся категориях.

2. Аномалия последовательности событий – характеризуется необычным порядком событий, нахождением какого-либо события в том месте, где его быть не должно, или отсутствием события в том месте, где оно ожидается. К примеру, некоторые вирусы умеют скрывать следы своего пребывания в системе путем отчистки логов.

3. Аномалия продолжительности события – вид аномалии, когда заметно, что процесс выполнялся либо слишком короткий, либо слишком длинный промежуток времени.

Обнаружение аномалий событий весьма сложная задача, которая в обычном случае состоит из двух этапов.

1. Анализ событий. Так как разные события записываются в разные журналы, стоит обращать внимание на временные метки каждого события. В программном интерфейсе Windows имеется возможность выгрузки событий из разных журналов сразу в один файл, что упрощает отслеживание событий по временным меткам, но усложняет поиск аномалий из-за увеличения объема информации в одном файле.

2. Моделирование нормального поведения системы. Помогает в поиске тех самых аномальных случаев поведения, записанных событий, которые не должны появляться при нормальной работе системы.

Аномалии в событиях операционной системы Windows могут быть признаками кибератаки на компьютер пользователя, причем такая активность может создаваться как вредоносным ПО, так и злоумышленником, пытающимся использовать уязвимости операционной системы для получения контроля над ней, похищения или блокирования информации пользователя.

2.3 Формальная модель компьютерной атаки

В работе используется подход, основанный на исследовании изменений, оставляемых в ходе компьютерной атаки в журналах Security, System и Application операционной системы Microsoft Windows [84].

В связи с этим, изучение кибератак ограничивается теми из них, которые фиксируются в операционной системе.

Рассмотрим множество U – множество всех возможных наблюдаемых действий в системе:

$$U = \{A_1, \dots, A_k, L_1, \dots, L_m\}, \quad (3)$$

$$n = k + m, \quad (4)$$

где A – множество действий злоумышленников,

L – множество легитимных действий пользователей,

n – количество всех возможных действий,

k – количество всех возможных действий злоумышленников (компьютерные атаки),

m – количество всех возможных действий легитимных пользователей.

Пусть множество $Sec = \{Sec_1, \dots, Sec_p\}$, множество булевых функций, где p – количество всех возможных событий, отображаемых в журнале Security.

Каждый элемент множества Sec имеет следующий смысл: функция Sec_i имеет значение 1, тогда и только тогда, когда за время выполнения действия происходило i-е событие из множества Sec, множества всех возможных событий журнала Security.

Пусть множество $Sys = \{Sys_1, \dots, Sys_t\}$, множество булевых функций, где t – количество всех возможных событий, отображаемых в журнале System.

Каждый элемент множества Sys имеет следующий смысл: функция Sys_i имеет значение 1, тогда и только тогда, когда за время выполнения действия происходило i-е событие из множества Sys, множества всех возможных событий журнала System.

Пусть множество $App = \{App_1, \dots, App_s\}$, множество булевых функций, где s – количество всех возможных событий, отображаемых в журнале Application.

Каждый элемент множества App имеет следующий смысл: функция App_i имеет значение 1, тогда и только тогда, когда за время выполнения действия происходило i-е событие из множества App, множества всех возможных событий журнала Application.

Определим любой элемент множества U, множества всех возможных действий как набор следующих векторов.

Каждый элемент U_i из множества U имеет вид:

$$\mathbf{U_i} = \left\{ \begin{bmatrix} Sec_{1,i} \\ \dots \\ Sec_{p,i} \end{bmatrix}, \begin{bmatrix} Sys_{1,i} \\ \dots \\ Sys_{t,i} \end{bmatrix}, \begin{bmatrix} App_{1,i} \\ \dots \\ App_{s,i} \end{bmatrix} \right\}, i = \overline{1,n}. \quad (5)$$

Пусть $\varphi(U_i)\colon U_i \to \{0,1\}$ – функционал, обозначающий выполнение действия U_i и приводящий либо к безопасному состоянию системы (0), либо к небезопасному состоянию (1).

В силу стохастической природы событий и пересечения сигнатур атак с легитимной активностью необходимо дополнительное введение порога классификации θ:

Пусть $P(\phi(U_i) = 1)$ – это число, рейтинг угрозы от 0 до 1.

Тогда правило классификации примет следующий вид:

Если $P(\phi(U_i) = 1) \geq \theta$, то действие считается атакой;

Если $P(\phi(U_i) = 1) < \theta$, то действие считается легитимным.

Определим область вероятностей компьютерной атаки как $V = \{U_i\colon U_i \in U, P(\varphi(U_i) = 1) \geq \theta\}$.

Исходя из вышеперечисленного существуют множества L (множество всех легитимных действий) и множество A (множество всех компьютерных атак): $L = \{L_1, \dots, L_k\}$ и $A = \{A_1, \dots, A_m\}$.

Элемент $U_i \in U$ является элементом множества A тогда и только тогда, когда $U_i \in V$.

Элемент $U_i \in U$ является элементом множества L тогда и только тогда, когда $U_i \notin V$.

Предложенный подход позволяет представить рассматриваемый объект (действие) в виде набора признаков, соответствующих наступлению определенного события, которое фиксируется в журналах Security, System и Application [85].

2.4 Перспективы адаптации под отечественные операционные системы в рамках импортозамещения

В условиях текущей геополитической ситуации и курса на технологический суверенитет Российской Федерации, импортозамещение

программного обеспечения, включая операционные системы, является стратегической задачей. На рынке представлен ряд российских ОС, таких как «Альт» (компании «Альт Линукс»), «РЕД ОС» («Ред Софт»), «РОСА» («НТЦ ИТ РОСА»), Astra Linux («РусБИТех») и другие, основанные на ядре Linux.

Российские ОС на базе Linux, как и Windows, используют комплексные системы журналирования для аудита и диагностики. Основным подсистемам Windows (Security, System, Application) в Linux соответствуют:

Журнал безопасности. Основные события входа и выхода, аутентификации, использования привилегий фиксируются в журналах secure, auth.log. Аналогично журналу Security в Windows, эти данные критичны для расследования инцидентов.

Системный журнал. Общесистемные сообщения, данные от ядра kern.log, служб и драйверов записываются в syslog или системный журнал journald. Это прямой аналог журнала System в Windows.

Журналы приложений. Приложения и сервисы могут писать в собственные файлы логов (/var/log/) или в централизованную систему journald. Это соответствует назначению журнала Application в Windows.

Предложенный формальный подход полностью применим к российским ОС на базе Linux с необходимой адаптацией.

Определение множеств событий: множества Sec, Sys и App будут формироваться не из журналов Windows, а из соответствующих источников данных Linux:

Sec – события из journald (записи, связанные с аутентификацией) и из журналов подсистемы auditd (события с идентификаторами AUDIT_ID, например, SYSCALL, PATH, CWD);

Sys – сообщения ядра (kernel), системных служб (systemd), диспетчеров устройств из journald и /var/log/syslog;

App – логи конкретных приложений из journald или файлов в /var/log/.

Формальная модель, представляющая действие как вектор признаков, остается неизменной. Изменяется лишь источник и способ парсинга данных для заполнения этих векторов. Логика разделения на множества легитимных действий и атак сохраняется.

Теоретический подход, изложенный в данной главе, обладает высокой степенью переносимости. Его практическая реализация для российских операционных систем семейства Linux потребует этапа адаптации, заключающегося в замене источника данных и корректировке набора анализируемых идентификаторов событий под соответствующие подсистемы. Это делает предложенную модель ценным инструментом не только для анализа безопасности Windows, но и как основу для создания систем защиты отечественного программного обеспечения в рамках политики импортозамещения.

2.5 Выводы по главе

Во второй главе диссертационного исследования рассмотрена структура системных событий операционной системы Microsoft Windows.

На основе проведенного анализа построена формальная модель действий легитимных пользователей и злоумышленников в операционной системе.

Предложенный подход к построению формальной модели позволяет обнаруживать компьютерные атаки без использования дополнительных инструментов мониторинга, что повышает его универсальность и снижает затраты на внедрение.

Сделан вывод о возможности адаптации предложенного подхода под отечественные операционные системы в рамках политики импортозамещения.

3 Формирование набора данных из записей системных событий

3.1 Алгоритм моделирования компьютерных атак и формирования набора данных

Обнаружение компьютерных атак остается актуальной темой ввиду постоянного появления новых видов киберугроз. Одним из перспективных направлений в данной области является применение искусственного интеллекта и методов машинного обучения (англ. Machine Learning, ML). Исследования [86-87] подчеркивают, что алгоритмы машинного обучения могут значительно повысить точность обнаружения атак, позволяя системам учиться на больших объемах данных.

Проблема сбора и разметки данных для систем кибербезопасности является активно развивающейся областью научных знаний . Использование машинного обучения, автоматизация разметки, анализ аномалий и стандартизация данных становятся ключевыми аспектами для повышения эффективности и точности систем защиты.

Для решения задачи автоматизированного сбора данных и их разметки в соответствии с проводимыми компьютерными атаками проектируются и создаются элементы информационной инфраструктуры, которые могут быть как физическими, так и виртуальными [88].

Для моделирования компьютерных атак исследователи часто используют матрицу MITRE ATT&CK (англ. Adversarial Tactics, Techniques, and Common Knowledge), которая представляет собой структурированную базу знаний о поведении злоумышленников, которая описывает тактики и техники, используемые в кибератаках [89]. Матрица MITRE ATT&CK для корпоративных сетей[12] включает множество тактик и техник.

[12] Доступна по ссылке https://attack.mitre.org/

С помощью фильтров можно выделить техники, которые можно выявить путем анализа системных журналов операционной системы Windows.

1. Первоначальный доступ (Initial Access)

T1566: Phishing – рассылка фишинговых писем для получения учетных данных или заражения системы;

2. Выполнение (Execution)

T1059: Command and Scripting Interpreter – выполнение команд через оболочки (CMD, PowerShell, Bash);

T1106: Native API – использование встроенных API Windows для выполнения команд;

T1053: Scheduled Task/Job – запуск заданий через планировщик Windows;

T1569: System Services – запуск вредоносного кода через системные службы;

T1204: User Execution – выполнение кода с помощью действий пользователя (например, открытие вложений в письмах);

T1047: Windows Management Instrumentation – использование WMI для удалённого выполнения команд;

3. Закрепление (Persistence)

T1197: BITS Jobs – использование Background Intelligent Transfer Service (BITS) для скрытого выполнения команд;

T1547: Boot or Logon Autostart Execution – автоматический запуск вредоносного ПО при загрузке системы;

T1037: Boot or Logon Initialization Scripts – внедрение вредоносных скриптов в сценарии загрузки;

T1136: Create Account – создание новых учетных записей с правами администратора;

T1546: Event Triggered Execution – запуск кода по определённым событиям (например, вход пользователя в систему);

T1053: Scheduled Task/Job – создание запланированных задач для автозапуска вредоносного кода;

4. Повышение привилегий (Privilege Escalation)

T1134: Access Token Manipulation – подмена маркеров доступа для выполнения команд от имени других пользователей;

T1098: Манипуляции с учетной записью - действия с учетной записью для сохранения или расширения доступа;

T1068: Exploitation for Privilege Escalation – эксплуатация уязвимостей для повышения привилегий;

5. Предотвращение обнаружения (Defense Evasion)

T1140: Deobfuscate/Decode Files or Information – расшифровка и декодирование данных;

T1112: Modify Registry – изменение реестра Windows для обхода защитных механизмов;

T1218 System Binary Proxy Execution - использование легитимных системных бинарных файлов для запуска вредоносного кода;

6. Получение учётных данных (Credential Access)

T1110: Brute Force – подбор паролей и учетных данных с использованием атак перебора;

T1555: Credentials from Password Stores – извлечение учетных данных из хранилищ паролей (например, браузеров, менеджеров паролей);

T1056: Input Capture – перехват пользовательского ввода с клавиатуры, мыши или других устройств (кейлоггеры, снифферы, перехват экранных данных);

7. Обнаружение (Discovery)

T1087: Account Discovery – поиск учетных записей в системе;

T1217: Browser Information Discovery – получение информации из браузеров (куки, история, сохраненные пароли);

T1083: File and Directory Discovery – поиск файлов и папок, содержащих чувствительную информацию;

T1012: Query Registry – исследование реестра Windows в поисках интересующих данных;

8. Боковое перемещение (Lateral Movement)

T1210: Exploitation of Remote Services – использование уязвимостей для проникновения в удаленные сервисы;

9. Сбор данных (Collection)

T1560: Archive Collected Data – упаковка собранных данных в архивы перед отправкой;

T1114: Email Collection – сбор писем и вложений из почтовых клиентов;

T1113: Screen Capture – создание снимков экрана;

10. Эксфильтрация данных (Exfiltration)

T1048: Exfiltration Over Alternative Protocol – эксфильтрация с использованием нестандартных протоколов;

T1556: Exfiltration Over Web Service – передача данных через веб-сервисы (Google Drive, Dropbox);

11. Деструктивное воздействие (Impact)

T1485: Data Destruction – уничтожение данных;

T1486: Data Encrypted for Impact – шифрование данных для получения выкупа (Ransomware).

В открытом доступе небольшое количество научных работ, дающих описание процесса моделирования компьютерных атак и формирования наборов данных для их обнаружения.

В ранних работах, таких как «Finding anomalies in windows event logs using standard deviation» [90] Джон Дуайер (John Dwyer) и Траян Мариус Трута

(Traian Marius Truta) из Университета Северного Кентукки описываются аналитические методы работы с файлами журналов. С помощью запросов SQL авторы собирают данные и вычисляют среднее количество событий определённого типа в любое время дня для любого сервера или пользователя в наборе данных. Затем определяется среднее количество событий определённого типа и стандартное отклонение этих событий, превышение которого является признаком компьютерной атаки.

Задачей выявления компьютерных атак с помощью анализа журналов операционной системы занимается группа исследователей Эгейского университета Греции во главе с Кристосом Смилиотопулосом (Christos Smiliotopoulos). В своих работах «On the detection of lateral movement through supervised machine learning and an open- source tool to create turnkey datasets from Sysmon logs», «Use of Sysmon tool to detect lateral movement attacks», «Revisiting the detection of lateral movement through Sysmon» [91-93] авторы подробно рассматривают вопросы обнаружения признаков бокового перемещения в сети с помощью инструмента Sysmon. Автором разработан и опубликован собственный инструмент для анализа журналов Python_Evtx_Analyzer[13].

Исследователи из Бельгийской Королевской военной академии под Халед Рахаль (Khaled Rahal), Арбия Риахи (Arbia Riahi) и Тибо Дебатти (Thibault Debatty) в работе «Dataset of APT Persistence Techniques on Windows Platforms Mapped to the MITRE ATT&CK Framework» делают упор на анализе тактик различных APT-группировок и выявлении техник закрепления в инфраструктуре. Для эмуляции атак и создания реалистичных сценариев использовались инструменты Caldera[14], Atomic Red Team[15]. Для создания пользовательского трафика использовался инструмент GHOSTS Framework[16].

[13] Описание инструмента по ссылке https://github.com/ChristosSmiliotopoulos/Python_Evtx_Analyzer
[14] Описание инструмента по ссылке https://caldera.mitre.org/
[15] Описание инструмента по ссылке https://www.atomicredteam.io/atomic-red-team/docs
[16] Описание инструмента по ссылке https://cmu-sei.github.io/GHOSTS/

В статье описана настройка виртуальных сред с различными пользовательскими профилями для сбора данных, а также инструменты автоматизации. Авторами собран и опубликован собственный датасет [94].

В магистерской работе Цюрихского университета «Anomaly Detection with Windows Event Logs: A comparative study between traditional and ML based approaches» Лейлы Хуссельман (Layla Husselman) проведен подробный анализ выявления некоторых техник проведения компьютерных атак как традиционным сигнатурным способом, так и с применением алгоритмов машинного обучения. Основным инструментом в ходе исследования являлся Splunk. Автором использовались журналы, опубликованные в открытом доступе. В работе приведены результаты сигнатурного выявления атак с использованием системных журналов операционной системы, которые составили 0,9165 [95].

Сравнительный анализ подходов к моделированию атак и формированию датасета приведен в Таблице 7.

Как видно из таблицы, большинство исследователей, сосредотачиваются на изучении конкретной одной тактики, ввиду трудоемкости проведения экспериментов и формирования наборов данных.

Можно выделить следующие лучшие практики, применяемые при моделировании:

1. Описание по MITRE ATT&CK. Использование общепринятой таксономии для классификации атак;

2. Виртуализация. Применение виртуальных сред для безопасного моделирования атак;

3. Автоматизация. Разработка скриптов и конфигурационных файлов для автоматизации процесса сбора и разметки данных;

4. Метаданные. Включение дополнительной информации для точной разметки событий;

5. Реалистичность. Стремление к созданию датасетов, отражающих реальные условия;

6. Баланс классов. Учет несбалансированности данных в реальных условиях, а также при обучении моделей;

7. Документация. Детальное описание процесса создания датасета для воспроизводимости.

Таблица 7. Подходы к моделированию атак и формированию датасетов

Автор, год публикации	Описание атак по MITRE ATT&CK, покрытие тактик	Источник данных	Описание моделирования атак, инструменты	Описание формирования датасета, инструменты	Дополнительная информация
John Dwyer, Traian Marius Truta, 2013	нет	Win Event Logs	частично, ручной метод	частично, SQL-запросы	
Christos Smiliotopoulos at al., 2022, 2023	да, боковое перемещение	Sysmon	частично, ручной метод	да, ETCExptool[17]	
Khaled Rahal, Arbia Riah, Thibault Debatty, 2024	да, закрепление	Win Event Logs, Sysmon	да, для атак Caldera, Atomic Red Team, для пользователей GHOSTS Framework[18]	нет	Автоматизация атак с помощью YAML-правил, Chocolatey для развертывания инфраструктуры, валидация выполнения скриптов
Layla Husselman, 2025	да, повышение привилегий, получение учетных данных, предотвращение обнаружения, боковое перемещение	Win Event Logs	нет, использовались публичные наборы данных	да, Splunk[19]	В работе приведены результаты сигнатурного выявления данных атак, которые составили 0,9165
Настоящая работа	да, повышение привилегий, закрепление, получение учетных данных, предотвращение обнаружения	Win Event Logs	да, для атак Caldera, Atomic Red Team, для пользователей скрипты Python	да, Elastic[20]	Автоматизация атак с помощью YAML-правил, валидация выполнения скриптов, передача метаданных об атаках и пользовательской активности

[17] Описание инструмента по ссылке github.com/ChristosSmiliotopoulos/evtx_To_CSV_ExportTool
[18] Описание инструмента по ссылке https://cmu-sei.github.io/GHOSTS/
[19] Описание инструмента по ссылке https://www.splunk.com/
[20] Описание инструмента по ссылке https://www.elastic.co/

Проведенный анализ существующих подходов и лучших практик дает возможность сформулировать алгоритм моделирования компьютерных атак и формирования набора данных из записей событий операционной системы [96]. Визуальное представление алгоритма в виде блок-схем представлено на Рисунке 14.

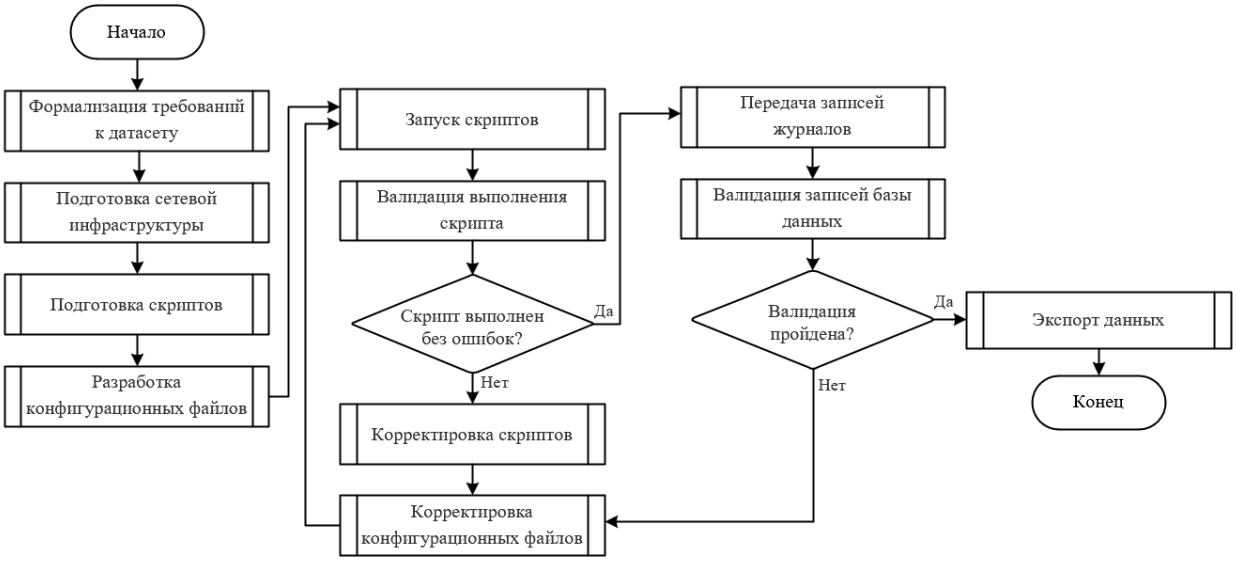

Рис. 14. Алгоритм моделирования атак и формирования набора данных

Данный алгоритм включает в себя следующие шаги:

1. Формализация требований к датасету. На этапе планирования необходимо четко определить целевые показатели: соотношение легитимных и атакующих событий, покрытие тактик и техник согласно MITRE ATT&CK, требуемый объем записей, временные рамки. Данный этап позволяет объективно оценить результат на шагах валидации и собрать качественный датасет.

2. Подготовка сетевой инфраструктуры. На данном этапе готовится сетевая инфраструктура, на которой будут эмулироваться компьютерные атаки и легитимные действия пользователей. Также в инфраструктуре разворачиваются средства автоматизации и база данных для хранения полученных записей событий операционной системы;

3. Подготовка скриптов. Ведется разработка набора скриптов атак и пользовательских действий. Атаки должны привязываться к матрице MITRE ATT&CK. Эмуляция пользовательских действий необходима для создания реалистичных данных для датасета. Пример атакующего скрипта приведен на Рисунке 15;

```
# add_user_to_admin_group.ps1
param(
    [string]$Username = "alice",
    [string]$Group = "Administrators",
    [string]$Domain = $env:COMPUTERNAME
)

Write-Host "[T1098] Начало манипуляции с учетной записью" -ForegroundColor Yellow

try {
    # Проверяем текущее членство
    $currentMembers = net localgroup "$Group" | Select-String $Username
    if ($currentMembers) {
        Write-Host "[INFO] Пользователь $Username уже в группе $Group" -ForegroundColor Gray
        return $true
    }

    # Добавляем пользователя в группу
    Write-Host "[ACTION] Добавление $Username в группу $Group..." -ForegroundColor Cyan
    net localgroup "$Group" $Username /add

    # Проверяем успешность
    Start-Sleep -Seconds 2
    $newMembers = net localgroup "$Group" | Select-String $Username

    if ($newMembers) {
        Write-Host "[SUCCESS] Пользователь $Username успешно добавлен в $Group" -ForegroundColor Green
```

Рис. 15. Пример атакующего скрипта

4. Разработка конфигурационных файлов. Для автоматизации процесса формирования датасета и его воспроизводимости создаются конфигурационные YAML-правила – файлы, в которых описываются условия атаки (например, уязвимость, версия ПО), действия (эксплуатация уязвимости, выполнение команд, выполнение кода), параметры (IP-адреса, порты), зависимости (например, необходимость предварительной аутентификации), а также количество итераций запуска скриптов. Инструменты автоматизации (такие как Metasploit Framework[21]) загружают эти правила и выполняют атаки в автоматическом или полуавтоматическом режиме. Для избежания зацикливания в качестве одного из параметров указывается количество

[21] Описание инструмента по ссылке https://www.metasploit.com/

итераций выполнения скрипта. Пример параметров, содержащихся в конфигурационном файле приведен на Рисунке 16;

```yaml
# 1. ACCOUNT_MANIPULATION (T1098)
- attack_id: "attack_001"
  name: "Account Manipulation - Local Group Modification"
  mitre_technique: "T1098"
  mitre_tactic: "TA0004"
  description: "Добавление пользователя в группу локальных администраторов"
  target_host: "win10-client-01"
  execution_time: "random"  # random, scheduled, immediate

  stages:
    - stage: 1
      action: "initial_compromise"
      script: "phishing_delivery.ps1"
      validator: "check_process_running.ps1"

    - stage: 2
      action: "account_enumeration"
      script: "enum_local_users.ps1"
      validator: "check_file_exists.ps1"

    - stage: 3
      action: "group_modification"
      script: "add_user_to_admin_group.ps1"
      parameters:
        username: "alice"
        group: "Administrators"
      validator: "check_group_membership.ps1"

    - stage: 4
      action: "persistence_verification"
      script: "verify_admin_access.ps1"
      validator: "check_success_code.ps1"
```

Рис. 16. Настройка параметров в конфигурационном файле

5. Запуск скриптов. Согласно конфигурационному файлу осуществляется запуск скриптов, эмулирующих компьютерные атаки и легитимные действия пользователей;

6. Валидация выполнения скрипта. С помощью специальных индикаторов осуществляется проверка выполнения скрипта. Пример функции-валидатора приведен на Рисунке 17;

7. В случае, если валидация не пройдена, производится доработка скриптов, а также конфигурационных файлов, в которых указывается новая последовательность выполнения и количество итераций. Далее осуществляется повторный запуск скриптов;

```
function Test-BitsJobExistence {
    param($JobName)

    $checkName = "JobExists"
    Write-Host "[VALIDATION] Проверка существования задания BITS: $JobName" -ForegroundColor Cyan

    try {
        # Способ 1: Через PowerShell cmdlet
        $job = Get-BitsTransfer -Name $JobName -ErrorAction SilentlyContinue

        if (-not $job) {
            # Способ 2: Через bitsadmin
            $bitsadminOutput = bitsadmin /list /allusers 2>&1 | Select-String $JobName
            $exists = $null -ne $bitsadminOutput
        } else {
            $exists = $true
        }

        $result = @{
            CheckName = $checkName
            Status = if ($exists) { "PASS" } else { "FAIL" }
            Details = if ($exists) { "Задание BITS найдено" } else { "Задание BITS не найдено" }
            Timestamp = Get-Date -Format "HH:mm:ss"
        }

        if ($exists) {
            $ValidationResults.ArtifactsFound += "BITS Job: $JobName"
        }

        return $result
    }
```

Рис. 17 Валидация выполнения скрипта

8. Передача записей событий. На данном этапе осуществляется передача и сохранение информации из системных событий операционной системы. Также передаются метаданные (например, вид атаки или действия пользователя) и уникальный идентификатор сессии;

9. Валидация записей базы данных. На данном этапе проводится проверка полученных записей на предмет достаточности и сбалансированности данных, а также иных критериев, установленных на этапе планирования. Валидация проводится с помощью специально разработанного скрипта, как это приведено на Рисунке 18;

10. В случае, если данных недостаточно, либо фиксируется дисбаланс классов, с помощью редактирования конфигурационного файла указывается недостающее количество необходимых скриптов и проводятся дополнительные итерации их выполнения;

```
class DatasetValidator:
    def __init__(self, config_path):
        with open(config_path, 'r') as f:
            self.config = json.load(f)

        self.results = {
            "validation_timestamp": datetime.now().isoformat(),
            "checks_passed": 0,
            "checks_failed": 0,
            "details": []
        }

    def check_class_balance(self, data):
        """Проверка сбалансированности классов"""
        check_name = "Class Balance Check"

        try:
            # Анализируем метки классов
            class_counts = Counter(data['fields.category '])

            total_samples = len(data)
            required_ratio = self.config['validation']['class_balance']['target_ratio_attack_to_normal']

            # Парсим требуемое соотношение
            attack_ratio, normal_ratio = map(int, required_ratio.split(':'))
            target_ratio = attack_ratio / normal_ratio

            # Считаем текущее соотношение
            attack_samples = sum(count for label, count in class_counts.items()
                                 if label != 'USER_ACTION')
            normal_samples = class_counts.get('USER_ACTION', 0)
```

Рис. 18. Валидация записей датасета

11. Экспорт данных. На финальном этапе алгоритма осуществляется сохранение полученных данных в удобном для дальнейшего использования формате.

3.2 Реализация разработанного алгоритма

В рамках исследования спроектирован и развернут исследовательский стенд, в который входит 4 типовых сетевых сегмента:

1. Сегмент атакующих – включает пару машин на Windows и Linux (Kali), с которых производятся атаки;

2. Сегмент пользователей – набор машин на Linux и Windows, имитирующих рабочие места сотрудников. На них расположены разработанные скрипты, имитирующие легитимные действия пользователей;

3. Серверный сегмент – сегмент, имитирующий серверы атакуемой организации, в него входят такие сервисы как файловый сервер, сервер Active Directory, WEB-сервер и другие;

4. Сегмент обработки данных – тут размещаются серверы-обработчики, в том числе сервер базы данных, в которую передаются записи из системных событий пользовательской машины и метаданные, необходимые для разметки;

На Рисунке 19 приведены основные элементы информационной инфраструктуры для проведения эксперимента.

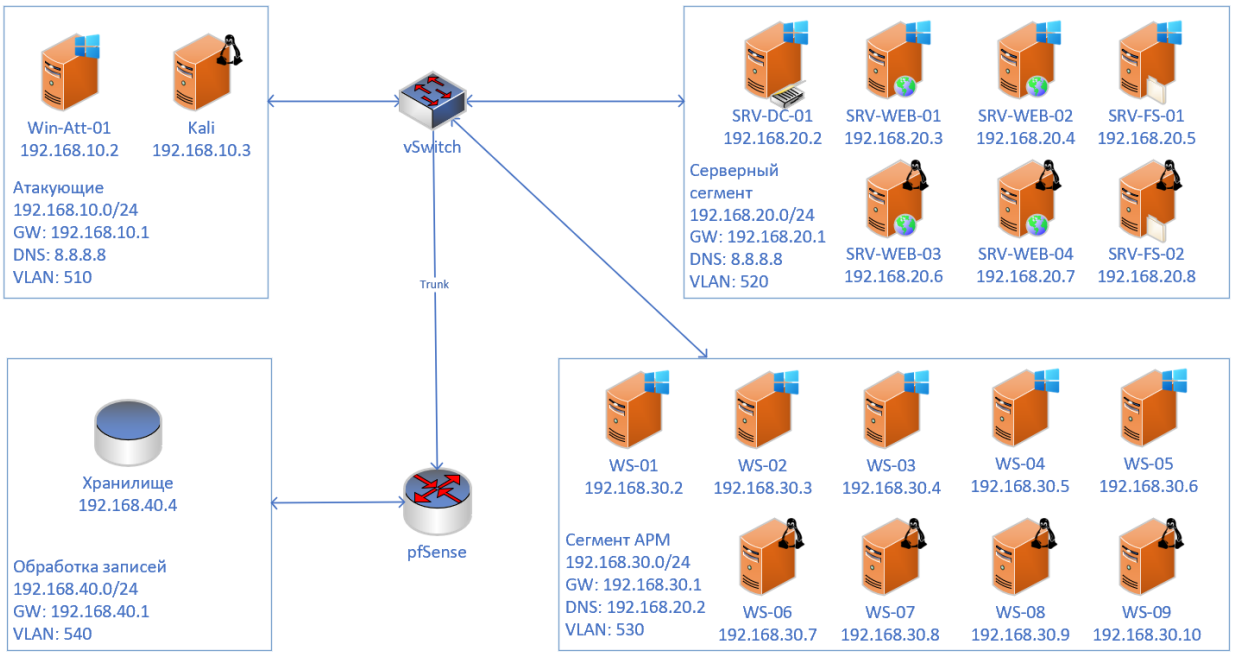

Рис. 19. Основные элементы исследовательского стенда

В качестве программной среды для исследования выбрана следующая конфигурация: гипервизор Oracle VM VirtualBox[22] для эмуляции элементов компьютерной сети; специализированное ПО Winlogbeat[23] для передачи системных событий; база данных Elastic для хранения информации из системных событий и метаданных.

[22] Описание инструмента по ссылке https://www.virtualbox.org/
[23] Описание инструмента по ссылке https://www.elastic.co/beats/winlogbeat

Действия, воспроизводимые в информационной системе в рамках исследования, включают в себя две группы:

1. Компьютерные атаки;

2. Легитимные действия пользователей.

Для моделирования компьютерных атак разработан ряд атакующих скриптов:

1. Подбор паролей для компрометации учетных записей - атака на SMB-сервис с перебором логинов и паролей (тактика TA0006: Получение учетных данных);

2. Изменение параметров учетных записей для получения повышенных прав или обхода контроля доступа - добавление пользователя в группу администраторов (тактика TA0004: Повышение привилегий);

3. Использование встроенной службы Windows Background Intelligent Transfer Service для скрытой загрузки и выполнения вредоносного кода (тактика TA0003: Закрепление);

4. Запуск вредоносного кода через доверенные системные утилиты - выполнение DLL через rundll32.exe для обхода антивируса (тактика TA0005: Предотвращение обнаружения).

Для моделирования легитимных действий пользователей разработаны скрипты, эмулирующие распространенные сценарии офисной работы:

1. Просмотр веб-страниц и видеоконтента при помощи браузера;

2. Работа с пакетом офисных приложений;

3. Получение и отправка электронной почты;

4. Использование файлового хранилища.

С целью автоматизации запуска компьютерных атак и различных пользовательских сценариев написан конфигурационный файл и разработан управляющий скрипт. Данный программный продукт [97] также предназначен

для передачи метаданных, для дальнейшей передачи в базу данных и разметки полученных результатов.

Передаваемые метаданные содержат информацию о виде действия: компьютерная атака (ATTACK), либо легитимные действия (USER_ACTION), а также номер техники из матрицы MITRE ATT&CK и конкретный тип пользовательских действий (WEB, OFFICE, FTP, MAIL).

Непосредственно процесс моделирования компьютерных атак и действий пользователей включает в себя следующие этапы:

1. Исследователь с использованием управляющего скрипта и конфигурационных файлов инициирует запуск различных компьютерных атак или пользовательских сценариев;

2. После завершения компьютерной атаки или пользовательского сценария и валидации успешности выполнения скрипта, события операционной системы передаются с использованием инструмента Winlogbeat в хранилище, где обрабатываются и сохраняются в базе данных, где происходит дополнительная валидация полученных результатов. Помимо данных системных журналов также передаются метаданные и уникальный идентификатор;

3. После реализации каждой компьютерной атаки или пользовательского сценария пользовательская виртуальная машина сбрасывается в исходное состояние;

4. Из базы данных информация о записях системных событий, а также метаданные экспортируются в удобном формате для дальнейшей обработки.

3.3. Описание полученного набора данных

В результате эксперимента сформирован набор данных, содержащий размеченную информацию о записях системных событий, полученных с

виртуальной машины, на которой были реализованы различные компьютерные атаки и сценарии пользовательской работы [98]. Общий вид полученного набора представлен на Рисунке 20. Итоговое количество собранных событий составило 1 473 559.

	@timestamp	winlog.channel	event.code	fields.category	fields.session	fields.sub_category
0	Apr 28, 2025 @ 15:11:40.080	Security	4624	USER_ACTION	20250427221113	OFFICE
1	Apr 28, 2025 @ 15:11:40.080	Security	4672	USER_ACTION	20250427221113	OFFICE
2	Apr 28, 2025 @ 15:11:40.048	System	6	USER_ACTION	20250427221113	OFFICE
3	Apr 28, 2025 @ 15:11:40.036	Security	4624	USER_ACTION	20250427221113	OFFICE
4	Apr 28, 2025 @ 15:11:40.036	Security	4672	USER_ACTION	20250427221113	OFFICE
...	...	...	...	...	...	...
1642041	May 1, 2025 @ 21:41:15.803	Security	4672	ATTACK	20250501214051	SYSTEM_BINARY_PROXY_EXECUTION_(T1218)
1642042	May 1, 2025 @ 21:41:15.799	Application	4625	ATTACK	20250501214051	SYSTEM_BINARY_PROXY_EXECUTION_(T1218)
1642043	May 1, 2025 @ 21:41:15.748	System	24	ATTACK	20250501214051	SYSTEM_BINARY_PROXY_EXECUTION_(T1218)
1642044	May 1, 2025 @ 21:41:15.748	System	1	ATTACK	20250501214051	SYSTEM_BINARY_PROXY_EXECUTION_(T1218)
1642045	May 1, 2025 @ 21:41:15.748	Security	4616	ATTACK	20250501214051	SYSTEM_BINARY_PROXY_EXECUTION_(T1218)

1473559 rows × 6 columns

Рис. 20. Визуальное представление сформированного набора данных

Описание полей в наборе данных:

1. @timestamp – дата и время регистрации события;
2. winlog.channel – системный журнал Windows, в котором зафиксировано событие. Принимает одно из следующих значений: Security, System, Application;
3. event.code – код события Event ID из набора Windows Event Logs;
4. fields.category – указывает на вид события и принимает два значения: ATTACK – компьютерная атака, USER_ACTION – пользовательские действия;
5. fields.session – уникальный идентификатор сессии, в рамках которого зафиксировано событие;
6. fields.sub_category – конкретный вид компьютерной атаки или пользовательского действия.

Компьютерные атаки маркируются согласно матрицы MITRE ATT&CK:

ACCOUNT_MANIPULATION_(T1098): тактика TA0004 Повышение привилегий (Privilege Escalation);

CREDENTIAL_BRUTEFORCE_(T1110): тактика TA0006 Получение учетных данных (Credential Access);

BITS_JOBS_(T1197): тактика TA0003 Закрепление (Persistence);

SYSTEM_BINARY_PROXY_EXECUTION_(T1218): тактика TA0005 Предотвращение обнаружения (Defense Evasion);

Виды пользовательских действий:

OFFICE – работа с офисными приложениями;

WEB – работа в Интернет-браузере;

MAIL – получение и отправка электронной почты;

FTP – работа с файловым хранилищем.

В рамках эксперимента была поставлена задача собрать сбалансированный набор данных, содержащий примерно одинаковое количество записей для каждого класса. Для этой цели были отфильтрованы сессии для ряда категорий. Результаты проверки датасета на сбалансированность приведены в Таблице 8.

Анализ распределения категорий и субкатегорий в датасете позволяет оценить степень сбалансированности набора данных, собранного в рамках эксперимента. Датасет включает две основные категории: ATTACK (атаки) и USER_ACTION (пользовательские действия), каждая из которых разделена на субкатегории. Общий объем данных составляет 921 777 уникальных записей (483 572 для ATTACK, что составляет 52,5% от общего объема и 438 205 для USER_ACTION – 47,5% от общего объема), что свидетельствует о достижении равновесия между классами. Незначительное преобладание ATTACK (на 5%) обусловлено особенностями сбора данных, поскольку атаки генерируют больше событий по сравнению с пользовательскими действиями.

Таблица 8. Распределение категорий с субкатегорий в датасете

fields.category	ATTACK	USER_ACTION
fields.sub_category		
ACCOUNT_MANIPULATION_(T1098)	125751	
CREDENTIAL_BRUTEFORCE_(T1110)	131530	
BITS_JOBS_(T1197)	124139	
SYSTEM_BINARY_PROXY_EXECUTION_(T1218)	102152	
OFFICE		136035
WEB		124682
MAIL		94057
FTP		83431
Всего	483572	438205

Категория ATTACK включает четыре субкатегории, соответствующие техникам из матрицы MITRE ATT&CK:

ACCOUNT_MANIPULATION_(T1098): 125 751 запись (26% от ATTACK);

CREDENTIAL_BRUTEFORCE_(T1110): 131 530 записей (27,2% от ATTACK);

BITS_JOBS_(T1197): 124 139 записей (25,7% от ATTACK);

SYSTEM_BINARY_PROXY_EXECUTION_(T1218): 102 152 записи (21,1% от ATTACK).

Распределение субкатегорий ATTACK демонстрирует высокую степень сбалансированности: диапазон варьируется от 102 152 до 131 530 записей, с коэффициентом вариации около 9,5%. Это свидетельствует об успешной фильтрации данных для минимизации дисбаланса, что важно для предотвращения переобучения моделей на наиболее представленных классах.

Наибольшее количество записей приходится на CREDENTIAL_BRUTEFORCE_(T1110), что отражает частоту реализации этой техники в реальных атаках, в то время как SYSTEM_BINARY_PROXY_EXECUTION_(T1218) представлено наименьшим объемом ввиду более специфических условий ее применения.

Такая вариативность подчеркивает необходимость тщательной фильтрации сессий для поддержания баланса при формировании датасетов, обеспечивая равные возможности для моделирования различных векторов угроз.

Категория USER_ACTION охватывает четыре субкатегории, отражающие типичные пользовательские активности в операционной системе:

OFFICE: 136 035 записей (31% от USER_ACTION);

WEB: 124 682 записи (28,4% от USER_ACTION);

MAIL: 94 057 записей (21,5% от USER_ACTION);

FTP: 83 431 запись (19,1% от USER_ACTION).

Распределение субкатегорий USER_ACTION также демонстрирует стремление к максимальной сбалансированности: диапазон варьируется от 83 431 до 136 035 записей, с коэффициентом вариации около 18%. Это указывает на менее идеальный баланс, что может быть обусловлено естественными различиями в частоте использования приложений в реальных сценариях. Наибольший объем приходится на OFFICE, что отражает преобладание офисных задач в корпоративных средах.

В целом, датасет достигает близкой к равномерной представленности классов на уровне категорий (52,5% ATTACK и 47,5% USER_ACTION) и субкатегорий, с коэффициентами вариации 9,5% для ATTACK и 18% для USER_ACTION. Это минимизирует смещение в моделях машинного обучения, обеспечивая репрезентативность для различения вредоносной активности и легитимных действий на основе EventID в системных событиях. Стремление к сбалансированности датасета обусловлена повышением точности обнаружения угроз в реальных системах. Такие распределения подтверждают эффективность формальной модели, где каждая активность оставляет уникальный след, и способствуют надежному анализу событий.

Согласно разработанной формальной модели, каждая компьютерная атака и пользовательская активность оставляет отличительный набор значений в системных журналах операционной системы.

На Рисунках 21-23 наглядно представлено распределение 10 самых часто встречающихся EventID для каждого вида активности в журналах Security, System и Application.

Для наглядности распределения событий по всем категориям удобно использовать тепловую карту, как это представлено на Рисунке 24.

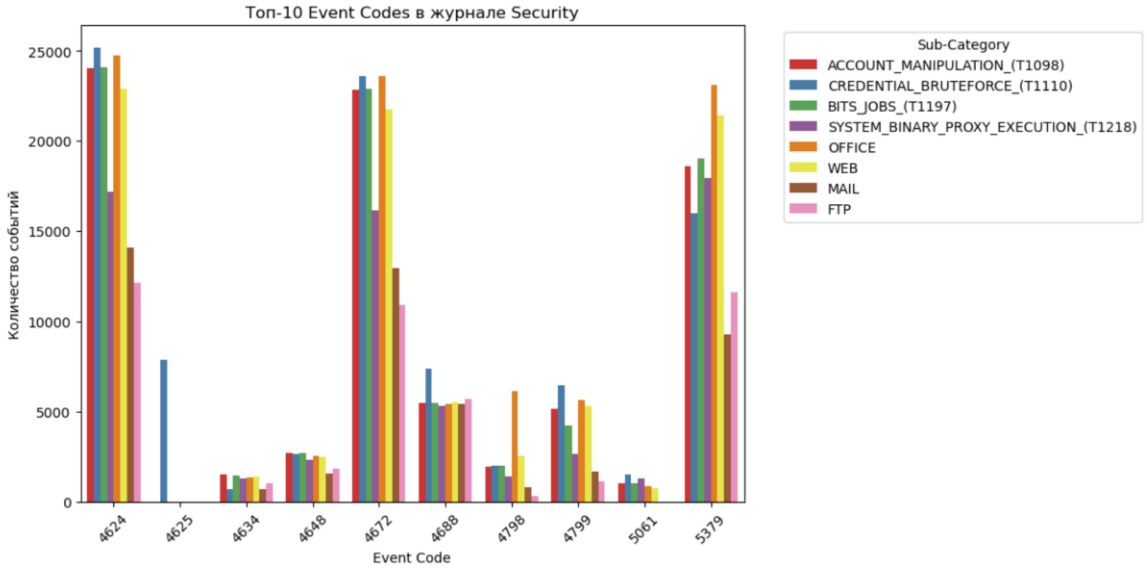

Рис. 21 Распределение EventID в журнале Security

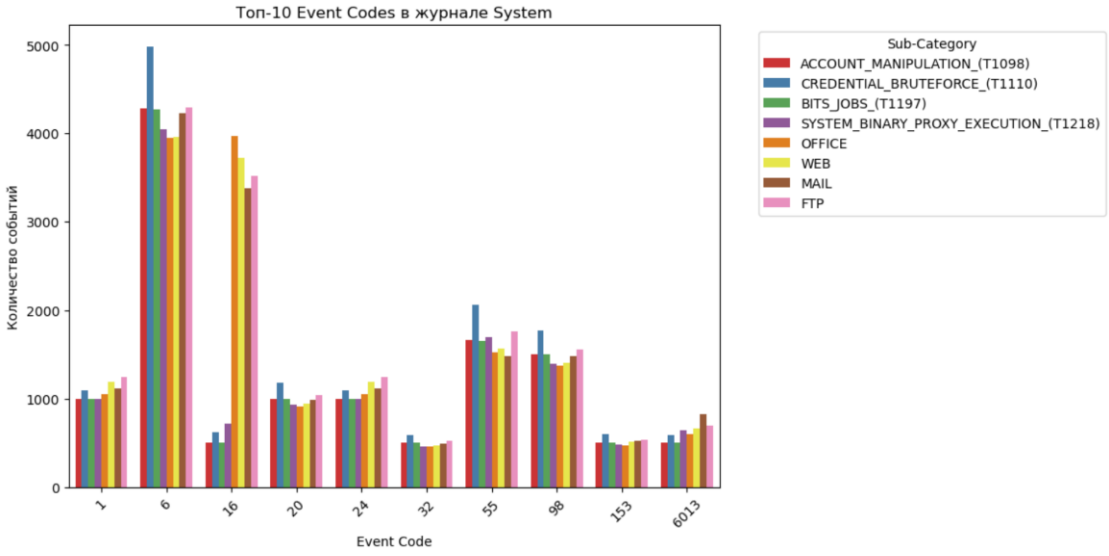

Рис. 22 Распределение EventID в журнале System

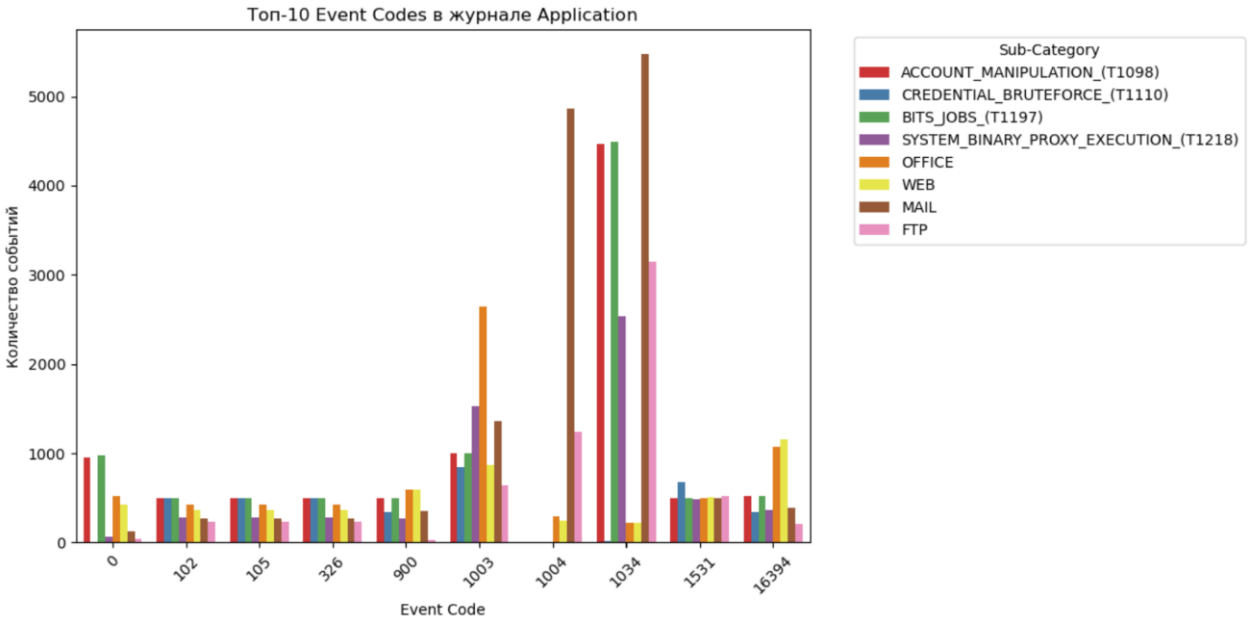

Рис. 23 Распределение EventID в журнале Application

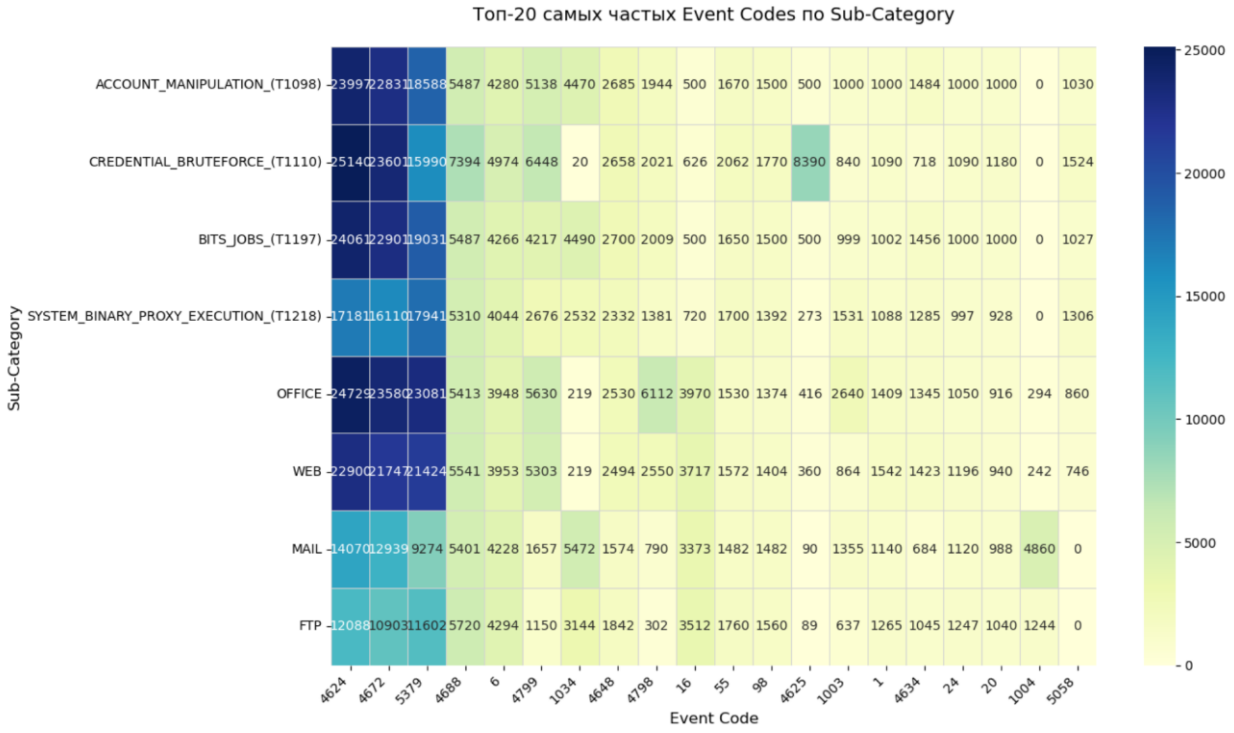

Рис. 24 Распределение EventID по всем категориям

Наборы EventID для каждого вида компьютерной атаки и пользовательского действия представлены в Таблице 9.

Таблица 9. Наборы EventID для каждой категории

fields.sub_category	event.code
ACCOUNT_ MANIPULATION_ (T1098)	0, 1, 6, 12, 14, 16, 18, 20, 24, 25, 27, 32, 55, 98, 102, 105, 153, 172, 326, 900, 902, 903, 1000, 1001, 1003, 1033, 1034, 1066, 1531, 4608, 4616, 4624, 4625, 4634, 4648, 4672, 4688, 4696, 4738, 4781, 4798, 4799, 4826, 4902, 5024, 5033, 5058, 5059, 5061, 5379, 5382, 5615, 5617, 6005, 6009, 6013, 7001, 7026, 9027, 10010, 10016, 16384, 16394, 16962, 16977, 16983, 50036, 50103, 51046
CREDENTIAL_ BRUTEFORCE_(T1110)	1, 6, 12, 14, 15, 16, 18, 20, 24, 25, 27, 32, 41, 55, 98, 102, 105, 129, 153, 172, 256, 326, 508, 900, 902, 903, 1000, 1001, 1003, 1014, 1033, 1034, 1066, 1101, 1130, 1531, 4608, 4616, 4624, 4625, 4634, 4648, 4672, 4688, 4696, 4798, 4799, 4826, 4902, 5024, 5033, 5058, 5059, 5061, 5379, 5382, 5615, 5617, 6005, 6008, 6009, 6013, 7001, 7026, 7045, 8197, 8230, 9027, 10010, 10016, 12288, 12289, 16384, 16394, 16962, 16977, 16983, 50036, 50103, 51046
BITS_JOBS_(T1197)	0, 1, 6, 12, 14, 16, 18, 20, 24, 25, 27, 32, 55, 98, 102, 105, 129, 153, 172, 326, 900, 902, 903, 1000, 1001, 1003, 1033, 1034, 1066, 1531, 4608, 4616, 4624, 4625, 4634, 4648, 4672, 4688, 4696, 4798, 4799, 4826, 4902, 5024, 5033, 5058, 5059, 5061, 5379, 5382, 5615, 5617, 6005, 6009, 6013, 7001, 7026, 9027, 10010, 10016, 16384, 16394, 16962, 16977, 16983, 50036, 50103, 51046
SYSTEM_BINARY_PROXY_ EXECUTION_(T1218)	0, 1, 6, 12, 14, 15, 16, 18, 20, 24, 25, 27, 32, 34, 35, 37, 41, 52, 55, 98, 102, 105, 129, 153, 172, 256, 300, 301, 302, 326, 507, 508, 900, 902, 903, 1000, 1001, 1003, 1014, 1024, 1033, 1034, 1040, 1066, 1101, 1130, 1531, 4199, 4608, 4616, 4624, 4625, 4634, 4648, 4672, 4688, 4696, 4798, 4799, 4826, 4902, 5024, 5033, 5058, 5059, 5061, 5379, 5382, 5611, 5615, 5617, 6005, 6008, 6009, 6013, 7000, 7001, 7009, 7026, 7040, 7045, 8197, 8230, 9027, 10000, 10001, 10010, 10016, 10400, 12288, 12289, 16384, 16394, 16962, 16977, 16983, 50036, 50103, 51046
OFFICE	0, 1, 6, 12, 14, 15, 16, 18, 20, 24, 25, 27, 29, 32, 34, 35, 37, 41, 52, 55, 98, 102, 103, 105, 129, 153, 172, 256, 300, 301, 302, 326, 507, 508, 900, 902, 903, 1001, 1003, 1004, 1013, 1014, 1033, 1034, 1040, 1042, 1066, 1101, 1130, 1531, 4199, 4608, 4616, 4624, 4625, 4634, 4648, 4672, 4688, 4696, 4798, 4799, 4826, 4902, 5024, 5033, 5058, 5059, 5061, 5379, 5382, 5615, 5617, 6005, 6008, 6009, 6013, 7000, 7001, 7009, 7026, 7040, 7045, 8197, 8198, 8224, 8230, 9027, 10000, 10001, 10010, 10016, 10400, 11707, 12288, 12289, 16384, 16394, 16962, 16977, 16983, 50036, 50103, 51046
WEB	0, 1, 6, 12, 14, 15, 16, 18, 20, 24, 25, 27, 32, 34, 35, 37, 41, 43, 44, 55, 98, 102, 103, 105, 129, 134, 153, 172, 256, 300, 301, 302, 326, 508, 900, 902, 903, 1000, 1001, 1002, 1003, 1004, 1013, 1033, 1034, 1040, 1042, 1066, 1101, 1130, 1531, 4000, 4100, 4608, 4616, 4624, 4625, 4634, 4648, 4672, 4688, 4696, 4797, 4798, 4799, 4826, 4902, 5024, 5033, 5058, 5059, 5061, 5379, 5382, 5615, 5617, 6005, 6008, 6009, 6013, 7001, 7023, 7026, 7040, 7045, 8197, 8198, 8224, 8230, 9027, 10000, 10001, 10010, 10016, 11707, 12288, 12289, 16384, 16394, 16962, 16977, 16983, 50036, 50103, 51046
MAIL	0, 1, 6, 12, 14, 15, 16, 18, 20, 24, 25, 27, 32, 34, 37, 41, 55, 98, 102, 103, 105, 129, 153, 172, 256, 300, 301, 302, 326, 508, 510, 900, 902, 903, 1003, 1004, 1013, 1033, 1034, 1035, 1040, 1042, 1066, 1101, 1531, 4608, 4616, 4624, 4625, 4634, 4648, 4672, 4688, 4696, 4798, 4799, 4826, 4902, 5024, 5033, 5379, 5615, 5617, 6005, 6008, 6009, 6013, 7000, 7001, 7009, 7026, 7040, 7045, 8197, 8230, 10000, 10001, 10010, 10016, 11728, 12288, 12289, 16384, 16394, 16962, 16977, 16983, 50036, 50103, 51046
FTP	0, 1, 6, 12, 14, 15, 16, 18, 20, 24, 25, 27, 32, 34, 35, 37, 41, 55, 98, 102, 103, 105, 129, 153, 172, 256, 300, 301, 302, 326, 508, 900, 902, 903, 1003, 1004, 1013, 1033, 1034, 1035, 1040, 1042, 1066, 1101, 1531, 4608, 4616, 4624, 4625, 4634, 4648, 4672, 4688, 4696, 4798, 4799, 4826, 4902, 5024, 5033, 5379, 5382, 5615, 5617, 6005, 6008, 6009, 6013, 7001, 7026, 7040, 7045, 8197, 8230, 10000, 10001, 10010, 10016, 11728, 12288, 12289, 16384, 16394, 16962, 16977, 16983, 50036, 50103, 51046

Таким образом, составленная формальная модель нашла свое подтверждение в ходе эксперимента, демонстрирующего, что каждому виду компьютерной атаки или пользовательского действия соответствует характерный набор значений EventID из системных журналов Security, System и Application операционной системы Windows.

3.4 Выводы по главе

В третьей главе исследования предложен модифицированный алгоритм моделирования атак и сбора датасета из записей системных событий операционной системы, учитывающий различные подходы и лучшие практики.

На основании разработанного алгоритма спроектирована и развернута виртуальная сетевая инфраструктура, предназначенная для автоматизированного извлечения файлов системных событий операционной системы после запуска заранее подготовленных скриптов, моделирующих компьютерные атаки и легитимные действия пользователей. Дано описание входящих в состав системы модулей и их назначения.

Составлено подробное описание полученного набора данных, а также проведен его анализ. В результате проведенного эксперимента подтверждена формальная модель, согласно которой каждому виду компьютерной атаки или пользовательского действия соответствует отличительный набор записей системных событий операционной системы.

4 Апробация разработанной методики выявления компьютерных атак

4.1 Преобразование набора данных и выбор метрик

Для решения предстоящей задачи по обучению моделей необходимо произвести преобразование полученных данных о записях системных событий операционной системы в наборы признаков объекта (действия).

Каждый признак будет иметь числовое значение, соответствующее количеству зафиксированных Event ID в ходе реализации сценария атаки или пользовательской активности, а также 0 в случае, если запись о наступлении данного события отсутствует.

Набор для обучения моделей формируется следующим образом: в качестве столбцов используются уникальные идентификаторы событий Event ID, в качестве строк – наборы значений признаков исследуемых объектов (уникальные сессии, соответствующие проводимым сценарием, поле *fields.session*).

Также необходимо преобразовать название категорий и субкатегорий в числовые значения. Преобразование осуществляется по следующим правилам.

Поле *fields.category* переводится в целевую переменную *target*, которая имеет значение 1, соответствующая «ATTACK» и 0, соответствующее «USER_ACTION».

Из fields.sub_category формируется поле sub_category_encoded в соответствии со словарем: {'ACCOUNT_MANIPULATION_(T1098)': 0, 'CREDENTIAL_BRUTEFORCE_(T1110)': 1, 'BITS_JOBS_(T1197)': 2, 'SYSTEM_BINARY_PROXY_EXECUTION_(T1218)': 3, 'OFFICE': 4, 'WEB': 5, 'MAIL': 6, 'FTP': 7}.

Общий вид обработанного датасета приведен на Рисунке 25.

На следующем этапе для решения задачи по обнаружению признаков компьютерных атак необходима разработка классификатора, который с учетом значений набора признаков объекта сможет отнести исследуемые события к одной из двух категорий: компьютерная атака или легитимное действие пользователя.

	fields.session	0	1	6	12	14	15	16	18	20	...	16384	16394	16962	16977	16983	50036	50103	51046	sub_category_encoded	target
0	20250412144135	1	4	18	2	2	0	2	2	4	...	2	2	2	2	2	2	2	2	5	0
1	20250412144551	0	3	19	2	2	0	2	2	4	...	0	1	2	2	2	2	2	2	4	0
2	20250412145345	1	4	17	2	2	0	2	2	4	...	2	2	2	2	2	2	2	2	5	0
3	20250412145802	0	3	17	2	2	0	2	2	4	...	0	1	2	2	2	2	2	2	4	0
4	20250412150131	2	4	17	2	2	0	2	2	4	...	2	3	2	2	2	2	2	2	5	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	20250428232302	1	3	9	1	1	0	1	1	2	...	3	0	1	1	1	0	0	1	5	0
9996	20250519131652	0	1	8	1	1	0	1	1	2	...	0	1	1	0	1	1	0	1	0	1
9997	20250518004124	0	2	0	0	1	0	2	1	0	...	0	1	1	0	3	1	1	1	2	1
9998	20250405154854	0	3	8	2	1	2	2	1	2	...	1	1	1	1	1	0	2	0	3	1
9999	20250331135527	0	1	10	3	0	0	1	1	2	...	2	1	1	1	2	1	1	0	0	1

10000 rows × 123 columns

Рис. 25. Наборы признаков в преобразованном датасете

Ввиду значительного объема признаков в рассматриваемом датасете и сложности его ручной обработки необходимо применение автоматизированных методов обработки данных. Для решения данной задачи целесообразно использование методов машинного обучения.

В машинном обучении задача классификации представляет собой задачу по разделению множества объектов на группы, называемые классами, на основе анализа их формального описания. При классификации каждая единица наблюдения относится к определенной группе на основе некоторого качественного свойства.

Формализуем задачу классификации.

Пусть X - множество описаний объектов, Y - конечное множество номеров (имен, меток) классов. Существует неизвестная целевая зависимость $y^*: X \to Y$, значения которой известны только на объектах конечной обучающей

выборки $X_m=(x_1,y_1),\cdots,(x_m,y_m)$. Требуется построить алгоритм $a:X\to Y,$ способный классифицировать произвольный объект $x\in X.$

В машинном обучении задача классификации часто решается с использованием обучения с учителем, так как классы определяются заранее и для примеров обучающего множества метки классов заданы. Аналитические модели, которые решают задачу классификации, называются классификаторами.

Задача классификации представляет собой одну из базовых задач машинного обучения и искусственного интеллекта в целом.

К числу распространенных методов решения задачи классификации [99] относятся:

1. Линейные классификаторы:

- LogisticRegression – логистическая регрессия (для бинарной и многоклассовой классификации);

2. Методы опорных векторов (SVM):

- Support Vector Classification (SVC) – метод опорных векторов для классификации;

3. Деревья и ансамбли деревьев:

- DecisionTreeClassifier – дерево решений;
- RandomForestClassifier – случайный лес (ансамбль деревьев);
- GradientBoostingClassifier – градиентный бустинг;

4. Байесовские методы:

- GaussianNB – наивный Байесовский классификатор для нормально распределённых признаков;

5. Методы ближайших соседей:

- KNeighborsClassifier (KNN) – k-ближайших соседей.

6. Нейронные сети:

- MLPClassifier (MLP) – многослойный перцептрон (простая нейросеть);

7. Дискриминантный анализ:

- LinearDiscriminantAnalysis (LDA) – линейный дискриминантный анализ.

Для решения задач классификации необходимо ввести метрики, по которым мы будем оценивать полученные модели.

В задачах классификации общепринятыми являются следующие определения:

TP — true positive (истино-положительное решение): классификатор верно отнёс объект к рассматриваемому классу;

TN — true negative (истино-отрицательное решение): классификатор верно утверждает, что объект не принадлежит к рассматриваемому классу.

FP — false positive (ложно-положительное решение): классификатор неверно отнёс объект к рассматриваемому классу.

FN — false negative (ложно-отрицательное решение): классификатор неверно утверждает, что объект не принадлежит к рассматриваемому классу.

Для наглядности зачастую используют матрицу ошибок (Confusion matrix) [100], приведенную на Рисунках 26 и 27.

	Принадлежит классу (P)	**Не принадлежит классу (N)**
Предсказана принадлежность классу	TP	FP
Предсказано отсутствие принадлежности к классу	FN	TN

Рис. 26. Матрица ошибок для бинарной классификации

Предсказанный класс	**Класс 1 (C_1)**	**Класс 2 (C_2)**	**Класс 3 (C_3)**
1 (P_1)	T_1	F_{12}	F_{13}
2 (P_2)	F_{21}	T_2	F_{23}
3 (P_3)	F_{31}	F_{32}	T_3

Рис. 27. Матрица ошибок для многоклассовой классификации

Рассмотрим основные метрики.

Accuracy – доля правильных классификаций. Несмотря на очевидность и простоту, является одной из самых малоинформативных оценок классификаторов. Считается по следующей формуле:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

Recall – полнота, TPR (true positive rate), показывает отношение верно классифицированных объектов класса к общему числу элементов этого класса.

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

Precision – точность, показывает долю верно классифицированных объектов среди всех объектов, которые к этому классу отнес классификатор.

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

Fall-out – FPR (false positive rate), показывает долю неверных срабатываний классификатора к общему числу объектов за пределами класса. Характеризует, насколько часто классификатор ошибается при отнесении того или иного объекта к классу.

$$FPR = \frac{FP}{FP + TN} \quad (8)$$

Для получения более взвешенных оценок качества моделей зачастую используют различные виды агрегации точности и полноты.

Арифметическое среднее:

$$\mathrm{A} = \frac{1}{2}(precision + recall) \quad (9)$$

Минимум:

$$M = min\ (precision, recall) \quad (10)$$

Гармоническое среднее, F-мера:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (11)$$

В рамках настоящего исследования для оценки качества полученных моделей будем использовать F-меру, которая гармонично учитывает как

ложноположительные, так и ложноотрицательные значения классификатора, что имеет важное значение в рамках решения задачи по обнаружению признаков компьютерных атак.

Дополнительно для оценки качества моделей может использоваться кросс-валидация – это метод оценки обобщающей способности модели, при котором исходная выборка данных разбивается на несколько частей, а обучение и тестирование проводится многократно на разных подмножествах.

Кросс-валидация используется для оценки устойчивости модели – проверки, насколько хорошо алгоритм работает на разных подвыборках. Использование данного метода снижает зависимость от «неудачного» разбиения выборки, при котором классы в обучающей и тестовой выборке не сбалансированы.

4.2 Обучение классификаторов и оценка результатов

В качестве инструмента для обучения моделей машинного обучения в рамках настоящего исследования использовался язык программирования Python и библиотека для анализа данных Scikit-learn.

Для построения моделей необходимо исходную выборку разделить на тренировочную и тестовую.

Также для целей обучения модели необходимо удалить из набора признаки, не участвующие в классификации и целевой признак *target,* значения которого вынести в отдельный массив, как показано на Рисунке 28.

```
# Разделение данных
X = df.drop(['fields.session', 'sub_category_encoded', 'target'], axis=1)
y = df['target']

# Разделение на train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

Рис. 28. Разделение выборки и выделение целевой переменной

Для разработанного датасета применим следующие алгоритмы машинного обучения:

4.2.1 Логистическая регрессия (Logistic Regression)

В библиотеке Scikit-learn алгоритм логистической регрессии реализован в классе LogisticRegression.

Основные параметры:

penalty {'l1', 'l2', 'elasticnet', None}, default='l2' – тип регуляризации;

C float, default=1.0 – параметр регуляризации;

solver {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs' – алгоритм оптимизации;

max_iter int, default=100 – максимальное количество итераций оптимизации;

multi_class {'auto', 'ovr', 'multinomial'}, default='auto' – стратегия для многоклассовой классификации;

tol float, default=1e-4 – критерий остановки при достижении заданной точности.

4.2.2 Деревья решений

В библиотеке Scikit-learn алгоритм дерева решений [101] реализован в классе *DecisionTreeClassifier*.

Указанный алгоритм содержит следующие основные параметры:

criterion {"gini", "entropy", "log_loss"}, default="gini" – функция для измерения качества разделения;

splitter {"best", "random"}, default="best" – принцип разделения для каждого узла;

max_depth int, default=None – максимальная глубина дерева;

min_samples_split int or float, default=2 – минимальное число объектов, необходимое для того, чтобы узел дерева мог разделиться;

min_samples_leaf int or float, default=1 - минимальное число объектов в листьях.

4.2.3 Метод k-ближайших соседей

В рамках исследования рассмотрим подробнее алгоритм k-ближайших соседей [102].

В библиотеке Scikit-learn алгоритм k-ближайших соседей реализован в классе *KNeighborsClassifier*.

Указанный алгоритм содержит следующие основные параметры:

n_neighbors int, default=5 - количество соседей;

weights {'uniform', 'distance'} or callable, default='uniform' – весовая функция, используемая в предсказании, может принимать следующие значения:

'uniform': однородные веса, все точки в каждой окрестности имеют одинаковый вес;

'distance': взвешивание точек обратно их расстоянию, в этом случае более близкие соседи точки запроса будут иметь большее влияние, чем соседи, которые находятся дальше;

[callable]: определяемая пользователем функция, которая принимает массив расстояний и возвращает массив, содержащий веса;

algorithm {'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto' - алгоритм, используемый для вычисления ближайших соседей.

4.2.4 Наивный Байесовский классификатор

В рамках исследования рассмотрим подробнее гауссовский наивный Байесовский алгоритм для классификации [103].

В библиотеке Scikit-learn алгоритм реализован в классе *GaussianNB*.

Указанный алгоритм содержит следующие основные параметры:

priors array-like of shape (n_classes,) – априорные вероятности классов;

var_smoothing float, default=1e-9 - доля наибольшей дисперсии всех признаков, которая добавляется к дисперсии для стабильности вычислений.

4.2.5 Метод опорных векторов

В рамках исследования рассмотрим подробнее алгоритм C-Support Vector Classification [104].

В библиотеке Scikit-learn алгоритм реализован в классе *SVC*.

Указанный алгоритм содержит следующие основные параметры:

C float, default=1.0 – параметр регуляризации;

kernel {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf' – тип ядра, используемого в алгоритме;

max_iter int, default=-1 – установка лимита итераций работы алгоритма, при значении -1 нет ограничений.

4.2.5 Случайный лес

В библиотеке Scikit-learn алгоритм реализован в классе *RandomForestClassifier* [105].

Указанный алгоритм содержит следующие основные параметры:

n_estimators int, default=100 - число деревьев в «лесу»;

criterion {"gini", "entropy", "log_loss"}, default="gini" – функция оценки качества разделения;

max_depth int, default=None - максимальная глубина дерева;

min_samples_split int or float, default=2- минимальное число объектов, необходимое для того, чтобы узел дерева мог разделиться;

min_samples_leaf int or float, default=1 - минимальное число объектов в листьях;

max_features {"sqrt", "log2", None}, int or float, default="sqrt" – набор признаков для наилучшего ветвления;

bootstrap bool, default=True - использование для построения деревьев подвыборки с возвращением.

4.2.6 Градиентный бустинг (Gradient Boosting)

В библиотеке Scikit-learn алгоритм градиентного бустинга реализован в классе GradientBoostingClassifier.

Основные параметры:

loss {'log_loss', 'exponential'}, default='log_loss' – функция потерь для оптимизации;

learning_rate float, default=0.1 – коэффициент обучения, уменьшающий вклад каждого дерева;

n_estimators int, default=100 – количество деревьев (итераций бустинга);

criterion {'friedman_mse', 'squared_error'}, default='friedman_mse' – критерий для измерения качества разбиения;

max_depth int, default=3 – максимальная глубина каждого дерева;

min_samples_split int or float, default=2 – минимальное количество объектов для разделения узла;

min_samples_leaf int or float, default=1 – минимальное количество объектов в листе;

max_features {'sqrt', 'log2', None}, int or float, default=None – количество признаков для поиска лучшего разбиения;

subsample float, default=1.0 – доля выборки, используемая для обучения каждого дерева.

4.2.7 Многослойный перцептрон (MLP)

В библиотеке Scikit-learn алгоритм многослойного перцептрона реализован в классе MLPClassifier.

Основные параметры:

hidden_layer_sizes tuple, default=(100,) – количество нейронов в каждом скрытом слое (например, (50, 30) означает два слоя с 50 и 30 нейронами);

activation {'identity', 'logistic', 'tanh', 'relu'}, default='relu' – функция активации скрытых слоев;

solver {‘lbfgs’, ‘sgd’, ‘adam’}, default=’adam’ – алгоритм оптимизации;

alpha float, default=0.0001 – параметр L2-регуляризации;

batch_size int, default=’auto’ – размер мини-батча для стохастического градиентного спуска;

learning_rate {‘constant’, ‘invscaling’, ‘adaptive’}, default=’constant’ – стратегия изменения скорости обучения;

max_iter int, default=200 – максимальное количество эпох обучения;

early_stopping bool, default=False – остановка обучения при ухудшении качества на валидационной выборке;

tol float, default=1e-4 – порог остановки при изменении функции потерь.

4.2.8 Линейный дискриминантный анализ (LDA)

В библиотеке Scikit-learn алгоритм линейного дискриминантного анализа реализован в классе LinearDiscriminantAnalysis.

Основные параметры:

solver {‘svd’, ‘lsqr’, ‘eigen’}, default=’svd’ – метод решения задачи LDA;

shrinkage {‘auto’, float} or None, default=None – параметр регуляризации (используется при solver='lsqr' или 'eigen');

priors array-like, default=None – априорные вероятности классов;

n*_components int or None, default=None* – количество компонент для уменьшения размерности;

tol float, default=1e-4 – порог остановки при выполнении сингулярного разложения матриц.

Для обучения классификаторов создадим словарь с параметрами по умолчанию, как показано на Рисунке 29. Оценку качества классификации проведем отдельно для тренировочной и тестовой выборки. Дополнительно для F-меры, как основной метрики, проведем кросс-валидацию, чтобы получить максимально достоверный результат, как это показано на Рисунке 30.

```python
# Список классификаторов
classifiers = {
    "Logistic Regression": LogisticRegression(),
    "SVC": SVC(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Gradient Boosting": GradientBoostingClassifier(),
    "Gaussian NB": GaussianNB(),
    "KNN": KNeighborsClassifier(),
    "MLP": MLPClassifier(),
    "LDA": LinearDiscriminantAnalysis(),
}
```

Рис. 29. Список классификаторов для обучения

```python
# Метрики для train и test
metrics = {
    'Train Accuracy': accuracy_score(y_train, y_train_pred),
    'Test Accuracy': accuracy_score(y_test, y_test_pred),
    'Train Precision': precision_score(y_train, y_train_pred, average='weighted'),
    'Test Precision': precision_score(y_test, y_test_pred, average='weighted'),
    'Train Recall': recall_score(y_train, y_train_pred, average='weighted'),
    'Test Recall': recall_score(y_test, y_test_pred, average='weighted'),
    'Train F1': f1_score(y_train, y_train_pred, average='weighted'),
    'Test F1': f1_score(y_test, y_test_pred, average='weighted'),
}

# Кросс-валидация (F1-score)
cv_scores = cross_val_score(model, X_train, y_train, cv=StratifiedKFold(cv), scoring='f1_weighted')
metrics['CV F1 Train'] = np.mean(cv_scores)
```

Рис. 30. Список метрик и кросс-валидация для F-меры

Результаты обучения классификаторов приведены на Рисунках 31, 32.

Model	Train Accuracy	Test Accuracy	Train Precision	Test Precision	Train Recall	Test Recall	Train F1	Test F1
Random Forest	1.0000	0.9965	1.0000	0.9965	1.0000	0.9965	1.0000	0.9965
Gradient Boosting	0.9990	0.9960	0.9990	0.9960	0.9990	0.9960	0.9990	0.9960
MLP	0.9995	0.9950	0.9995	0.9950	0.9995	0.9950	0.9995	0.9950
Decision Tree	1.0000	0.9935	1.0000	0.9935	1.0000	0.9935	1.0000	0.9935
Logistic Regression	0.9975	0.9925	0.9975	0.9925	0.9975	0.9925	0.9975	0.9925
LDA	0.9940	0.9900	0.9940	0.9900	0.9940	0.9900	0.9940	0.9900
KNN	0.9945	0.9880	0.9945	0.9881	0.9945	0.9880	0.9945	0.9880
SVC	0.9710	0.9680	0.9712	0.9682	0.9710	0.9680	0.9710	0.9680
Gaussian NB	0.9260	0.9195	0.9288	0.9217	0.9260	0.9195	0.9260	0.9193

Рис. 31. Результаты обучения классификаторов

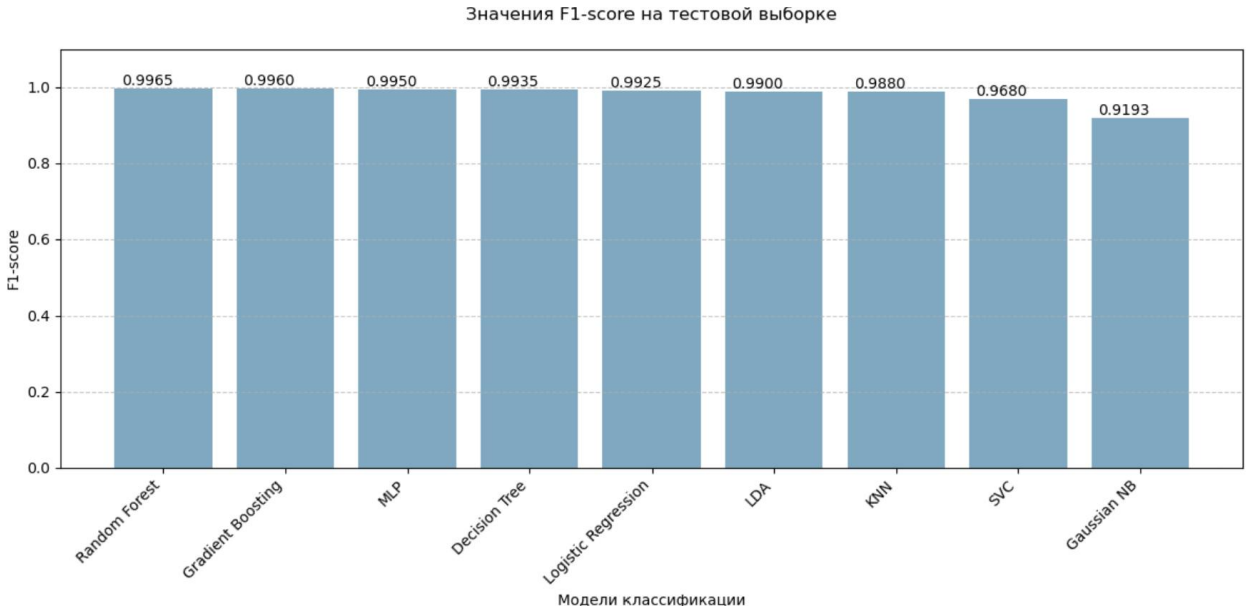

Рис. 32. Результаты обучения классификаторов

Как видно из представленных графиков, лучшие результаты из рассмотренных алгоритмов на заданном наборе данных показал алгоритм классификации «случайный лес» (Random Forest).

Помимо высоких показателей эффективности классификации, данный алгоритм обладает следующими важными преимуществами:

1. Устойчивость к переобучению.

«Случайный лес» является ансамблевым методом, который объединяет множество деревьев решений, что позволяет достичь высокой точности классификации [106]. Он устойчив к переобучению, что особенно важно при работе с зашумленными данными, такими как системные журналы. Также в системных журналах часто присутствуют пропуски данных и аномалии, которые могут негативно повлиять на качество модели. «Случайный лес» эффективно справляется с такими проблемами благодаря агрегированию результатов множества деревьев.

2. Устойчивость к дисбалансу классов

В задачах обнаружения атак данные часто несбалансированы (обычно легитимных действий гораздо больше, чем атак). «Случайный лес» позволяет

учитывать дисбаланс классов с помощью настройки весов или использования методов балансировки [107].

3. Интерпретируемость результатов

В отличие от нейронных сетей, «случайный лес» позволяет оценить важность признаков, что полезно для анализа и понимания, какие события в журналах наиболее значимы для обнаружения атак [108]. Для специалистов по кибербезопасности важно понимать, какие именно события (например, конкретные идентификаторы событий в журнале Security) являются индикаторами атаки. «Случайный лес» предоставляет возможность ранжирования признаков по их важности.

4. Эффективность на больших объемах данных

«Случайный лес» может эффективно обрабатывать большие объемы данных, что важно при работе с системными журналами, которые могут содержать миллионы записей. События Windows генерируют огромное количество данных, особенно в крупных организациях.

Таким образом, метод машинного обучения «случайный лес» является подходящим алгоритмом для обнаружения компьютерных атак с использованием анализа системных журналов операционной системы благодаря своей высокой точности, устойчивости к переобучению, способности работать с разнородными и зашумленными данными, а также простоте настройки и интерпретации результатов.

4.3 Интерпретация результатов и программная реализация

Для оценки результатов работы разработанной модели составим и проанализируем матрицу ошибок, приведенную на Рисунке 33.

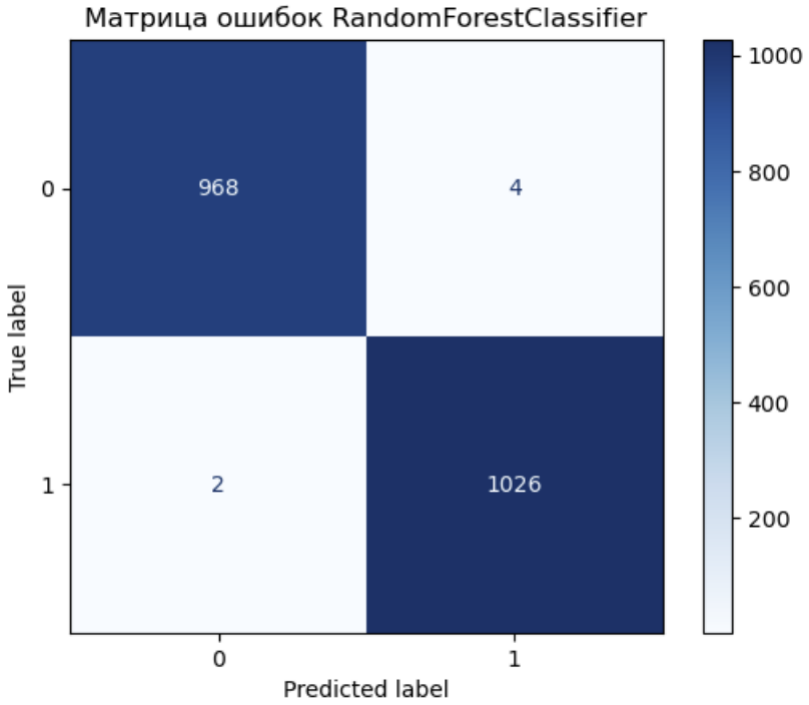

Рис. 33. Матрица ошибок для разработанной модели

Всего рассматривалось 2000 объектов. Как видно из указанной матрицы, классификатор верно отнес к классу 1 (компьютерные атаки) 1026 объектов, к классу 0 (легитимные действия пользователей) 968 объектов. Классификатор ошибочно отнес к классу компьютерных атак 4 легитимных действия пользователей, а также 2 кибератаки классифицировал как легитимное действие.

Рассмотрим подробнее неверно классифицированные объекты. Для этого необходимо установить значения fields.session и по ним найти сессии в первоначальном датасете. Дополнительно с помощью метода predict_proba получим значения вероятности, с которой классификатор дал оценку.

Сессии с FN ошибками:

- 20250422145459;
- 20250503203858.

Сессии с FP ошибками:

- 20250426145009;
- 20250413151533;

- 20250413200545;

- 20250413181838.

Аналитика по ложноотрицательным предсказаниям приведена в Таблице 10, по ложноположительным – в Таблице 11.

Таблица 10. Результаты неверной работы классификатора (FN)

Сессия	Вид действия	Вероятность
20250422145459	['SYSTEM_BINARY_PROXY_EXECUTION_(T1218)']	0.4000
20250503203858	['SYSTEM_BINARY_PROXY_EXECUTION_(T1218)']	0.2900

Таблица 11. Результаты неверной работы классификатора (FP)

Сессия	Вид действия	Вероятность
20250426145009	['WEB']	0.6700
20250413151533	['OFFICE']	0.5600
20250413200545	['WEB']	0.9800
20250413181838	['WEB']	0.7700

Как видно из приведенных таблиц и рисунков, ложноотрицательные предсказания с вероятностью 40% и 29% относятся к технике T1218, в рамках которой эксплуатируются легитимные исполняемые файлы, что затрудняет выявление такого рода компьютерных атак.

Ложноположительные предсказания касаются двух видов пользовательской активности – офисная работа (вероятность 56%) и работа с веб-приложениями (вероятности 67%, 98% и 77%). Ошибки классификатора могут быть связаны с загрузкой файлов и запуском новых процессов в ходе работы в Интернет-браузере.

Важным свойством алгоритма классификации является вычисление весов признаков [109], благодаря которым происходит дальнейшее отнесение объектов к тому или иному классу.

В рамках исследования признаки соответствуют идентификаторам событий операционной системы, что имеет важное значение для аналитиков в области информационной безопасности, поскольку каждое событие документировано и позволяет выявлять дополнительные характерные поведенческие признаки, присущие компьютерной атаке в ходе ее реализации.

10 самых важных с точки зрения классификатора признаков, полученных в рамках исследования, приведены на Рисунках 34, 35.

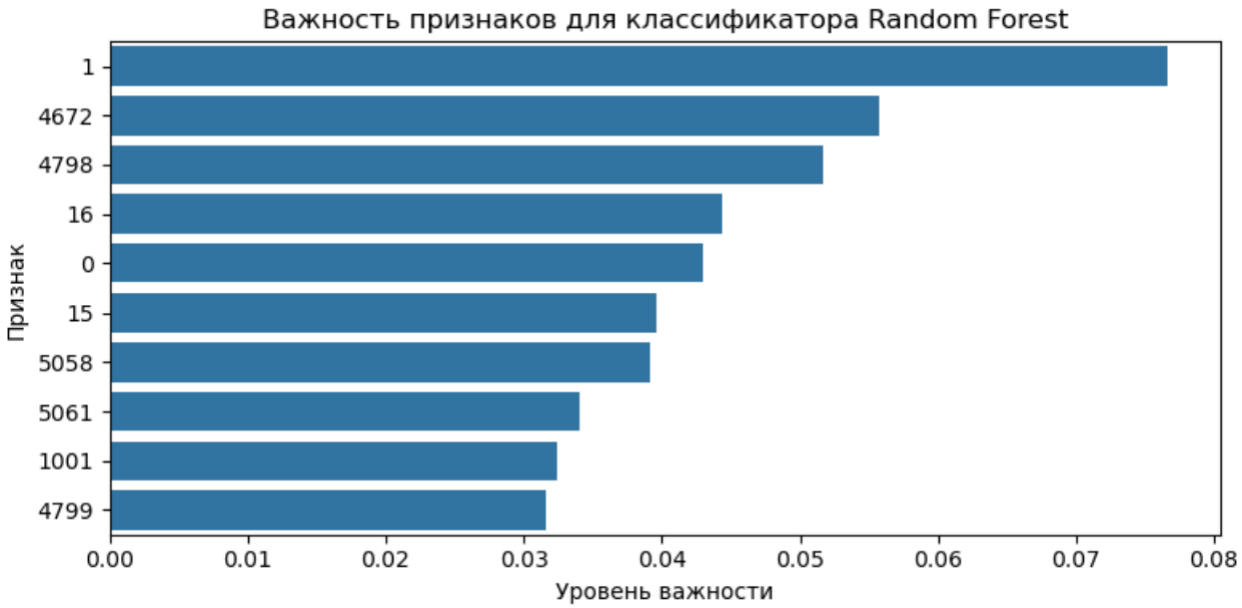

Рис. 34. Важность признаков при классификации

Feature	1	4672	4798	16	0	15	5058	5061	1001	4799
Importance	0.076633	0.055712	0.051627	0.044346	0.042954	0.03964	0.039084	0.034054	0.032464	0.031585

Рис. 35. Важность признаков при классификации

Итог работы классификатора составил 0,9965, что является высоким результатам, который можно использовать в практической деятельности по выявлению признаков компьютерных атак.

Классификаторы, обученные на сетевом трафике, приведенные в Таблице 1, демонстрируют максимальный результат 0,9279. Улучшение

составляет более 6 %. Сигнатурное выявление компьютерных атак по системным журналам дает результат 0,9165. Улучшение составляет 8 %.

На базе обученной модели машинного обучения разработано консольное приложение evtx_ml_analyzer, которое позволяет анализировать файлы системных событий операционной системы Windows, а также выводит информацию о наличии признаков компьютерных атак. Общий вид интерфейса приложения приведен на Рисунке 36.

Данный программный продукт прошел апробацию специалистами центра ГосСОПКА на базе ООО «Информационный центр» и используется в качестве одного из инструмента при расследовании компьютерных инцидентов. Получена справка о внедрении.

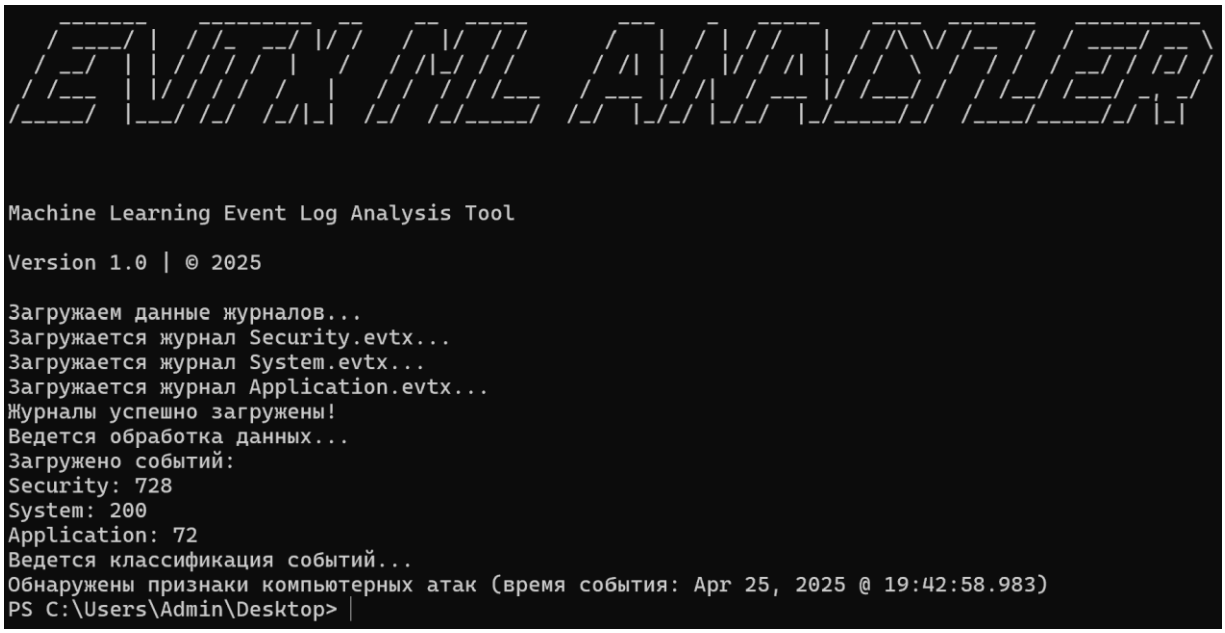

Рис. 36. Интерфейс программы evtx_ml_analyzer

Проведен сравнительный анализ разработанного приложения с имеющимися аналогами.

Сравнение проводилось с инструментами Python_Evtx_Analyzer[24], evtx_detector[25]. Интерфейсы приложений приведены на Рисунках 37 и 38.

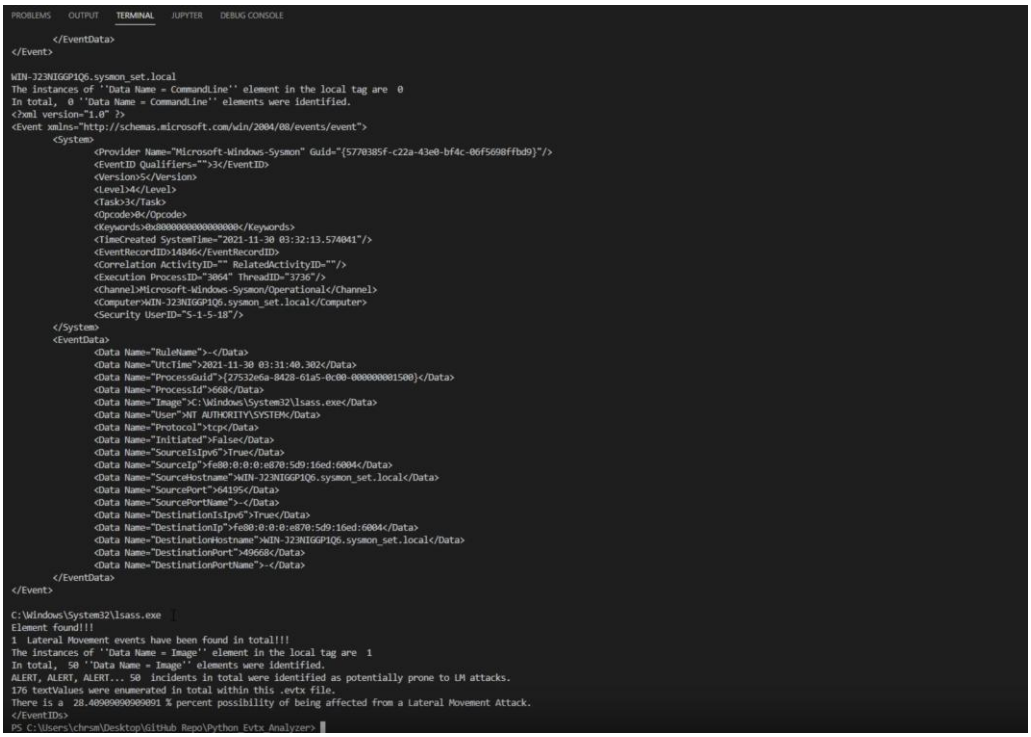

Рис. 37. Интерфейс программы Python_Evtx_Analyzer

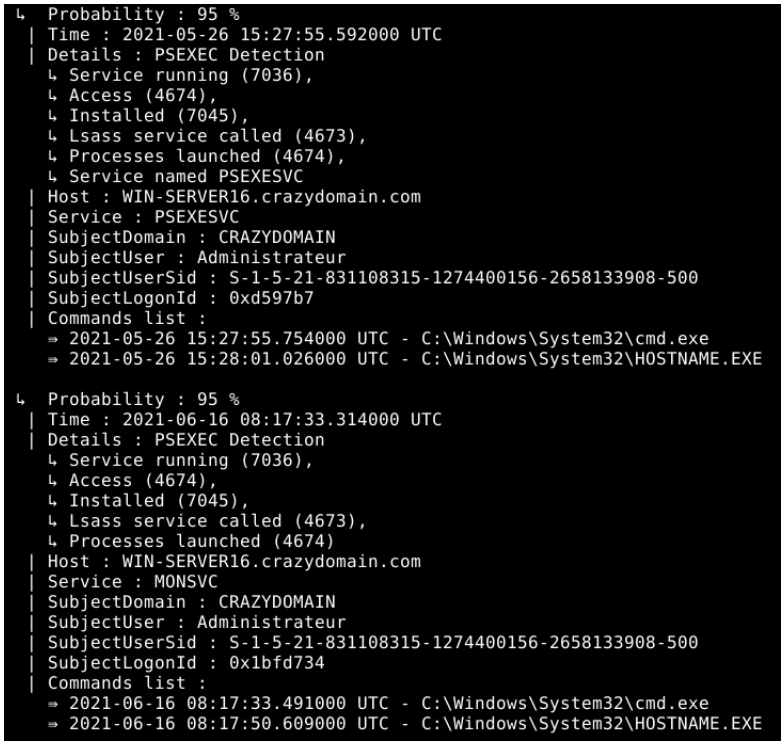

Рис. 38. Интерфейс программы evtx_detector

[24] Описание инструмента по ссылке https://github.com/ChristosSmiliotopoulos/Python_Evtx_Analyzer
[25] Описание инструмента по ссылке https://gitlab.com/Juquod/evtx_detector

Работа указанных программ проверена с помощью репозитория журналов операционной системы EVTX-ATTACK-SAMPLES[26], а также иные наборы журналов, находящиеся в открытом доступе.

Всего проверено 100 различных журналов. Сравнительная характеристика представленных приложений, а также результаты их работы по выявлению атак приведены в Таблице 12, а также на Рисунках 39-42.

Таблица 12. Сравнительный анализ приложений

Название	Интер-фейс	Метод обнаружения атак	Детализа-ция	Пакетная загрузка	Неизвест-ные атаки	Простота интегра-ции	Количество выявленных атак (из 100)
evtx_ml_analyzer	CLI	машинное обучение	Низкая	Да	Да	Средняя	96
Python_Evtx_Analyzer	CLI	сигнатурный	Высокая	Да	Нет	Средняя	82
evtx_detector	CLI	сигнатурный	Средняя	Нет	Нет	Высокая	89

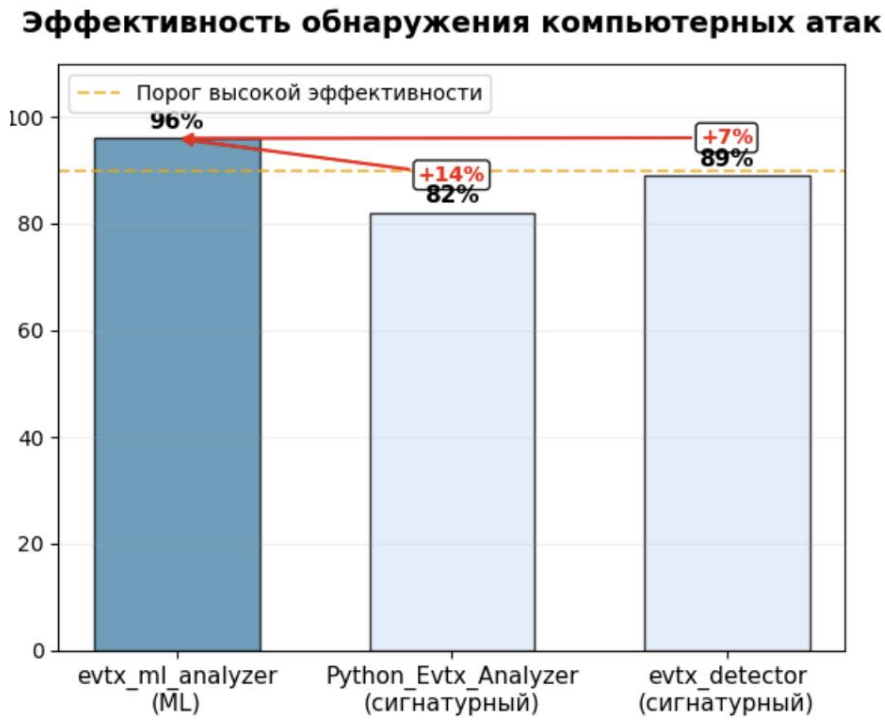

Рис. 39 Сравнительный анализ (эффективность обнаружения)

[26] Репозиторий доступен по адресу https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES

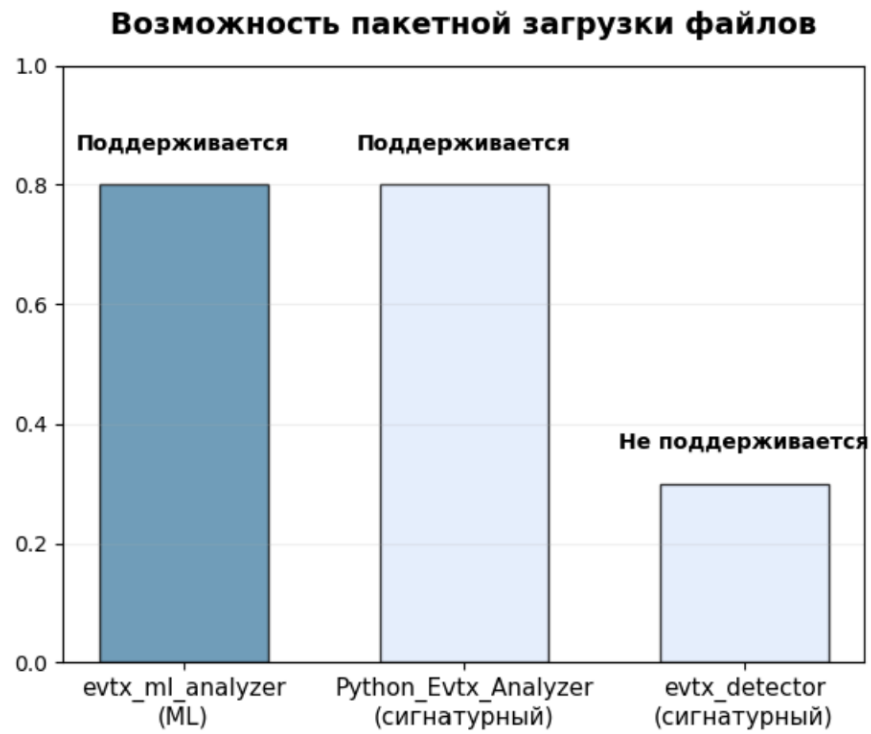

Рис. 40 Сравнительный анализ (пакетная загрузка)

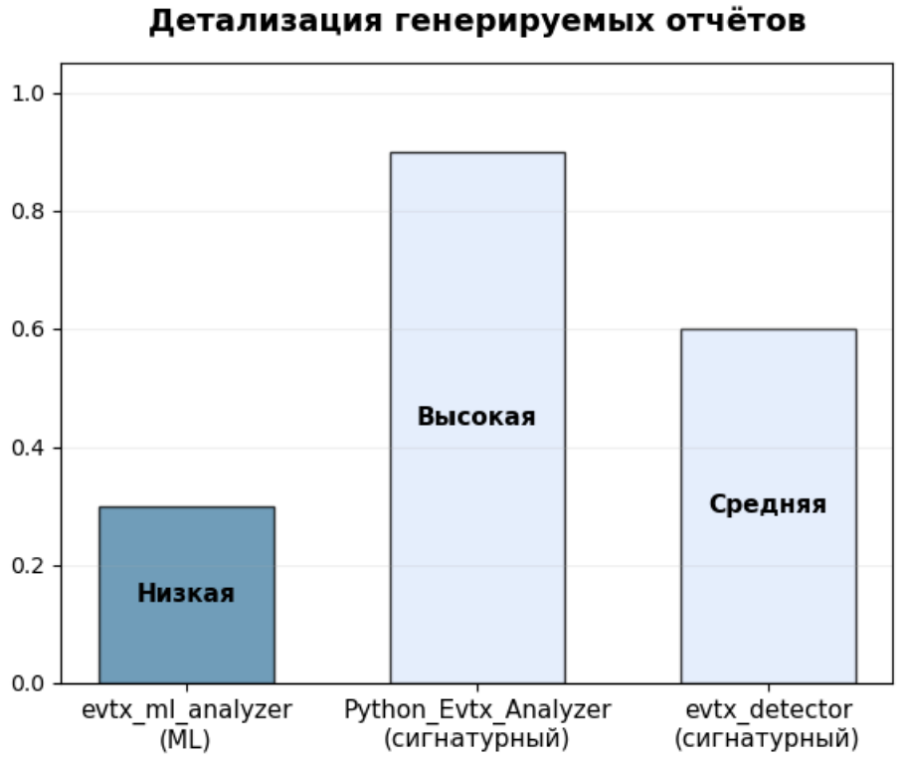

Рис. 41 Сравнительный анализ (детализация отчетов)

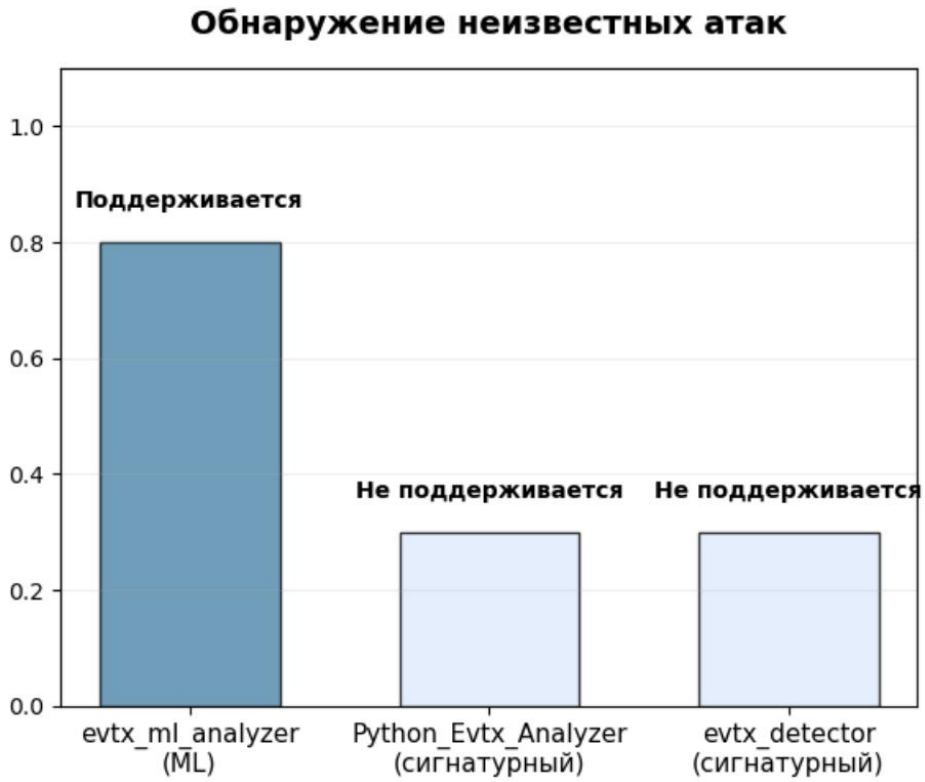

Рис. 42 Сравнительный анализ (неизвестные атаки)

Таким образом, разработанное в рамках исследования приложение демонстрирует лучшие результаты выявления компьютерных атак по журналам операционной системы, успешно выявив 96 из 100 атак.

Оценка достоверности полученных результатов

Достоверность результатов, полученных в ходе исследования и разработки модели машинного обучения для выявления компьютерных атак, подтверждается комплексным подходом к валидации, включающим следующие аспекты:

1. Статистическая значимость и высокая точность модели.

Результаты оценивались на репрезентативной выборке, содержащей 2000 объектов. Матрица ошибок (Рисунок 33) демонстрирует высокие показатели качества классификации:

Точность модели составила 0,9965;

Количество ложных срабатываний (FP) – 4 (0,2% от общего числа);

Количество пропусков атак (FN) – 2 (0,1% от общего числа).

Данные показатели значительно (на 6-8%) превосходят точность рассмотренных в обзоре литературы аналогов, основанных на анализе сетевого трафика (0,9279) и сигнатурном анализе журналов (0,9165), как приведено на Рисунке 43, что свидетельствует о практической эффективности предложенного подхода.

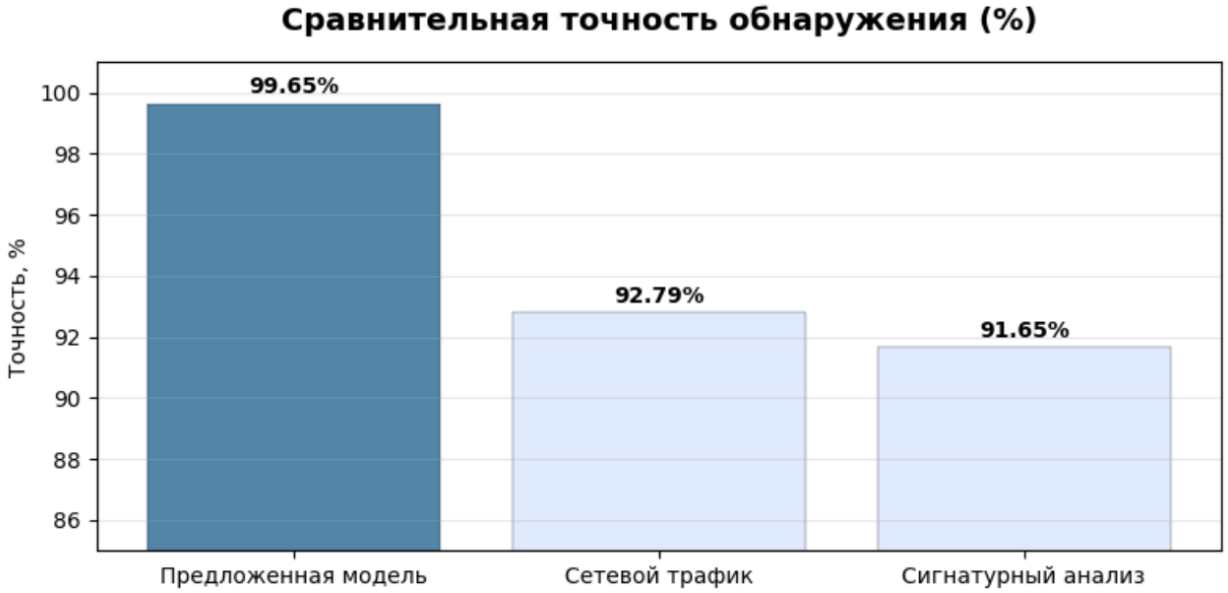

Рисунок 43. Сравнительный анализ подходов

2. Интерпретируемость и обоснованность ошибок модели.

Проведен детальный анализ ошибочных предсказаний (FN и FP). Для каждой ошибки идентифицирована конкретная сессия, определен вид активности и получена предсказанная моделью вероятность.

Установлено, что ложноотрицательные предсказания (пропущенные атаки) с низкой уверенностью модели (29% и 40%) относятся к сложно обнаруживаемой технике T1218 (System Binary Proxy Execution), использующей легитимные исполняемые файлы, как приведено на Рисунке 44. Это объясняет причину ошибок и согласуется с реальными сложностями в области обнаружения угроз.

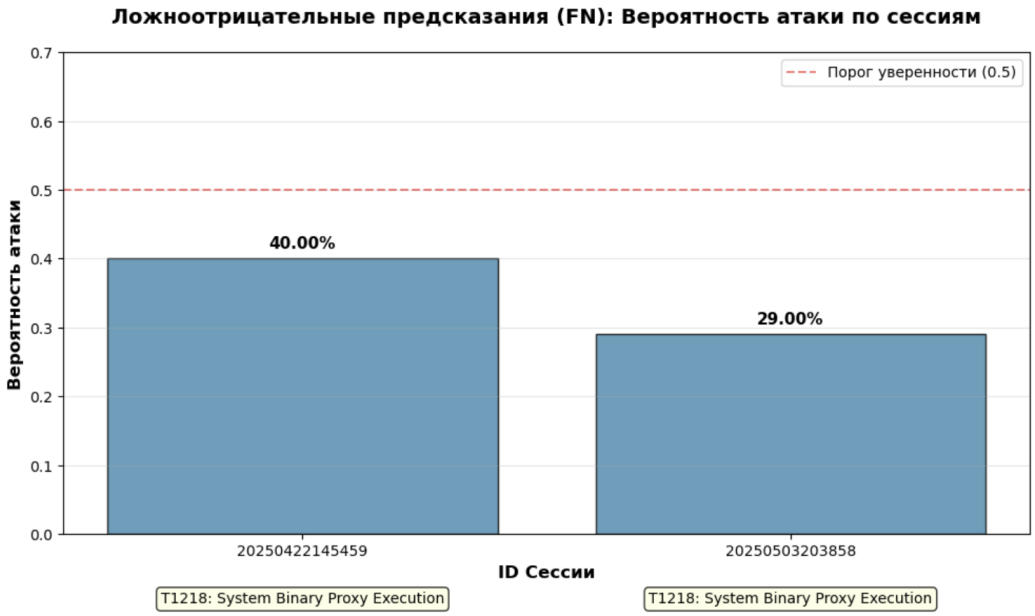

Рис. 44 Результаты неверной работы классификатора (FN)

Ложноположительные срабатывания связаны с активностью пользователей (WEB, OFFICE), имитирующей подозрительное поведение (загрузка и запуск файлов), что также является известным вызовом для систем классификации. Визуальное представление приведено на Рисунке 45.

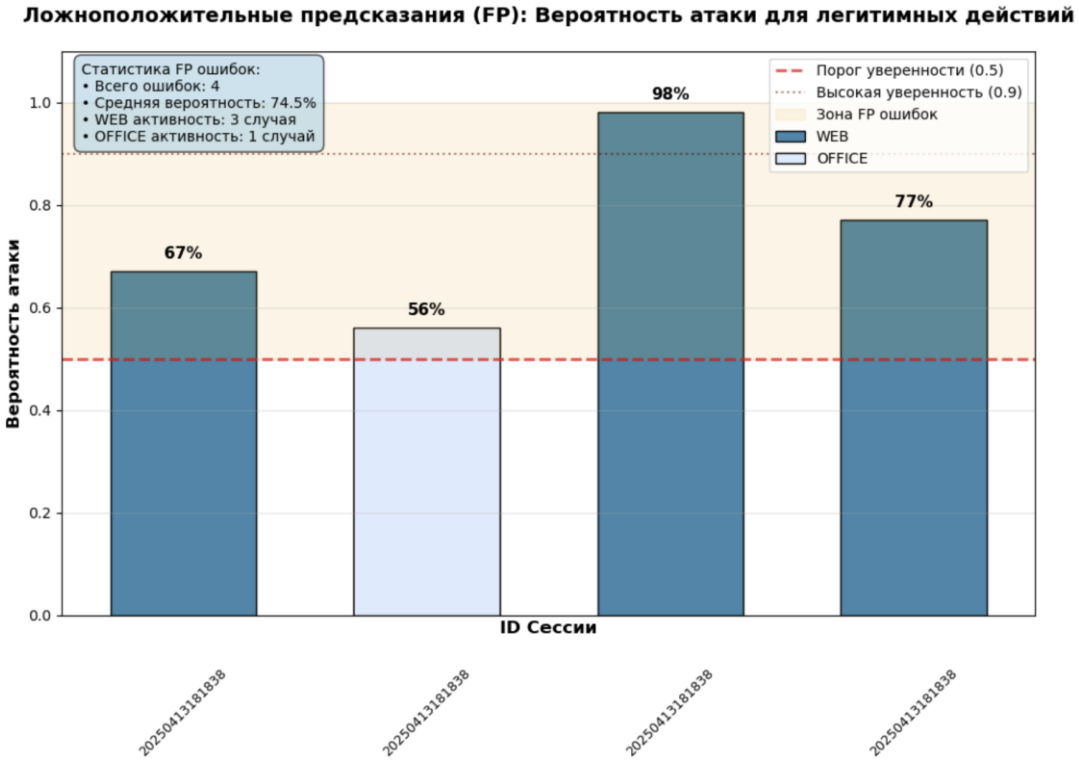

Рис. 45 Результаты неверной работы классификатора (FP)

Анализ подтверждает, что модель совершает ошибки в ожидаемых и объяснимых с практической точки зрения сценариях, а не демонстрирует хаотичное поведение.

3. Объективность валидации на независимых данных.

Для сравнительного анализа разработанного приложения evtx_ml_analyzer использовался публичный набор данных EVTX-ATTACK-SAMPLES, содержащий журналы с известными атаками, а также дополнительные журналы из открытых источников.

Тестирование проводилось на 100 независимых файлах журналов, что обеспечивает объективность оценки.

Результат в 96 корректно выявленных атак из 100 подтверждает высокую эффективность модели на новых, неизвестных в процессе обучения данных. Сравнение работы приложений приведено на Рисунке 46.

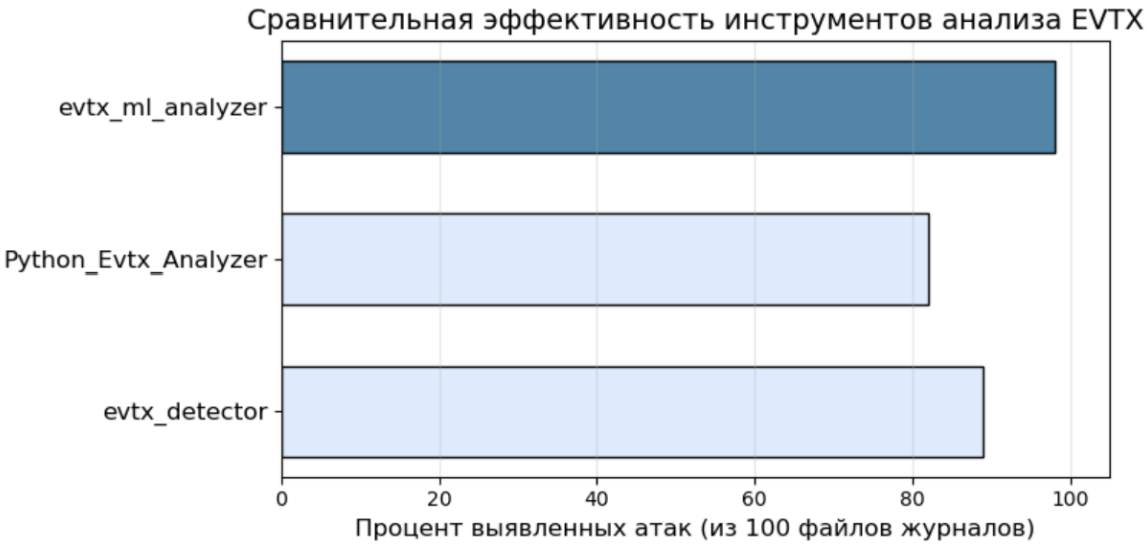

Рис. 46. Сравнительный анализ работы приложений

4. Практическое внедрение и экспертиза.

Разработанный на базе модели программный продукт прошел апробацию и внедрен в практическую деятельность специалистами центра ГосСОПКА ООО «Информационный центр».

Факт внедрения подтвержден официальной справкой.

Использование приложения в качестве инструмента при расследовании инцидентов подтверждает практическую значимость и достоверность полученных научно-технических результатов.

5. Сравнение с прямыми аналогами.

Проведено прямое сравнительное тестирование с существующими инструментами (Python_Evtx_Analyzer, evtx_detector), работающими в той же предметной области (анализ EVTX-журналов).

Превышение показателя обнаружения на 7–14% при равных условиях тестирования является статистически значимым и доказывает преимущество подхода на основе машинного обучения в данной задаче. Разработанное программное средство имеет и другие преимущества над рассматриваемыми аналогами, как это приведено на Рисунке 47.

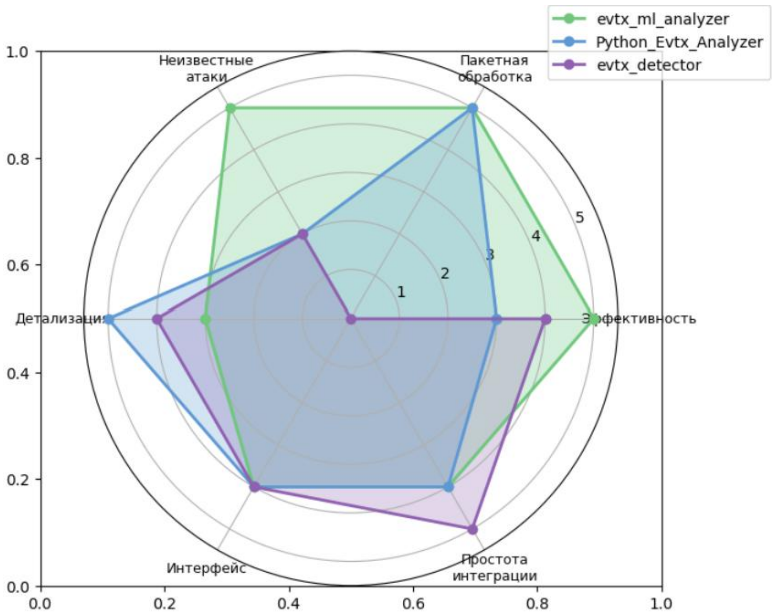

Рис. 47. Преимущества разработанного приложения

Высокие метрики качества, объяснимость ошибок, успешная валидация на независимом наборе данных, положительные результаты сравнительного анализа и факт внедрения в экспертную организацию в совокупности обеспечивают высокую степень достоверности полученных результатов.

4.4 Выводы по главе

В заключительной главе исследования проведена апробация разработанной методики выявления признаков проведения компьютерных атак с помощью анализа системных событий операционной системы Windows с применением алгоритмов машинного обучения.

Согласно разработанной формальной модели подготовлен датасет, включающий наборы признаков действий, соответствующих идентификаторам событий операционной системы. В качестве целевого признака введен тип действия: компьютерная атака или легитимная работа пользователя.

К полученному датасету последовательно применены алгоритмы машинного обучения: деревья решений, К-ближайших соседей, наивный Байесовский классификатор, опорных векторов, случайный лес, линейный дискриминантный анализ, многослойный перцептрон, логистическая регрессия.

Проведен сравнительный анализ полученных классификаторов согласно установленным метрикам и интерпретация полученных результатов. Итоговый результат работы разработанного классификатора составил 0,9965.

На базе обученной модели разработан программный продукт evtx_ml_analyzer, который прошел успешную апробацию и внедрен в деятельность центра ГосСОПКА на базе ООО «Информационный центр».

Разработанное приложение прошло сравнение с имеющимися аналогами и продемонстрировало лучшие результаты выявления компьютерных атак по журналам операционной системы, успешно выявив 96 из 100 атак.

Приведенные факты в совокупности свидетельствуют о высокой достоверности полученных результатов.

Заключение

В ходе диссертационного исследования в соответствующих главах были достигнуты следующие результаты.

Результатами работы по исследованию существующих видов компьютерных атак и методов их обнаружения стала авторская методика выявления компьютерных атак с использованием алгоритмов машинного обучения при обработке системных событий операционной системы, основанная на требованиях стандартов Российской Федерации ГОСТ серии 59709-59712.

Анализ структуры и содержания журналов операционной системы Microsoft Windows позволил разработать формальную модель компьютерных атак и легитимных действий пользователей, которая позволяет выявлять компьютерные атаки с использованием системных событий операционной системы. Предложенная модель может быть переиспользована для отечественных операционных систем в рамках политики импортозамещения.

В результате изучения имеющихся подходов и лучших практик, удалось разработан алгоритм моделирования компьютерных атак и формирования набора данных из системных событий операционной системы, в ходе выполнения которого расширен перечень выявляемых техник согласно матрице MITRE ATT&CK.

На базе предложенного алгоритма сформирован набор признаков в соответствии с идентификаторами системных событий операционной системы, который является характерным для каждого исследуемого действия. Данный подход может использоваться для решения задачи по обнаружению компьютерных атак, в том числе ранее не известных.

В рамках проведения эксперимента на специально разработанной виртуальной сетевой инфраструктуре проведено исследование системных

событий операционной системы, на которой запускались различные скрипты, эмулирующие компьютерные атаки и легитимные действия пользователей. В результате согласно разработанной модели собран датасет, содержащий наборы признаков, характерные для работы различных видов кибератак и действий пользователей, зафиксированных в операционной системе.

К подготовленному датасету применены различные методы машинного обучения, проведен их сравнительный анализ. Установлено, что наилучшие показатели по заданной метрике продемонстрировал алгоритм «случайный лес». В ходе исследования получен классификатор, который позволяет с точностью 0,9965 относить объект с заданным набором признаков (соответствующих наличию определенных записей системных событий операционной системы) либо к компьютерным атакам, либо к легитимным пользовательским сценариям. Данный результат более чем на 6 % превосходит точность работы классификаторов, обученных на сетевом трафике, а также на 8% выше результатов сигнатурного анализа.

На основания обученной модели разработано консольное программное средство evtx_ml_analyzer.py, которое позволяет анализировать файлы системных событий операционной системы, а также выводит информацию о наличии признаков компьютерных атак. Проведен сравнительный анализ приложения с известными аналогами. Приложение продемонстрировало максимальный результат, успешно выявив 96 из 100 компьютерных атак.

Предложенная методика и программный продукт прошли оценку специалистами центра ГосСОПКА на базе ООО «Информационный центр» и внедрены при расследовании компьютерных инцидентов. Получена справка о внедрении.

Внедрение разработанных в работе подходов по анализу данных в учебный процесс департамента информационной безопасности ИМКТ ДВФУ в рамках курса «Интеллектуальные информационные системы», позволяет

студентам ознакомиться с различными методами машинного обучения для решения задач в области кибербезопасности.

Поставленные в рамках работы задачи выполнены, а заявленная цель диссертационного исследования достигнута.

Дальнейшие исследования по теме диссертации направлены на разработку многоклассового классификатора, способного определить конкретный вид атаки или пользовательского действия, а также проведение дополнительных экспериментов по применению предложенной методики к отечественным операционным системам в рамках политики импортозамещения.

Список используемой литературы

1. ИИ-хакеры, каскадные сбои и дипфейки: ключевые киберугрозы 2025 года и прогнозы на 2026-й. URL: https://www.securitylab.ru /blog/company/pt/357216.php (дата обращения – 20.12.2025).

2. Итоги кода: как изменились кибератаки в России за 2025 год. URL: https://iz.ru/2005200/dmitrii-bulgakov/itogi-koda-kak-izmenilis-kiberataki-v-rossii -za-2025-god (дата обращения – 30.12.2025).

3. Кибербезопасность в 2025 году: итоги рынка, атаки, ИИ и прогнозы на 2026. URL: https://www.anti-malware.ru/analytics/Technology_Analysis/ Cybersecurity-in-2025-AMLive (дата обращения – 10.01.2026).

4. Тренды атак в 2026 году. URL: https://ptsecurity.com/ research/analytics/trendy-atak-v-2026-godu (дата обращения: 30.01.2026).

5. Solar 4RAYS: Хроники расследований инцидентов в 2025 году. URL: https://rt-solar.ru/solar-4rays/blog/6439/ (дата обращения – 15.03.2026).

6. Кого и как атаковали в 2025 году. URL: https://ib-bank.ru/bisjournal/post/2673 (дата обращения – 07.04.2026).

7. List of security hacking incidents. URL: https://en.wikipedia.org/ wiki/List_of_security_hacking_incidents (дата обращения – 15.03.2026).

8. Lehtinen, R.; Gangemi, G.T., Sr. Computer Security Basics // Computer Security. O'Reilly Media, Inc. Sebastopol, NY, USA. 2006. 429 P.

9. Manky D. Cybercrime as a service: a very modern business // Computer Fraud & Security. 2013. Vol. 6. P. 9-13.

10. Karri, R.; Rajendran, J.; Rosenfeld, K.; Tehranipoor, M. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. // Computer. 2010. Vol. 43. P. 39–46.

11. Tehranipoor M., Wang C. (ed.). Introduction to hardware security and trust. // Springer Science & Business Media. 2011. 426 P.

12. Aslan, Ö. How to decrease cyber threats by reducing software vulnerabilities and bugs. // In Proceedings of the 1st International Mediterranean

Science and Engineering Congress. Çukurova University, Adana, Turkey. 26–28 October 2016. P. 639–646.

13. Aslan Ö., Samet R. Mitigating cyber security attacks by being aware of vulnerabilities and bugs // 2017 international conference on cyberworlds (cw). IEEE. 2017. P. 222-225.

14. Thakur K., Pathan A. S. K. Cybersecurity fundamentals: A real-world perspective. // CRC Press. 2020. 304 P.

15. Krombholz K. et al. Advanced social engineering attacks // Journal of Information Security and applications. 2015. Vol. 22. P. 113-122.

16. Al-Khurafi O. B., Al-Ahmad M. A. Survey of web application vulnerability attacks // 2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT). IEEE. 2015. P. 154-158.

17. Pavlenko, A.; Buzdalov, M.; Ulyantsev, V. Fitness comparison by statistical testing in construction of SAT-based guess-and-determine cryptographic attacks. // In Proceedings of the Genetic and Evolutionary Computation Conference. Prague, Czech Republic. 13–17 July 2019. P. 312–320.

18. Vimal S., Kalaivani L., Kaliappan M. Collaborative approach on mitigating spectrum sensing data hijack attack and dynamic spectrum allocation based on CASG modeling in wireless cognitive radio networks // Cluster Computing. 2019. Vol. 22. P. 10491-10501.

19. Pawar M. V., Anuradha J. Network security and types of attacks in network // Procedia Computer Science. 2015. Vol. 48. P. 503-506.

20. Basit, A.; Zafar, M.; Liu, X.; Javed, A.R.; Jalil, Z.; Kifayat, K. A comprehensive survey of AI-enabled phishing attacks detection techniques // Telecommun. Syst. 2020. Vol. 76. P. 139–154.

21. Kramer S., Bradfield J. C. A general definition of malware // Journal in computer virology. 2010. Vol. 6. P. 105-114.

22. Silva S. S. C. et al. Botnets: A survey // Computer Networks. 2013. Vol. 57. No. 2. P. 378-403.

23. Raza M. et al. A survey of password attacks and comparative analysis on methods for secure authentication // World applied sciences journal. 2012. Vol. 19. No. 4. P. 439-444.

24. Conti M., Dragoni N., Lesyk V. A survey of man in the middle attacks // IEEE communications surveys & tutorials. 2016. Vol. 18. No. 3. – P. 2027-2051.

25. Израилов К. Е., Буйневич М. В. Метод обнаружения атак различного генеза на сложные объекты на основе информации состояния. Часть 1. Предпосылки и схема // Вопросы кибербезопасности. 2023. №. 3 (55). С. 90-100. DOI:10.21681/2311-3456-2023-3-90-100.

26. Израилов К. Е., Буйневич М. В. Метод обнаружения атак различного генеза на сложные объекты на основе информации состояния. Часть 2. Алгоритм, модель и эксперимент // Вопросы кибербезопасности. 2023. №. 4 (56). С. 80-93. DOI: 10.21681/2311-3456-2023-4-80-93.

27. Nehinbe J. O. Log Analyzer for Network Forensics and Incident Reporting // 2010 International Conference on Intelligent Systems, Modelling and Simulation. IEEE. 2010. P. 356-361.

28. Dong H., Kotenko I. Cybersecurity in the AI era: analyzing the impact of machine learning on intrusion detection // Knowledge and Information Systems. 2025. Vol. 67. No. 5. P. 3915-3966.

29. Lippmann R. et al. The 1999 DARPA off-line intrusion detection evaluation // Computer networks. 2000. Vol. 34. No. 4. P. 579-595.

30. Lippmann R. P. et al. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation // Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00. IEEE. 2000. Vol. 2. P. 12-26.

31. Brown C. et al. Analysis of the 1999 DARPA/Lincoln laboratory IDS evaluation data with NETADHICT // 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. IEEE. 2009. P. 1-7.

32. Bolon-Canedo V., Sanchez-Marono N., Alonso-Betanzos A. Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset // Expert Systems with Applications. 2011. Vol. 38. No. 5. P. 5947-5957.

33. Tavallaee M. et al. A detailed analysis of the KDD CUP 99 data set // 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE. 2009. P. 1-6.

34. Dhanabal L., Shantharajah S. P. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms // International journal of advanced research in computer and communication engineering. 2015. Vol. 4. No. 6. P. 446-452.

35. Moustafa N., Slay J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems // IEEE Transactions on Information Forensics and Security. 2020. Vol. 15. P. 3491–3504.

36. Sahu S. K., Sarangi S., Jena S. K. A detail analysis on intrusion detection datasets // 2014 IEEE international advance computing conference (IACC). IEEE. 2014. P. 1348-1353.

37. Khraisat A. et al. Survey of intrusion detection systems: techniques, datasets and challenges // Cybersecurity. 2019. Vol. 2. No. 1. P. 1-22.

38. Ghurab M. et al. A detailed analysis of benchmark datasets for network intrusion detection system // Asian Journal of Research in Computer Science. 2021. Vol. 7. No. 4. P. 14-33.

39. Mehmood T., Helmi B. Machine learning algorithms in context of intrusion detection // 2016 3rd international conference on computer and information sciences (ICCOINS). IEEE. 2016. P. 369-373.

40. Amor N. B., Benferhat S., Elouedi Z. Naive bayesian networks in intrusion detection systems // Proc. Workshop on Probabilistic Graphical Models for Classification, 14th European Conference on Machine Learning (ECML) and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). 2003. 258 P.

41. Jemili F., Zaghdoud M., Ahmed M. B. A framework for an adaptive intrusion detection system using Bayesian network // 2007 IEEE Intelligence and Security Informatics. IEEE. 2007. P. 66-70.

42. Thomas T. et al. Applications of decision trees // Machine learning approaches in cyber security analytics. 2020. P. 157-184.

43. Garcia V. H., Monroy R., Quintana M. Web attack detection using ID3 // IFIP World Computer Congress, TC 12. Springer. Boston, MA. 2006. P. 323-332.

44. Quinlan J. R. C4.5: programs for machine learning. // Elsevier. 2014. 312 P.

45. Hssina B. et al. A comparative study of decision tree ID3 and C4.5 // International Journal of Advanced Computer Science and Applications. 2014. Vol. 4. No. 2. P. 13-19.

46. Kruegel C., Toth T. Using decision trees to improve signature-based intrusion detection // International workshop on recent advances in intrusion detection. Springer. Berlin, Heidelberg. 2003. P. 173-191.

47. Resende P. A. A., Drummond A. C. A survey of random forest based methods for intrusion detection systems // ACM Computing Surveys (CSUR). 2018. Vol. 51. No. 3. P. 1-36.

48. Павлычев А.В., Стародубов М.И., Галимов А.Д. Использование алгоритма машинного обучения Random Forest для выявления сложных компьютерных инцидентов // Вопросы кибербезопасности. 2022. Т.51. № 5. С. 74-81.

49. Павлычев А.В., Кузьминец К.В. Выявление фишинговых интернет-доменов с помощью алгоритмов машинного обучения в режиме потоковой обработки данных // Вопросы кибербезопасности. 2025. Т.66. № 2. С. 141-153.

50. Negandhi P., Trivedi Y., Mangrulkar R. Intrusion detection system using random forest on the NSL-KDD dataset // Emerging Research in Computing, Information, Communication and Applications: ERCICA 2018, Volume 2. Springer Singapore. 2019. P. 519-531.

51. Biswas S. K. et al. Intrusion detection using machine learning: A comparison study / /International Journal of pure and applied mathematics. 2018. Vol. 118. No. 19. P. 101-114.

52. Pathak A. et al. Study on decision tree and KNN algorithm for intrusion detection system // International Journal of Engineering Research & Technology. 2020. Vol. 9. No. 5. P. 376-381.

53. Aburomman A. A., Reaz M. B. I. A novel SVM-kNN-PSO ensemble method for intrusion detection system // Applied Soft Computing. 2016. Vol. 38. P. 360-372.

54. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey // ACM computing surveys (CSUR). 2009. Vol. 41. No. 3. P. 1-58.

55. Hendry G. R., Yang S. J. Intrusion signature creation via clustering anomalies // Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008. SPIE. 2008. Vol. 6973. P. 119-130.

56. Montesinos López O. A., Montesinos López A., Crossa J. Fundamentals of artificial neural networks and deep learning // Multivariate statistical machine learning methods for genomic prediction. Springer International Publishing. 2022. P. 379-425.

57. Dastres R., Soori M. Artificial neural network systems // International Journal of Imaging and Robotics (IJIR). 2021. Vol. 21. No. 2. P. 13-25.

58. Cannady J. Artificial neural networks for misuse detection // National information systems security conference. 2018. Vol. 26. P. 443-456.

59. Salem F. M., Salem F. M. Recurrent neural networks (RNN) // Recurrent Neural Networks: From Simple to Gated Architectures. 2022. P. 43-67.

60. Shenfield A., Day D., Ayesh A. Intelligent intrusion detection systems using artificial neural networks // Ict Express. 2018. Vol. 4. No. 2. P. 95-99.

61. Zeiler M. D., Fergus R. Visualizing and understanding convolutional networks // European conference on computer vision. Springer. 2014. P. 818-833.

62. Narayan V., Shanmugapriya D. Big data analytics with machine learning and deep learning methods for detection of anomalies in network traffic // Research

Anthology on Big Data Analytics, Architectures, and Applications. IGI Global. 2022. P. 678-707.

63. Khan R. U. et al. An improved convolutional neural network model for intrusion detection in networks // 2019 Cybersecurity and cyberforensics conference (CCC). IEEE. 2019. P. 74-77.

64. Millar K. et al. Using convolutional neural networks for classifying malicious network traffic // Deep Learning Applications for Cyber Security. Springer. 2019. P. 103-126.

65. Al-Safaar D., Al-Yaseen W. L. Hybrid AE-MLP: Hybrid Deep Learning Model Based on Autoencoder and Multilayer Perceptron Model for Intrusion Detection System // International Journal of Intelligent Engineering & Systems. 2023. Vol. 16. No. 2.

66. Kasongo S. M. A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework // Computer Communications. 2023. Vol. 199. P. 113-125.

67. Chandrasekhar A. M., Raghuveer K. Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers // 2013 International Conference on Computer Communication and Informatics. IEEE. 2013. P. 1-7.

68. Chen W. H., Hsu S. H., Shen H. P. Application of SVM and ANN for intrusion detection // Computers & Operations Research. 2005. Vol. 32. No. 10. P. 2617-2634.

69. Pervez M. S., Farid D. M. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs // The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014). IEEE. 2014. P. 1-6.

70. Abraham A., Grosan C., Martin-Vide C. Evolutionary design of intrusion detection programs // Int. J. Netw. Secur. 2007. Vol. 4. No. 3. P. 328-339.

71. Kalnoor G., Gowrishankar S. A model for intrusion detection system using hidden Markov and variational Bayesian model for IoT based wireless sensor

network // International Journal of Information Technology. 2022. Vol. 14. No. 4. P. 2021-2033.

72. Du Y., Wang H., Pang Y. A hidden Markov models-based anomaly intrusion detection method // Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No. 04EX788). IEEE. 2004. Vol. 5. P. 4348-4351.

73. Joshi S. S., Phoha V. V. Investigating hidden Markov models capabilities in anomaly detection // Proceedings of the 43rd annual Southeast regional conference – Volume 1. 2005. P. 98-103.

74. Fan W. et al. Using artificial anomalies to detect unknown and known network intrusions // Knowledge and Information Systems. 2004. Vol. 6. P. 507-527.

75. Buczak A. L., Guven E. A survey of data mining and machine learning methods for cyber security intrusion detection //IEEE Communications surveys & tutorials. 2015. Vol. 18. No. 2. P. 1153-1176.

76. Павлычев А. В. Нотация и модификация методики выявления компьютерных инцидентов в соответствии с серией ГОСТ 59709-59712 // Доклады Томского государственного университета систем управления и радиоэлектроники. 2025. Т. 28. №. 3. С. 73-81.

77. Ren H. et al. Time-series anomaly detection service at Microsoft // Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019. P. 3009-3017.

78. Windows Events URL: https://docs.microsoft.com/en-us/windows/win32/events/windows-events (дата обращения: 02.08.2025).

79. Stallings W. The Windows Operating System // Operating Systems: Internals and Design Principles. 2005.

80. Amer E., Zelinka I. A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence // Computers & Security. 2020. Vol. 92. P. 1017-160.

81. Sahoo P. K., Chottray R. K., Pattnaiak S. Research issues on Windows event log // International Journal of Computer Applications. 2012. Vol. 41. No. 19.

82. Kotenko I. V., Levshun D. A. Machine Learning Methods of Intelligent System Event Analysis for Multistep Cyberattack Detection // Scientific and Technical Information Processing. 2024. Vol. 51. No. 5. P. 372-381.

83. Павлычев А. В., Солдатов К. С., Сказин В. А. Выявление сетевых аномалий в системных журналах операционной системы Microsoft Windows с использованием методов машинного обучения // Доклады Томского государственного университета систем управления и радиоэлектроники. 2021. Т. 24. №. 4. С. 27-32.

84. Павлычев А.В., Стародубов М.И., Галимов А.Д. Модель функционирования вредоносного программного обеспечения на основе анализа системных журналов операционной системы Microsoft Windows // Прикаспийский журнал: управление и высокие технологии. 2022. Т.66. № 4. С. 24-31.

85. Павлычев А.В. Формальная модель компьютерных атак, основанная на записях системных событий операционной системы // Научно-аналитический журнал «Вестник Санкт-Петербургского университета Государственной противопожарной службы МЧС России». 2026. № 1. С. 159–169.

86. Carvalho D. V., Pereira E. M., Cardoso J. S. Machine learning interpretability: A survey on methods and metrics // Electronics. 2019. Vol. 8. No. 8. P. 832.

87. Banerjee J. et al. Impact of machine learning in various network security applications // 2019 3rd International conference on computing methodologies and communication (ICCMC). IEEE. 2019. P. 276-281.

88. Magomedov S., Ilin D., Nikulchev E. Resource analysis of the log files storage based on simulation models in a virtual environment //Applied Sciences. – 2021. Vol. 11. №. 11. P. 4718.

89. Касимова А. Р., Золотарев В. В. Моделирование вектора сетевых атак на локальную сеть с применением базы MITRE ATT&CK // Прикаспийский журнал: управление и высокие технологии. 2023. №. 4 (64). С. 88-96.

90. Dwyer J., Truta T. M. Finding anomalies in windows event logs using standard deviation // 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing. IEEE. 2013. P. 563-570.

91. Smiliotopoulos C., Kambourakis G., Barbatsalou K. On the detection of lateral movement through supervised machine learning and an open-source tool to create turnkey datasets from Sysmon logs // International Journal of Information Security. 2023. Vol. 22. No. 6. P. 1893-1919.

92. Smiliotopoulos C. Use of Sysmon tool to detect lateral movement attacks. 2022.

93. Smiliotopoulos C., Barmpatsalou K., Kambourakis G. Revisiting the detection of lateral movement through Sysmon // Applied Sciences. 2022. Vol. 12. No. 15. P. 7746.

94. Rahal K., Riahi A., Debatty T. Dataset of APT Persistence Techniques on Windows Platforms Mapped to the MITRE ATT&CK Framework // 2025 28th Conference on Innovation in Clouds, Internet and Networks (ICIN). IEEE. 2025. P. 17-24.

95. Husselman L. Anomaly Detection with Windows Event Logs: A comparative study between traditional and ML based approaches. University of Zurich. 2024.

96. Pavlychev A.V. at al. Automated Collection And Marking Of Operating System Log Entries In The Task Of Detecting Computer Attacks // Advances in Automation VII. RusAutoCon 2025. Lecture Notes in Electrical Engineering. 2025. Vol. 1521. P. 266–276.

97. Свидетельство о государственной регистрации программы для ЭВМ № 2026618641. Российская Федерация. Программа для моделирования компьютерных атак и формирования наборов данных из системных событий

операционной системы / Павлычев А.В. Заявка № 2026617650. Дата поступления 18.03.2026. Зарегистрировано в реестре программ для ЭВМ 26.03.2026.

98. Павлычев А.В., Кузьминец К.В., Бреус Д.Е., Шелупанов А.А. Формирование размеченного набора данных на основе смоделированных компьютерных атак // Безопасность информационных технологий. 2025. Т. 32. № 4. С. 1–18.

99. Soofi A. A., Awan A. Classification techniques in machine learning: applications and issues // Journal of Basic & Applied Sciences. 2017. Vol. 13. P. 459-465.

100. Liang J. Confusion matrix: Machine learning // POGIL Activity Clearinghouse. 2022. Vol. 3. No. 4.

101. Sarang P. Decision Tree: A Supervised Learning Algorithm for Classification // Thinking Data Science: A Data Science Practitioner’s Guide. Springer International Publishing. 2023. P. 75-96.

102. Obid S. J. et al. Non-Parametric Methods. K-Nearest Neighbors Model // International Journal of Advance Scientific Research. 2023. Vol. 3. No. 12. P. 18-25.

103. Sarang P. Naive Bayes: A Supervised Learning Algorithm for Classification // Thinking data science: A data science practitioner’s guide. Springer International Publishing. 2023. P. 143-152.

104. Testas A. Support vector machine classification with Pandas, scikit-learn, and PySpark // Distributed Machine Learning with PySpark: Migrating Effortlessly from Pandas and Scikit-Learn. Berkeley, CA. Apress. 2023. P. 259-280.

105. Testas A. Random Forest Classification with Scikit-Learn and PySpark // Distributed Machine Learning with PySpark: Migrating Effortlessly from Pandas and Scikit-Learn. Berkeley, CA. Apress. 2023. P. 243-258.

106. Biau G., Scornet E. A random forest guided tour // Test. 2016. Vol. 25. No. 2. P. 197-227.

107. Salman H. A., Kalakech A., Steiti A. Random forest algorithm overview // Babylonian Journal of Machine Learning. 2024. Vol. 2024. P. 69-79.

108. Carvalho D. V., Pereira E. M., Cardoso J. S. Machine learning interpretability: A survey on methods and metrics // Electronics. 2019. Vol. 8. No. 8. P. 832.

109. Ayua S. I. Random forest ensemble machine learning model for early detection and prediction of weight category // Journal of Data Science and Intelligent Systems. 2024. Vol. 2. No. 4. P. 233-240.

Приложение А Свидетельство о регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2026618641

Программа для моделирования компьютерных атак и формирования наборов данных из системных событий операционной системы

Правообладатель: ***Павлычев Алексей Викторович (RU)***

Автор(ы): ***Павлычев Алексей Викторович (RU)***

Заявка № **2026617650**
Дата поступления **18 марта 2026 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***26 марта 2026 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 00a570e4f7add8d531b4b8818e75f29506
Владелец **Зубов Юрий Сергеевич**
Действителен с 04.09.2025 по 28.11.2026

Ю.С. Зубов

Приложение Б Акты внедрения и использования результатов диссертационной работы

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)
690922, Приморский край,
г. Владивосток, о. Русский, п. Аякс, 10
Тел. (423) 243 34 72, факс (423) 243 23 15
Эл. почта: rectorat@dvfu.ru http://www.dvfu.ru
ОКПО 02067942, ОГРН 1022501297785
ИНН/КПП 2536014538/254001001

02.12.2025 № 145/10/10-2463

В диссертационный совет
24.2.415.04 Томского
государственного университета
систем управления и
радиоэлектроники (ТУСУР)

АКТ ВНЕДРЕНИЯ
результатов диссертационной работы Павлычева А.В.

Результаты диссертационного исследования Павлычева А.В. на тему «Методика обнаружения компьютерных атак по записям системных журналов операционной системы с использованием алгоритмов машинного обучения» внедрены в учебный процесс Дальневосточного федерального университета в рамках дисциплины «Интеллектуальные информационные системы», реализуемой департаментом информационной безопасности Института математики и компьютерных технологий.

Внедрены следующие научно-методические разработки:

1. Модернизированная методика выявления компьютерных атак с использованием алгоритмов машинного обучения, основанная на требованиях стандартов ГОСТ серии 59709-59712;
2. Формальная модель компьютерных атак и легитимных действий пользователей на основе анализа системных журналов операционной системы;
3. Алгоритм моделирования компьютерных атак и формирования набора данных из системных журналов, расширяющий перечень выявляемых техник по матрице MITRE ATT&CK;
4. Обученная модель классификатора для различения атак и легитимных действий и разработанное на ее основе консольное программное средство.

Внедрение полученных результатов позволяет повысить практическую ориентированность учебного процесса за счет использования современных, актуальных методик и реальных данных, а также направлено на формирование у обучающихся компетенций в области создания и применения интеллектуальных систем для задач информационной безопасности.

Заместитель директора по науке Института математики и компьютерных технологий ДВФУ И.Л. Артемьева

информационный центр

690014, г. Владивосток
ул. Некрасовская, д. 88А, 3 этаж
тел.: 8 (423) 240-48-66
www.ic-dv.ru, mail@ic-dv.ru

от «23» мая 2025 г. № 05-25/0834
на № _____ от «__» _____ 2025 г.

Доценту департамента ИБ ДВФУ
Павлычеву А.В.

ООО "Информационный центр" является лицензиатом ФСТЭК России и ФСБ России, а также аккредитованным центром ГосСОПКА,.

Компания более 15 лет реализует различные проекты в области информационной безопасности:

- любые аудиты ИБ (в том числе по ГОСТ 57580);
- разработка проектной, технической и эксплуатационной документации по ИБ;
- поставка и внедрение средств защиты информации;
- услуги центра мониторинга событий безопасности;
- аттестация информационных систем по требованиям безопасности информации;
- исследования фактической защищенности (анализ уязвимостей, тестирование на проникновение).

Специалистам центра мониторинга на оценку представлена методика обнаружения компьютерных атак с использованием алгоритмов машинного обучения, а также программное средство обнаружения компьютерных атак evtx_ml_analyzer.py, которое позволяет анализировать файлы системных журналов операционной системы Windows.

Предложенная методика разработана согласно государственным стандартам России в области защиты информации, расширяет и дополняет ее. Использование элементов машинного обучения представляет значительный интерес, поскольку для обнаружения современных компьютерных атак необходимо использовать аналитические инструменты, способные выявлять аномалии в трафике и системных логах.

При расследовании компьютерных инцидентов специалистам центра мониторинга зачастую приходится вручную исследовать файлы журналов операционной системы с целью поиска индикаторов компрометации. Данный процесс может занимать от 1 часа до нескольких дней. Программа evtx_ml_analyzer.py позволяет в полуавтоматическом режиме анализировать файлы системных журналов операционной системы Windows, значительно ускоряя процесс расследования.

В рамках тестирования проверены различные журналы: 10 из них были заведомо с признаками проведения компьютерных атак, 10 с обычных офисных компьютеров. Программа верно детектировала все 10 зараженных журналов с указанием времени обнаружения признаков атаки. Чистые журналы программа также верно классифицировала как безопасные. Среднее время обработки журналов составило 5-10 минут.

С учетом изложенного, предложенные методика и программное средство оцениваются как имеющие практическое применение и внедрены в деятельность центра мониторинга ООО «Информационный центр».

Генеральный директор
ООО «Информационный центр» В.С. Ким

Исполнитель Шашонкова М.Е.