Аккредитованное образовательное частное учреждение высшего образования «Московский финансово-юридический университет МФЮА»
(Московский финансово-юридический университет МФЮА)

На правах рукописи

Русанов Михаил Андреевич

Интеллектуальные методы обнаружения несанкционированных вторжений в операционные системы

Специальность: 2.3.5 – «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
Кандидат физико-математических наук, доцент
Лапина Мария Анатольевна

Москва – 2026

ОГЛАВЛЕНИЕ

Введение

Актуальность работы. В условиях цифровой трансформации возрастает сложность современных информационно-вычислительных систем, характеризующихся распределенной архитектурой, высокой интенсивностью обмена данными и увеличением объемов потоковой информации. Современные программные комплексы функционируют в гетерогенных вычислительных средах, включающих облачные платформы, распределенные сервисы и ресурсно-ограниченные вычислительные узлы, что накладывает дополнительные ограничения на методы интеллектуальной обработки системных событий.

Развитие распределённых облачных технологий и программной инфраструктуры организации глобально распределённой обработки данных приводит к росту объёмов потоковых данных о работе программных компонентов. Анализ таких потоковых данных в режиме реального времени с высокой точностью представляет собой сложную научно-практическую задачу.

Традиционные подходы, основанные на правилах, сигнатурах и шаблонах, недостаточно адаптивны и не учитывают связи между событиями, поэтому всё чаще применяются методы машинного обучения. Используются модели семейства трансформеров, позволяющие учитывать контекстные взаимосвязи между элементами последовательностей системных событий и предназначенных для анализа текстовых данных. Их применение позволяет выявлять аномалии в последовательности системных событий и обеспечивать устойчивость и защищенность системы. Вместе с тем их практическое применение остаётся ресурсоёмким, что делает актуальной оптимизацию потокового вывода, направленную на повышение пропускной способности, сокращение времени обработки и параллельную обработку данных без изменения архитектуры модели.

Значительный научный вклад в рассматриваемую область внесли следующие исследователи: А.В. Федорченко, И.В. Котенко, А.В. Хорошилов, И. Г. Сидоркина, С. В. Михалищев, A. L. Buczak, E. Guven, R. Sommer и V. Paxson, T. Vyšniūnas, D. Čeponis, N. Goranin, A. Čenys, F. Wang, Q. Weng, M. Zhang, Y. Shao, Z. Alomari, A. Makanju, Z. Li, Zhang S., Zhao P., An Z. и др.

Степень разработанности темы исследования.

Выявление несанкционированного проникновения в операционные системы с использованием машинного обучения активно рассматривается в научной литературе, однако универсальное решение пока не сформировано. Основные трудности связаны с разнообразием сценариев атак, изменчивостью поведения нарушителя и разрывом между лабораторными экспериментами и промышленной эксплуатацией. В обзорной работе Buczak и Guven [1] систематизированы методы машинного обучения и интеллектуального анализа данных, применяемые в задачах обнаружения вторжений. Авторы рассматривают классификацию, кластеризацию, ассоциативный анализ и гибридные схемы, а также показывают, для каких типов IDS-задач эти подходы наиболее применимы.

При использовании наборов KDD Cup, NSL-KDD и UNSW-NB15 необходимо учитывать их ограничения. Часть профилей атак в них устарела, а нормальное поведение представлено недостаточно разнообразно. Поэтому качество ML-модели определяется не только выбранным алгоритмом, но и тем, как сформировано признаковое описание, каким способом векторизуются события и насколько корректно поставлена сама задача. Модель, демонстрирующая высокий результат при обнаружении DoS-атак, может оказаться менее устойчивой при анализе APT-цепочек или латерального перемещения.

На экспериментальных наборах такие модели часто выглядят убедительно по метрикам, но перенос в рабочий контур обычно оказывается сложнее. В журналах появляются неполные и шумные записи, доля редких атак мала, обычное поведение сотрудников постепенно меняется, а атакующие подстраивают последовательность команд и маскировку под существующие средства контроля. Поэтому результаты, полученные при стабильном распределении трафика, чистой разметке и известном составе атак, следует рассматривать как верхнюю оценку. В промышленной среде эти предпосылки выполняются не полностью, и модель требует дополнительной проверки на данных конкретной инфраструктуры.

R. Sommer и V. Paxson [2] провели анализ применимости методов машинного обучения для задач обнаружения вторжений в сетевую инфраструктуру. Основной вывод работы заключается в том, что задача

обнаружения атак принципиально отличается от классических задач машинного обучения: она характеризуется высокой неопределённостью, сильной зависимостью от контекста и отсутствием стабильного распределения «нормального» поведения.

В работе [3] рассматривается задача обнаружения вторжений в операционные системы на основе анализа системных вызовов с применением методов машинного обучения. Авторы предлагают метод группировки последовательностей системных вызовов с учетом их уровня риска, при котором различные системные вызовы ранжируются по степени потенциальной опасности и объединяются в информативные последовательности признаков. Такой подход позволяет существенно повысить качество классификации по сравнению с использованием «сырых» последовательностей вызовов, поскольку снижает шум и усиливает различимость поведенческих паттернов.

Особое внимание в работе уделяется применению моделей машинного обучения для анализа полученных признаков в задачах host-based intrusion detection. Показано, что использование риск-ориентированного представления системных вызовов улучшает эффективность обнаружения вредоносной активности и позволяет выявлять скрытые этапы атак на уровне операционной системы. Из рассмотренных работ следует, что системные вызовы остаются удобным источником признаков для поиска проникновений: по ним можно восстановить порядок действий процесса и заметить цепочки, характерные для APT-сценариев. При этом отдельным блоком приходится учитывать работу с учётными данными, потому что захваченная или слабая учётная запись часто даёт атакующему первый устойчивый доступ. В работах [4] и [5] эта проблема рассматривается через оценку стойкости паролей обучаемыми моделями; для обучения используются списки запрещённых, слабых и уже скомпрометированных комбинаций. Современные исследования показывают, что такая предварительная фильтрация учетных данных существенно снижает риск успешного подбора и повышает надежность систем аутентификации.

Объектом исследования являются интеллектуальные системы анализа потоковых событий операционных систем семейства Linux в распределенной

вычислительной среде.

Предмет исследования – интеллектуальные методы, модели и программные средства формирования модулей обнаружения и классификации несанкционированных вторжений в операционные системы семейства Linux.

Целью является разработка интеллектуальных методов и программных средств анализа аутентификационных данных и потоковых системных событий операционных систем семейства Linux, направленных на повышение эффективности обнаружения и классификации несанкционированных вторжений и угроз компрометации учетных данных.

Для решения поставленной научной задачи была произведена ее декомпозиция на ряд **частных задач:**

1. Анализ методов обработки и классификации системных событий операционных систем и выявление ограничений их применения в задачах обнаружения и классификации несанкционированных вторжений.

2. Разработка методов предварительной обработки, представления и интеллектуального анализа потоковых событий операционных систем семейства Linux, включая модели обнаружения и классификации вредоносной активности, а также выявления потенциально компрометируемых аутентификационных данных.

3. Разработка методов и программных средств параллельной потоковой обработки событий операционных систем семейства Linux, обеспечивающих снижение времени обработки данных без уменьшения точности обнаружения и классификации.

Методология и методы исследования включают использование методов машинного обучения, обработки естественного языка, анализа данных, теория вероятностей, математическая статистика, а также методы математического моделирования.

Соответствие паспорту научной специальности.

Область исследования соответствует паспорту специальности 2.3.5 – «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» по следующим пунктам:

п. 4. Интеллектуальные системы машинного обучения, управления базами

данных и знаний, инструментальные средства разработки цифровых продуктов:

п. 8. Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования.

п. 9. Модели, методы, алгоритмы, облачные технологии и программная инфраструктура организации глобально распределенной обработки данных.

Научная новизна диссертационной работы:

1. Разработан интеллектуальный метод оценки вероятности компрометации аутентификационных данных операционных систем семейства Linux, отличающийся от существующих подходов использованием статистических характеристик случайности в сочетании с моделями машинного обучения.

2. Разработан интеллектуальный метод обработки потоковых событий операционных систем семейства Linux для обнаружения вредоносной активности, отличающийся от существующих методов применением контекстного представления событий на основе моделей трансформеров с предварительной нормализацией и семантическим упрощением событий, что позволяет повысить точность классификации атак и уменьшить время работы моделей.

3. Разработан метод параллельной потоковой обработки событий операционных систем семейства Linux, отличающийся от существующих подходов использованием адаптивного формирования вычислительных пакетов, динамического выравнивания длины входных последовательностей и группировки событий по вычислительной сложности, что обеспечивает уменьшение времени обработки и повышение пропускной способности интеллектуальных моделей анализа системных событий без изменения архитектуры модели и снижения точности обнаружения и классификации.

Теоретическая значимость результатов диссертационного исследования заключается в следующем**:**

1) Разработанный метод выявления потенциально компрометируемых аутентификационных данных позволяет уточнить оценки вероятности их компрометации за счет интеллектуального анализа косвенных признаков.

2) Разработанный интеллектуальный метод анализа событий операционных

систем семейства Linux на основе языковых моделей вносит вклад в развитие теоретических основ для обнаружения вредоносной активности и расширяет представления о их применимости для анализа сложных и модифицированных атак.

3) Разработанный метод параллельной потоковой обработки событий операционных систем семейства Linux позволяет уменьшить время их обработки за счет адаптивного формирования вычислительных пакетов, динамического выравнивания длины входных последовательностей и группировки событий по вычислительной сложности.

Практическая значимость результатов диссертационного исследования заключается в следующем**:**

1) Разработанный метод выявления потенциально компрометируемых аутентификационных данных позволяет уточнить вероятность их компрометации, что обеспечивает возможность его применения в подсистемах контроля защищенности, аудита безопасности и управления доступом.

2) Разработанный интеллектуальный метод анализа событий операционных систем семейства Linux на основе языковых моделей позволяет уменьшить время их обработки в среднем в 5 раз, что обеспечивает возможность его применения при создании программного обеспечения для центров мониторинга информационной безопасности.

3) Разработанный метод параллельной потоковой обработки событий операционных систем семейства Linux позволяет уменьшить время обработки в среднем в 2 раза и повысить пропускную способность интеллектуальных подсистем мониторинга информационной безопасности на 123% без снижения точности классификации, что обеспечивает возможность его применения при создании программного обеспечения для центров мониторинга информационной безопасности.

На защиту выносятся следующие положения:

1. Интеллектуальный метод оценки вероятности компрометации аутентификационных данных операционных систем семейства Linux, основанный на вычислении статистических характеристик случайности в сочетании с моделями

машинного обучения.

2. Интеллектуальный метод обработки потоковых событий операционных систем семейства Linux для обнаружения вредоносной активности, основанный на применении контекстного представления событий и моделей трансформеров с предварительной нормализацией и семантическим упрощением событий.

3. Метод параллельной потоковой обработки событий операционных систем семейства Linux с использованием адаптивного формирования вычислительных пакетов, динамического выравнивания длины входных последовательностей и группировки событий по вычислительной сложности.

Достоверность и обоснованность полученных в диссертационной работе теоретических результатов и формулируемых на их основе выводов подтверждается корректным и обоснованным применением классических методов исследования, строгими математическими доказательствами и результатами анализа эффективности реализации разработанных моделей и методов с использованием языка программирования Python. Результаты теоретического анализа согласуются с результатами проведенных экспериментов.

Публикации. По теме диссертации опубликовано 6 печатных работ, в том числе в изданиях, рекомендованных ВАК [6,7,10] и Web of Science и Scopus [8,9], а также получено 2 свидетельства о государственной регистрации программ для ЭВМ [12,13].

Личный вклад автора. Диссертационная работа представляет собой исследование автора, объединенное тематикой и методами исследования. Все выносимые на защиту результаты получены лично автором. Из совместных работ в диссертацию включены только те результаты, которые принадлежат непосредственно автору. В опубликованных совместных работах постановка и решение задач осуществлялись совместными усилиями соавторов при непосредственном участии соискателя. В работе [8] автором разработан интеллектуальный метод оценки вероятности компрометации аутентификационных данных, основанный на анализе статистических характеристик случайности паролей и применении моделей машинного обучения для выявления потенциально компрометируемых паролей. В работах [6,7,9-11]

автором разработан интеллектуальный метод обработки потоковых системных событий операционных систем семейства Linux, обеспечивающий обнаружение вредоносной активности и классификацию атак по тактикам MITRE ATT&CK на основе контекстного представления событий и трансформерных моделей, а также повышение вычислительной эффективности их потоковой обработки за счёт адаптивного формирования вычислительных пакетов и динамического выравнивания входных последовательностей. В государственных регистрациях программ для ЭВМ [12,13] автором выполнена разработка программных комплексов обнаружения и классификации вредоносной активности в системных событиях и приложениях на базе нейронной сети.

Апробация работы. Результаты данной работы докладывались на конференциях:

1. XIII Всероссийская научно-техническая конференция с международным участием «Актуальные проблемы информационной безопасности», 2025, Самара;

2. Международная весенняя конференция молодых учёных в области программной инженерии (SYRCoSE Software Engineering Colloquium), 2025, Пятигорск;

3. Всероссийская конференция с международным участием "Радиоэлектронные устройства и системы для инфокоммуникационных технологий (РЭУС-ИТ), 2026, Москва;

4. XVII Международная научно-практическая конференция имени Олега Борисовича Макаревича «Современные методы, средства и технологии защиты информации – 2026», 2026, Таганрог.

5. Международная конференция «AITHD-2025: International Conference on Artificial intelligence as a Technologies of Human Development», 2025, Ставрополь.

Внедрение результатов. Результаты, полученные в рамках данной работы, были внедрены в подсистемы мониторинга событий информационной безопасности в АО «Газпромбанк» (акт о внедрении 302-3/12/26 от 05.05.2026).

Структура диссертации. Полный объем диссертации составляет 130 страниц, включая 30 рисунков и 20 таблиц. Список литературы содержит 108 наименований.

Глава 1 Исследование методов обнаружения несанкционированного вторжения в операционные системы

1.1 Исследование существующих решений

Командная строка предоставляет широкие возможности для администрирования системы, однако в большинстве случаев она является основным инструментов проведения атак злоумышленниками и используется для выполнения различных вредоносных действий таких как загрузка и запуск вредоносного программного обеспечения, повышение привилегий или кража данных. Аналогично, веб-приложения являются частой целью атак, причем приблизительно 80% атак на веб-сервисы выполняются через HTTP и HTTPS запросы, согласно данным Open Web Application Security Project (OWASP) [14]. Такие действия часто оставляют следы как в системных журналах операционной системы, так и в журналах приложений, что делает их основным источником информации для выявления атак.

Корректно настроенный аудит событий в сочетании с традиционными системами обнаружения атак, которые основаны на сигнатурах и правилах, может выявлять вредоносную активность даже на самых ранних этапах. Однако этот подход зачастую оказывается неэффективным против современных атак, когда злоумышленник может оставаться незамеченным внутри инфраструктуры в течение длительного периода времени, выполняя минимум возможных действия [15]. Более того, методы, основанные на правилах, часто оказываются неспособными выявлять вредоносные полезные нагрузки, встроенные в обфусцированные команды.

Для решения задачи совершенствования систем обнаружения атак исследователи начали применять методы машинного обучения [6] и обработки естественного языка для анализа системных событий и выявления аномалий [7]. Например, в работе [16] представлен новый подход к классификации событий центра безопасности (СЦБ), предполагающий определение набора новых признаков на основе графового анализа и выполнение классификации с использованием глубоких нейронных сетей.

Несмотря на значительные достижения в применении методов машинного обучения [6,7] к анализу логов безопасности, большинство существующих исследований ориентировано на среды серверов или облачных вычислений, где ресурсы не являются критичным ограничением [17]. Такие модели часто предполагают наличие мощных графических процессоров, больших объемов оперативной памяти и постоянного сетевого соединения. Однако в реальных инфраструктурах - включая устройства интернета вещей, промышленные контроллеры, удаленные серверы пограничных вычислений - такие возможности часто недоступны. В результате высокоточные алгоритмы становятся практически непригодными в условиях ограниченной вычислительной мощности [18]. Следовательно, задача эффективной передачи моделей обнаружения угроз в маломощные среды остается недостаточно изученной и требует разработки специализированных методов оптимизации и предварительной обработки данных.

Современные методы обнаружения вредоносной активности в большинстве систем сталкиваются с двумя ключевыми ограничениями. Во-первых, высокоточные модели анализа журналов, основанные на глубоких архитектурах, обычно требуют значительных вычислительных затрат, что усложняет их использование в средах с ограниченными ресурсами [19]. Во-вторых, традиционные сигнатурные и эвристические подходы часто оказываются недостаточными для обнаружения новых или модифицированных вариантов атак, поскольку они опираются на заранее определенные правила и часто не способны адаптироваться к изменяющемуся поведению злоумышленников. Дополнительную задачу создает структура системных журналов: события содержат большое количество переменных и параметров, что может привести к переобучению на незначительных деталях или искусственным закономерностям, часто присутствующим в синтетических наборах данных. Следовательно, возникает необходимость разработки ресурсоэффективного подхода к обнаружению и классификации вредоносной активности в системных событиях, объединяющего оптимизированную предварительную обработку событий и использование компактных моделей таким образом, чтобы высокая точность и производительность сохранялись при ограниченных вычислительных ресурсах.

Один из вариантов анализа команд операционной системы Linux рассматривается в работе [20], где авторы используют нейронную сеть ShellCore. Они составили крупный набор данных, содержащий как команды обычных пользователей системы, так и вредоносные, извлеченные из различных образцов вредоносных программ для Интернета вещей. Предложенный метод продемонстрировал высокую точность.

С другой стороны, автор работы [21] исследует применение методов обработки естественного языка (ОЕЯ) и методов машинного обучения для анализа команд, сбор которых осуществляется через систему аудита. Особое внимание уделяется токенизации команд оболочки, их преобразованию в числовые векторы с использованием Hashing Vectorizer и обучению моделей для выявления вредоносных команд.

В работе [22] авторы представляют платформу LOLAL, предназначенную для обнаружения атак типа Living-Off-The-Land (LotL), при которых злоумышленники используют легитимные системные утилиты и команды операционной системы, затрудняя традиционное сигнатурное обнаружение. Предложенный подход основывается на анализе текстового представления командной строки, где команды рассматриваются как последовательности токенов, отражающие поведение пользователя или процесса. Для повышения семантической выразительности и выявления скрытых паттернов атак используется векторизация слов и эмбеддинговые представления, позволяющие моделировать контекстное сходство между командами, а не только их синтаксическую структуру. Это дает возможность выявлять аномальные последовательности даже в случаях, когда используются стандартные системные утилиты (например, PowerShell, Bash или WMI), характерные для LotL-атак. В результате LOLAL обеспечивает более высокую чувствительность к поведенческим отклонениям по сравнению с классическими методами анализа логов, ориентированными на фиксированные правила или сигнатуры.

В работе [23] представлена система обнаружения вторжений, ориентированная на масштабируемый анализ командной активности с использованием языковой модели командной строки. Авторы предлагают

предварительное обучение модели на очень больших объёмах телеметрии: десятки миллионов записей командной строки, собранных из реальных системных логов. Такой подход позволяет модели изучать статистические и семантические закономерности нормального поведения пользователей и процессов в операционной системе.

Ключевая идея заключается в том, что командная строка рассматривается как язык, где отдельные команды и их последовательности формируют «предложения», отражающие действия пользователя. Для их моделирования используется языковая модель CLLM (command-line language model), способная предсказывать или оценивать вероятность появления той или иной последовательности команд. Отклонения от выученного распределения интерпретируются как потенциально аномальное поведение, связанное с вторжением или вредоносной активностью.

Особенностью подхода является его масштабируемость и способность адаптироваться к разнообразным сценариям эксплуатации без необходимости ручного задания правил или признаков. За счёт предварительного обучения на больших данных модель приобретает устойчивость к шуму и вариативности командных паттернов, что позволяет эффективно выявлять как известные, так и ранее не наблюдавшиеся типы атак в реальных условиях эксплуатации.

В работе [24] акцент сделан на трансформерных моделях и переносе знаний из больших корпусов логов. Сначала модель обучается на широком наборе командных последовательностей, а затем донастраивается для поиска редких опасных команд. Другой вариант практической реализации описан в [25]: для обнаружения шифровальщиков используется eBPF, собирающий данные о системных вызовах активных процессов прямо на уровне ядра Linux. Это уменьшает объём обмена с пользовательским пространством и снижает число переключений контекста. Внутрь eBPF-программы авторы помещают лёгкие классификаторы, в том числе Decision Tree и MLP с одним скрытым слоем, поэтому оценка события выполняется сразу при его обработке. Для ransomware-сценариев такая задержка критична: даже несколько секунд могут означать потерю большого числа файлов. При этом код eBPF проходит проверку и работает в изолированной

среде ядра, что уменьшает вероятность того, что сам модуль мониторинга станет дополнительной точкой риска.

Отдельная трудность связана с получением обучающих примеров атак. У организации может не быть достаточной истории инцидентов, а ручная разметка требует специалистов, которые понимают как системные журналы, так и логику действий нарушителя. Искусственно собранные сценарии помогают расширить выборку, но обычно хуже передают разнообразие реального поведения. Кроме того, в журналах нормальных событий почти всегда на порядок больше, чем вредоносных, поэтому обучение без коррекции приводит к смещению модели в сторону большинства. На этом этапе используют семплирование, веса классов или генерацию дополнительных примеров. Ещё одно ограничение связано с размещением модели: тяжёлые решения трудно запускать в ядре Linux и на обычных серверах без запаса ресурсов. Если метод изначально создавался для IoT, контейнеров или виртуализированной среды, перед переносом на типовую Linux-инфраструктуру его приходится отдельно адаптировать.

Модель Digraph-MMB, предложенная в работе [26], использует ориентированные графы для описания многоэтапных атак в среде Linux. Пример такого графа приведён на рисунке 1.1. Основное преимущество модели связано с применением графовых нейронных сетей, способных учитывать связи между отдельными командами и их последовательностями. В экспериментах на наборе Linux-MMA-Dataset-2024 была получена F1-мера около 99,9 %, что относится к числу наиболее высоких результатов для данного класса задач. Вместе с тем такой уровень качества достигается при значительных требованиях к памяти и вычислительным ресурсам. Поэтому развёртывание Digraph-MMB на пограничных устройствах, промышленных контроллерах и маломощных узлах остаётся затруднительным, тогда как в серверной среде с достаточным ресурсным резервом данный подход может использоваться более эффективно.

В качестве основного компонента решения использована модель BERT (Bidirectional Encoder Representations from Transformers), разработанная для предварительного обучения глубоких двунаправленных представлений на неразмеченном тексте с учетом как левого, так и правого контекста на всех

уровнях. Эта архитектура позволяет выполнять тонкую настройку всего с одним дополнительным выходным слоем [27], что делает возможным создание высокоточных классификаторов при минимальных архитектурных модификациях. Эти классификаторы были адаптированы под задачи выявления вредоносной активности в событиях SIEM (Security Information and Event Management) и их классификации в соответствии с матрицей MITRE ATT&CK.

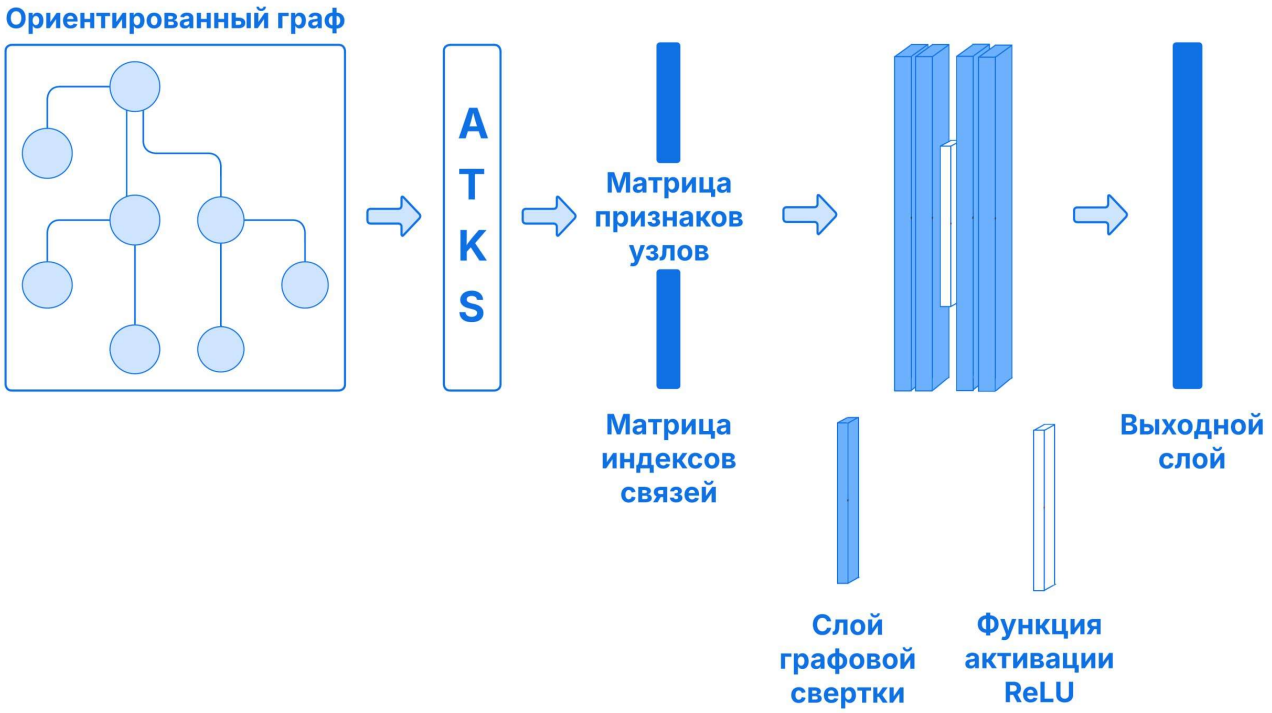


Рисунок 1.1 – Структурная схема рабочего процесса на основе ориентированного графа Digraph MMB [26]

Отказ от ГНС в пользу трансформеров объясняется несколькими причинами.

Ключевым аргументом в пользу трансформерных моделей является их эффективность при анализе последовательных данных, характерных для журналов событий операционных систем и SIEM-платформ. В рамках задач машинного обучения для обеспечения информационной безопасности установлено [28], что даже классификация отдельных команд, системных вызовов и событий безопасности позволяет выявлять ранние признаки многоэтапных атак класса APT. Это имеет принципиальное значение для построения систем раннего обнаружения, поскольку позволяет фиксировать отклонения от нормального поведения на самых низких уровнях операционной системы, не дожидаясь накопления сложных

корреляционных паттернов.

Так, в работе [29] авторы показывают, что короткие последовательности системных вызовов могут эффективно использоваться как поведенческие признаки, позволяющие различать нормальную и аномальную активность различных UNIX-программ. Ключевая идея исследования заключается в том, что классификация отдельных команд и системных вызовов уже на низком уровне операционной системы позволяет фиксировать отклонения от нормального поведения, которые часто соответствуют ранним стадиям атак.

Для противодействия APT-атакам анализ локальных последовательностей команд имеет особое значение. В таких сценариях отклонения проявляются не только на уровне общего поведения системы, но и в порядке вызовов, временных интервалах, сочетаниях параметров и структуре команд. Нарушитель может имитировать штатную активность на макроуровне, однако отдельные низкоуровневые шаблоны выполнения всё равно отличаются от типичного поведения пользователя или администратора. Поэтому анализ системных вызовов, таймингов и комбинаций параметров argv помогает фиксировать ранние этапы вторжения до перехода атаки к закреплению в инфраструктуре, запуску заданий в планировщике или эксфильтрации данных.

Авторы также отмечают, что профили нормального поведения процессов могут формироваться двумя способами: в лабораторных условиях и на основе реальных пользовательских сценариев, собранных с рабочих станций или серверов в период штатной работы. Второй вариант точнее отражает специфику конкретной организации и способствует снижению числа ложных срабатываний. Вместе с тем он требует аккуратного выделения периодов, свободных от атак, что на практике может быть затруднительно.

Построение цепочек APT-атак для обучения моделей требует детального знания тактик, техник и процедур нарушителя, использованных при формировании обучающего набора данных. Если разметка не содержит сведений о конкретных шагах атаки и выполняемых вредоносных командах, классификатор хуже разделяет нормальные и аномальные последовательности, а итоговые результаты становятся менее интерпретируемыми. Из-за отсутствия детальной информации о

поведении злоумышленника определить точную последовательность действий при атаке на основе набора команд в небольших временных окнах становится сильно затруднено, поскольку злоумышленник может выполнять атаки с использованием нескольких тактик одновременно.

В том же исследовании [26] представляется система обнаружения вторжений на основе крупной языковой модели, обученной на десятках миллионов команд в облачной среде. Архитектура модели основана на BERT-base (12 слоев Transformer с размером скрытого представления 768), с токенизацией BPE (словарь 50 тысяч токенов) и обрезкой последовательностей до 1024 токенов. Данные предобрабатываются с использованием парсера bashlex для валидации синтаксиса и фильтрации невалидных команд. Предварительное обучение проводится по задаче Masked Language Modeling (аналогично RoBERTa), где токены маскируются с заданной вероятностью для предсказания по контексту, что позволяет генерировать семантические эмбеддинги команд для последующих задач.

Для обнаружения вторжений применяются несколько подходов. Неинтерпретируемое обнаружение аномалий использует эмбеддинги в комбинации с методами PCA, Isolation Forest и OC-SVM на основе ошибки реконструкции, демонстрируя эффективность на простых атаках (например, masscan), но с путаницей benign-аномалий. Если обучающие метки формируются коммерческими IDS и содержат шум, требуется более сложная схема анализа. В таких конфигурациях применяются реконструкционный метод с чередованием дообучения (fine-tuning) и сингулярного разложения, последующее сокращение размерности через SVD-PCA, классификатор на основе CLS-эмбеддинга, вариант классификации с конкатенацией трёх последовательных команд, а также поиск релевантных фрагментов во внешних источниках через 1-NN в пространстве эмбеддингов. Подобная архитектура масштабируема и обеспечивает достаточно высокое качество обнаружения, но неоптимизированная модель семейства BERT сохраняет значительные требования к памяти и вычислительным ресурсам. Аналогичное ограничение рассматривается в работе [30], где используется архитектура Llama, обычно имеющая больше параметров по сравнению с BERT-подобными моделями. Без оптимизации такие решения трудно разместить на

малопроизводительных устройствах. В корпоративной инфраструктуре с достаточными серверными ресурсами CLLM и LlamaIDS могут использоваться после квантизации, прунинга или замены отдельных слоёв более эффективными компонентами. Архитектура решения изображена на рисунке 1.2.

В работе [31] графовые методы используются для анализа атак на веб-приложения. На основе журналов доступа и событий формируется поведенческий граф, после чего применяется Graph Attention Network (GAT), выделяющая характерные связи между действиями нарушителя. На этапе извлечения признаков языковые модели, включая BERT, используются для получения агрегированных векторных представлений HTTP-запросов и событий веб-сервера. Такой граф описывает не только отдельные обращения, но и их последовательность: например, сканирование, попытку SQL-инъекции и последующий обход фильтрации через закодированные параметры [32]. За счёт этого система может распознавать как типовые угрозы, включая XSS и SQL-инъекции, так и более сложные многоэтапные сценарии взаимодействия с приложением.

В работе [33] контрастивное обучение используется для выделения признаков вредоносных PE- и ELF-файлов. Смысл подхода состоит в том, чтобы перенести образцы в пространство эмбеддингов: близкие варианты одной семьи вредоносного ПО оказываются рядом, а отличающиеся файлы отодвигаются друг от друга. В публикации CrowdStrike [34] это направление развивается за счёт модифицированной triplet loss. Авторы добиваются более устойчивого разделения классов и используют гибридную функцию потерь, которая сохраняет различимость признаков даже при сильном дисбалансе выборки. Для задач кибербезопасности это важно, поскольку редкие вредоносные семейства обычно представлены значительно хуже, чем массовые или нормальные объекты.

В работе [35] задача рассматривается с другой стороны – через сжатие данных средствами глубокого обучения. Авторы показывают, что качество классификации нельзя оценивать отдельно от стоимости обработки: модель должна не только находить атаки, но и экономно использовать память, вычисления и объём передаваемых данных. Такой вывод важен для систем мониторинга, где поток событий поступает непрерывно, и избыточная обработка

быстро становится ограничением.

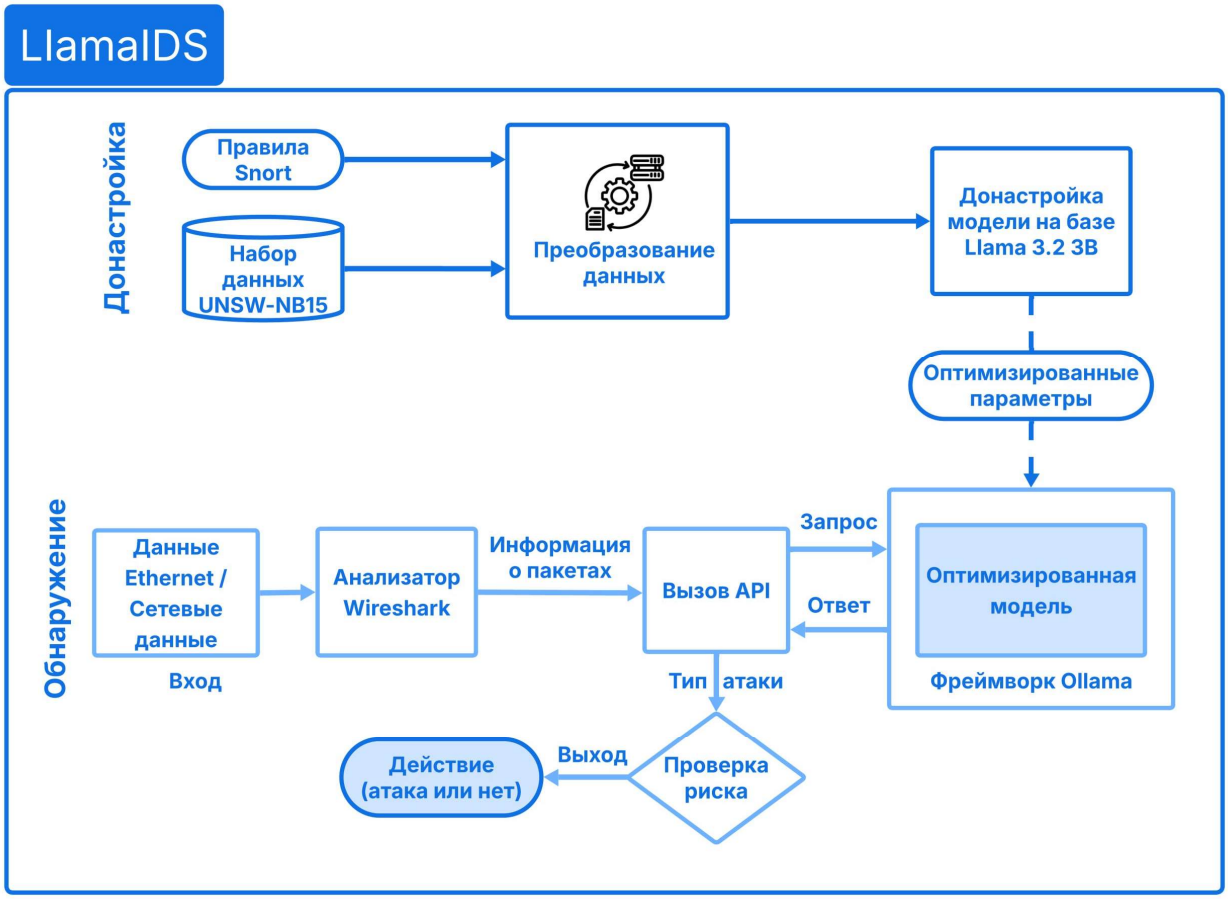


Рисунок 1.2 – Архитектура LlamaIDS [30]

1.2 Анализ существующих подходов к повышению вычислительной эффективности интеллектуальных систем анализа системных событий

При внедрении интеллектуального анализа журналов главным ограничением часто становится не сама точность, а цена её получения. Современная модель может хорошо распознавать вредоносную активность на тесте, но в рабочем контуре ей приходится обрабатывать поток событий без заметной задержки. Если вычислений слишком много, растёт время реакции, снижается пропускная способность подсистемы анализа и появляется риск пропустить часть событий при пиковых нагрузках. Особую актуальность данная проблема приобретает в условиях непрерывного поступления системных событий, характерных для современных SIEM, где требуется обеспечивать обработку больших потоков данных в режиме, близком к реальному времени.

Существующие исследования в области оптимизации вывода моделей

преимущественно ориентированы на уменьшение времени обработки, повышение пропускной способности вычислительной системы и снижение объема избыточных вычислений, возникающих при анализе последовательностей различной длины. При этом существенное внимание уделяется вопросам формирования вычислительных пакетов, динамического выравнивания последовательностей и адаптации вычислительного процесса к характеристикам входных данных. Результаты проведенного сравнительного анализа существующих решений представлены в таблице 1.1.

Таблица 1.1 – Сравнение существующих решений оптимизации вывода моделей

Решение	Архитектура модели	Коэффициент изменения пропускной способности	Коэффициент изменения времени вывода
[36]	Рекуррентные нейронные сети	1.8 – 2.1	-
[37]	Сверточные нейронные сети, рекуррентные нейронные сети, трансформерные модели	1.12 – 1.19	-
[38]	Трансформерные модели	1.5	6.4 – 7.3

В работе [36] рассматривается проблема неэффективного использования вычислительных ресурсов при обработке последовательностей различной длины в составе одного пакета. Авторы отмечают, что при пакетной обработке значительная часть вычислений расходуется на обработку дополнительных элементов выравнивания, вводимых для приведения последовательностей к одинаковому размеру. Особенно выраженной данная проблема становится при совместной обработке коротких и длинных последовательностей, когда вычислительная сложность определяется наиболее длинным элементом пакета.

Эту проблему авторы E-BATCH решают за счёт динамического

формирования пакетов по длине входных последовательностей. При таком разбиении короткие записи меньше зависят от самых длинных примеров в пакете, а лишние операции, возникающие из-за выравнивания до общего размера, сокращаются. В экспериментах метод увеличил пропускную способность в 1,8–2,1 раза по сравнению с обычной пакетной обработкой. Однако область применения всё же ограничена: работа в основном ориентирована на рекуррентные сети и специализированные ускорители, поэтому для потокового анализа текстовых журналов на маломощных узлах этот механизм нельзя перенести без дополнительной переработки.

В работе [37] исследуются особенности пакетного вывода глубоких нейронных сетей и влияние параметров вычислений на производительность. Авторы анализируют связь между размером пакета, структурой входных последовательностей и пропускной способностью системы. Показано, что увеличение пакета уменьшает относительную долю накладных расходов и повышает эффективность использования ресурсов, но только до определённого предела.

Эксперименты подтверждают, что пакетная обработка заметно превосходит последовательный режим по пропускной способности. После достижения некоторого размера пакета прирост замедляется, а задержка обработки может увеличиваться из-за насыщения вычислительных ресурсов. Существенный источник потерь связан с неоднородностью длин входных последовательностей: если короткие и длинные записи попадают в один пакет, общее время обработки определяется наиболее длинной последовательностью. В результате короткие примеры требуют почти таких же затрат, как длинные. В качестве оптимизации предлагается учитывать фактическую длину записей и группировать их более однородно. При низкой и средней нагрузке влияние таких решений на итоговое качество вывода, по оценке авторов, остаётся в пределах статистической погрешности.

Также стоит заметить, что для трансформерной модели семейства Bert предлагаемые механизмы оптимизации либо незначительно дают прирост пропускной способности, либо наоборот уменьшают ее значения примерно на 25%.

В статье [38] предложен комплекс методов повышения эффективности вывода трансформерных моделей, включающий оптимизацию вычислительных операций, распределение вычислительной нагрузки и адаптивную организацию пакетной обработки. Авторы исходят из предположения, что традиционные механизмы вывода крупных языковых моделей недостаточно эффективно используют доступные вычислительные ресурсы, вследствие чего возникают ограничения по времени обработки и пропускной способности системы.

Предложенный подход предусматривает использование специализированных механизмов организации вычислений, позволяющих существенно повысить производительность вывода. Согласно результатам экспериментов, использование предложенных механизмов обеспечивает сокращение времени обработки до 7 раз, а увеличение пропускной способности — более чем в 1,5 раза по сравнению с традиционными механизмами выполнения моделей Transformer. Однако достижение подобных показателей обеспечивается использованием распределённых GPU-вычислений, специализированных библиотек и перераспределением модели между вычислительными устройствами, что усложняет инфраструктуру вычислений и ограничивает применимость подхода в задачах потокового анализа событий безопасности в средах с ограниченными ресурсами.

Таким образом, анализ современных исследований показывает, что наиболее перспективными направлениями оптимизации вывода интеллектуальных моделей являются динамическое формирование вычислительных пакетов, уменьшение объёма избыточных вычислений, возникающих при выравнивании последовательностей, и адаптация вычислительного процесса к характеристикам входных данных. В существующих работах демонстрируется увеличение пропускной способности вычислительных систем и сокращение времени обработки. Вместе с тем большинство подходов ориентировано на специализированные GPU-платформы, распределённые вычислительные среды либо изменение вычислительной инфраструктуры. Вопросы повышения производительности потокового вывода событий безопасности в средах с ограниченными ресурсами без внесения изменений в архитектуру используемой

модели и без снижения качества классификации исследованы ограниченно, что в свою очередь определяет актуальность данного исследования.

1.3 Анализ тактик атак

Перечень рассматриваемых тактик атак строится на концепции Cyber Kill Chain - модели жизненного цикла кибератаки, разработанная компанией Lockheed Martin. Cyber Kill Chain описывает атаку как последовательность логически связанных этапов, начиная с подготовки и разведки и заканчивая достижением цели злоумышленника, такой как кража данных, саботаж или полный контроль над системой. [39] Ключевая идея данной модели заключается в том, что для успешной атаки противник вынужден последовательно проходить через эти стадии, и прерывание атаки на любом из этапов делает невозможным её дальнейшее развитие [40]. Именно поэтому обнаружение признаков атаки не обязательно должно происходить на финальной стадии - раннее выявление отдельных шагов уже обеспечивает эффективную защиту [10].

В общем виде последовательность этапов Cyber Kill Chain изображена на рисунке 1.3.

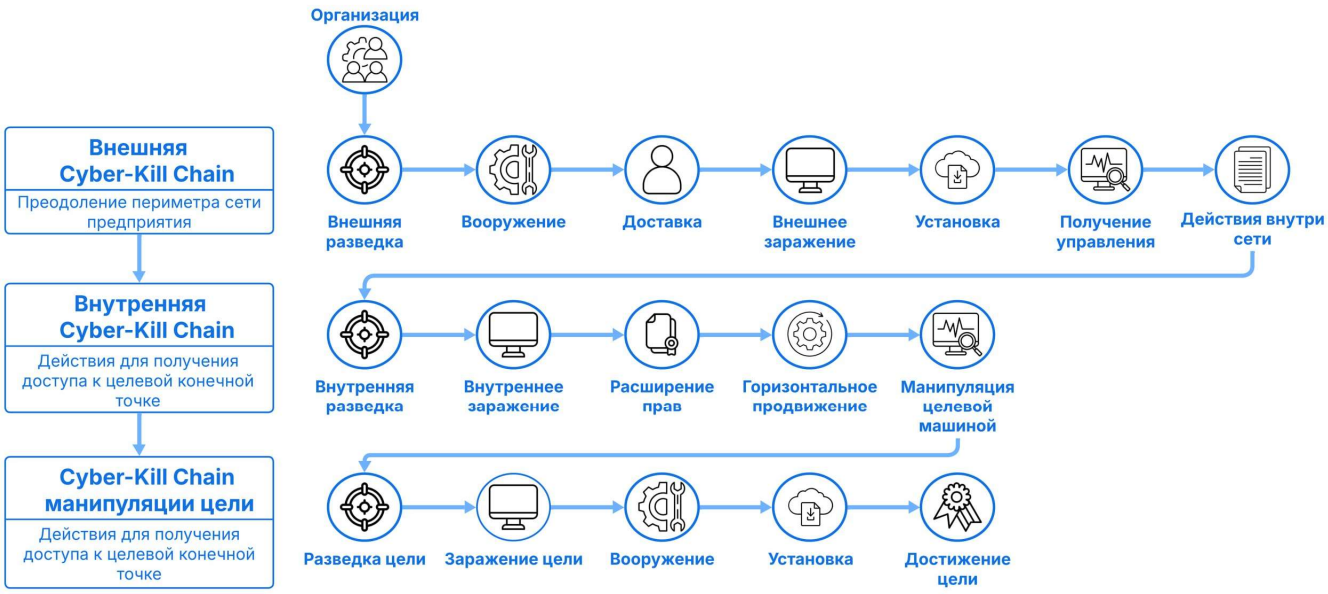


Рисунок 1.3 – этапы Cyber Kill Chain

Тактики, представленные в модели MITRE ATT&CK, могут рассматриваться как формализованное развитие концепции Cyber Kill Chain, адаптированное к анализу реальных поведенческих проявлений атак в сетевых

взаимодействиях, операционных системах и прикладных программных средах. С точки зрения анализа событий безопасности и моделей поведения программных систем, каждая тактика отражает функциональную цель злоумышленника на конкретном этапе реализации системно-ориентированной атаки, включая этапы первичного проникновения, закрепления в операционной системе (ОС), эскалации привилегий и горизонтального перемещения в распределённой инфраструктуре.

Реализация тактик осуществляется через набор техник, которые, несмотря на разнообразие используемых инструментов и программных средств, обладают схожей операционной семантикой на уровне взаимодействия с операционной системой, сетевыми стековыми протоколами и прикладными сервисами. В результате такие действия формируют наблюдаемые артефакты в журналах событий ОС, сетевых логах и телеметрии приложений, что делает возможным их анализ средствами SIEM-систем и интеллектуальных моделей обработки данных. С позиции машинного обучения и управления знаниями подобные следы могут рассматриваться как устойчивые поведенческие признаки, используемые для классификации этапов атаки и выявления аномальной активности в распределённых вычислительных и облачных инфраструктурах [6].

Начальные этапы модели Cyber Kill Chain связаны с фазой разведки и подготовительных действий, которые в современных сетевых и распределённых программных системах рассматриваются как ключевой этап формирования модели угроз. На данном уровне злоумышленник осуществляет сбор информации о целевой инфраструктуре, включая параметры операционной системы, состав установленного программного обеспечения, конфигурации сетевых сервисов, доступные интерфейсы взаимодействия и потенциальные точки входа в распределённую среду. С позиции анализа поведения программных систем этап разведки или рекогносцировки во многом определяет дальнейший ход атаки. На этой стадии нарушитель выбирает возможный вектор компрометации, оценивает доступные сервисы и определяет способы взаимодействия с сетевыми и прикладными компонентами. Раннее выявление признаков разведки позволяет зафиксировать подготовительные действия до фактического внедрения и начала эксплуатации уязвимостей. В распределённых и облачных средах, где сервисы

активно взаимодействуют через сеть и API, блокирование разведывательной активности снижает вероятность перехода APT-сценария к следующим фазам [41].

Следующий критически важный этап Cyber Kill Chain связан с получением первоначального доступа. В терминологии MITRE ATT&CK ему соответствует тактика Initial Access, при которой нарушитель получает возможность войти в инфраструктуру через скомпрометированные учётные данные, удалённую эксплуатацию сервиса или доставку вредоносного вложения. На этом этапе происходит переход от внешнего сетевого воздействия к выполнению действий непосредственно внутри операционной системы или прикладной среды, развернутой в распределённой инфраструктуре. Данный этап является обязательным для реализации любой сетевой атаки, поскольку без установления первичного канала доступа невозможна дальнейшая реализация последующих этапов компрометации и закрепления в системе [42].

Реализация техник Initial Access может существенно варьироваться в зависимости от используемой злоумышленником стратегии и архитектуры целевой системы. В ряде случаев используется эксплуатация уязвимостей сетевых сервисов и веб-приложений, взаимодействующих через стандартные протоколы прикладного уровня, в других сценариях может быть злоупотребление легитимными учётными данными в рамках механизмов аутентификации операционной системы или корпоративных сервисов. Несмотря на различие инструментов и методов, все варианты данной тактики приводят к формированию аномальных событий в журналах безопасности, включая нестандартные сетевые подключения, ошибки аутентификации, нетипичные веб-запросы и запуск процессов, не соответствующих профилю нормального поведения системы [42]. Таким образом, перекрытие тактики Initial Access позволяет блокировать основной входной вектор атаки ещё до этапов закрепления и эскалации привилегий.

После первоначального доступа нарушитель переходит к выполнению кода (Execution) и повышению привилегий (Privilege Escalation). В модели Cyber Kill Chain это соответствует переходу от первичного проникновения к управлению скомпрометированной средой, а в MITRE ATT&CK связано с запуском вредоносных скриптов, использованием планировщика задач, инжектированием

кода и применением легитимных административных утилит. На практике это означает выполнение команд на целевом узле, развёртывание дополнительных компонентов и получение прав, необходимых для отключения средств защиты, доступа к учётным данным и дальнейшего перемещения по инфраструктуре [43]. На данном этапе происходит запуск команд и вредоносных процессов в операционной системе, а также расширение уровня доступа в локальной или доменной инфраструктуре.

Выполнение команд является базовым механизмом управления атакой в операционной системе. Оно обеспечивает загрузку дополнительных полезных нагрузок, изменение конфигурации системных и сетевых служб, а также подготовку условий для последующего перемещения внутри распределённой инфраструктуры. Повышение привилегий является критически важным этапом, поскольку большинство целевых ресурсов, включая системные файлы, сетевые настройки и параметры безопасности приложений, недоступны в рамках ограниченного пользовательского контекста, характерного для начальной компрометации.

Данные тактики важны тем, что они представляют собой точку перехода от локального инцидента к системной компрометации, а их техники, как правило, имеют чёткие поведенческие признаки, отражающиеся в аудите команд и системных вызовов.

Дальнейшие этапы Cyber Kill Chain направлены на обеспечение устойчивости присутствия злоумышленника в скомпрометированной системе и расширение масштаба воздействия атаки. Тактики уклонения от механизмов защиты (Defense Evasion) и закрепления (Persistence) непосредственно связаны с поддержанием длительного несанкционированного доступа в операционной системе и распределённой сетевой инфраструктуре, повышая вероятность того, что атака будет оставаться незамеченной в течение продолжительного времени [44]. В отсутствие механизмов закрепления компрометация носит кратковременный и эпизодический характер, легко устраняется средствами реагирования и восстановления, поэтому в рамках целевых сетевых атак данные действия практически всегда являются обязательным компонентом цепочки атаки.

Несмотря на применение техник маскировки, модификации механизмов автозапуска, изменения конфигураций безопасности операционной системы и корректировки параметров аудита, подобные действия практически неизбежно формируют следы в журналах событий ОС, сетевых сервисов и прикладных приложений. Эти артефакты представляют собой важные индикаторы компрометации, поскольку отражают нетипичное изменение состояния программной среды и нарушение стандартных политик взаимодействия компонентов системы, что особенно заметно в распределённых и облачных инфраструктурах.

Этапы разведки внутренней инфраструктуры (Internal Reconnaissance) и бокового перемещения (Lateral Movement) соответствуют расширенной интерпретации Cyber Kill Chain и направлены на исследование внутренней сетевой архитектуры и распространение атаки на другие узлы распределённой системы. Их основная цель заключается в увеличении охвата компрометации и достижении более значимых ресурсов внутри корпоративной или облачной среды [45]. Даже при использовании легитимных административных инструментов операционной системы и сетевых сервисов такие действия формируют аномальные последовательности событий, отражающие отклонения от типового поведения пользователей, процессов и сервисов в инфраструктуре.

Завершающие этапы Cyber Kill Chain связаны со сбором и эксфильтрацией данных, управлением скомпрометированными компонентами и непосредственным воздействием на целевые системы. Эти тактики отражают достижение конечной цели атаки и, как правило, наблюдаются в большинстве успешных инцидентов информационной безопасности. Даже при скрытой реализации каналов управления и контроля (C2), включая использование сетевых протоколов прикладного уровня или распределённых инфраструктур, сопутствующие действия (подготовка рабочей среды, выполнение вспомогательных команд, изменение конфигураций операционной системы и сетевых служб) формируют достаточный аналитический контекст для выявления атаки как единого многоэтапного процесса [46].

Таким образом, каждая тактика важна как отдельно, так и в совокупности с остальными, как функциональный элемент общей цепочки атаки. В совокупности

они образуют полное покрытие Cyber Kill Chain: от первых подготовительных действий до финального воздействия. Поскольку злоумышленник не может достичь своей цели, минуя все эти этапы, перекрытие соответствующих тактик обеспечивает обнаружение большинства реальных атак независимо от используемых инструментов и конкретных техник. Именно поэтому анализ событий в разрезе тактик позволяет закрыть практически все возможные пути обнаружения атак и обеспечивает устойчивость системы к эволюции угроз.

1.4 Учетные записи

Учетные записи, как пользовательские, так и технические, являются неотъемлемой частью большинства современных информационных систем. Они предоставляют своим владельцам, будь то люди или сервисы, возможность аутентификации в системе или одном из ее компонентов. Наиболее распространённым подходом к подтверждению подлинности субъекта является использование фактора знания, прежде всего пароля, который проверяется в процессе аутентификации при обращении к системным или сетевым сервисам [8]. Такой метод широко применяется в операционных системах, корпоративных сетях и распределённых информационных системах благодаря своей относительной простоте реализации, низкой вычислительной стоимости и удобству эксплуатации по сравнению с альтернативными механизмами.

Альтернативные подходы, основанные на факторах владения (аппаратные и программные токены, смарт-карты, цифровые удостоверения) или факторах свойств (отпечатки пальцев, голос, походка), требуют более сложной инфраструктуры интеграции с прикладными системами и сетевыми сервисами, а также дополнительных механизмов защиты и синхронизации в распределённых средах. В результате парольная аутентификация остаётся наиболее распространенным механизмом в большинстве операционных систем и сетевых приложений, несмотря на её известные уязвимости.

С точки зрения информационной безопасности, учетные записи могут представлять значительную угрозу для системы и инфраструктуры. Поэтому крайне важно обеспечить надлежащую защиту от их компрометации. С этой целью

администраторы информационных систем внедряют различные парольные политики, устанавливающие минимальные требования к паролям для обеспечения их устойчивости к атакам, направленным на их компрометацию различными методами [47].

Это делает учетные записи наиболее информативным источником для обнаружения атакующих действий. Компрометация учетной записи, даже не обладающей изначально расширенными привилегиями, зачастую является достаточным условием для перехода к следующим стадиям атаки, включая повышение привилегий, боковое перемещение и закрепление в системе.

В рамках жизненного цикла атаки учетные записи играют ключевую роль практически на всех этапах, начиная с первичного доступа до финальной стадии. Независимо от используемого вектора злоумышленник стремится либо немедленно получить учетные данные, либо создать новую учетную запись, маскируя ее под легитимную. На последующих этапах именно учетные записи используются для легализации присутствия в системе: выполнение команд от имени существующих пользователей, запуск сервисов с сохранением контекста аутентификации и использование штатных механизмов администрирования позволяют атакующему минимизировать вероятность обнаружения. Таким образом, анализ активности учетных записей дает возможность выявлять атаки не по отдельным техническим артефактам, а по смысловому нарушению нормального поведения идентичностей внутри системы [8].

Особое значение в современных информационно-вычислительных и распределённых программных системах приобретают учётные данные, используемые в автоматизированных сценариях взаимодействия сервисов и приложений. Сервисные и технические учётные записи, как правило, применяются для обеспечения межсервисной аутентификации в операционных системах, облачных платформах и распределённых инфраструктурах обработки данных. Практика эксплуатации таких систем показывает, что подобные учётные записи часто обладают расширенными привилегиями в рамках локальных и сетевых политик доступа, при этом уровень их защиты нередко оказывается ниже по сравнению с пользовательскими аккаунтами, поскольку основной акцент делается

на непрерывность работы программных компонентов и автоматизированных процессов, а не на интерактивную аутентификацию [48].

Подобная специфика создаёт повышенные риски в контексте информационной безопасности, особенно при использовании слабых, предсказуемых или повторно применяемых паролей. Дополнительным фактором уязвимости является хранение учётных данных в открытом виде в конфигурационных файлах, скриптах автоматизации, переменных окружения, что существенно повышает риск их компрометации. В этом контексте учетные записи становятся целью атаки, позволяя злоумышленнику быстро расширять контроль над системой без необходимости эксплуатации сложных уязвимостей.

По этой причине учетные записи обособляются в отдельный аналитический слой: компрометация аккаунта отражается в событиях системы иначе, чем эксплуатация программных ошибок, но при этом оставляет устойчивые поведенческие следы. Анализ команд с явным указанием учетных данных, нетипичных сценариев аутентификации и использования паролей позволяет выявлять потенциально уязвимые учетные записи еще до того, как они будут использованы злоумышленником в атаке. Такой подход дополняет классические механизмы контроля паролей и политики аутентификации, ориентированные в основном на момент создания или изменения пароля, и переносит фокус на фактическое использование учетных данных в инфраструктуре.

С точки зрения комплексной защиты именно учетные записи обеспечивают перекрытие большинства сценариев атак, поскольку практически любая целенаправленная атака в конечном итоге сводится к злоупотреблению легитимными учетными данными. Независимо от того, идет ли речь о внутреннем нарушителе, внешнем атакующем или вредоносном программном обеспечении, взаимодействие с системой осуществляется через механизмы аутентификации и авторизации. Это согласуется с современными рекомендациями по управлению цифровыми идентичностями, в том числе отраженными в документах NIST 800-63B [49], где учетные данные рассматриваются как основной объект защиты в операционных системах. В результате выделение учетных записей в отдельный раздел и их углубленный анализ позволяют не только повысить точность

обнаружения атак, но и обеспечить сквозное покрытие всего спектра угроз, связанных с несанкционированным доступом, закреплением и развитием атаки внутри системы.

В то же время методы машинного обучения [6,7,50,51] активно разрабатываются и внедряются в сфере информационной безопасности. Множество работ посвящены разработке методов машинного обучения для классификации паролей на простые или надежные. Например, Sarkar S. и Nandan M. [52] смоделировали прогнозирование надежности пароля как задачу классификации, используя несколько алгоритмов машинного обучения с учителем, рассмотрев два новых алгоритма, XGBoost и многослойный перцептрон. Farooq U. [53] рассмотрел задачу классификации паролей веб-приложений в режиме реального времени.

Авторы [54] использовали алгоритм RoBERTa для прогнозирования сложности паролей, применяя набор данных текстовых описаний. Они обучили две модели и оценили их производительность, используя матрицы ошибок и показатели точности для трех категорий классификации. В исследовании [55] представлен новый набор данных о надежности паролей, содержащий данные на международных и литовских языках. Набор включал шесть часто используемых характеристик паролей и тридцать шесть показателей сходства для каждой записи, полученных из девяти словарей и четырех мер сходства. Предложенное решение для прогнозирования надежности паролей по пяти классам достигло точности 77%. Учитывая сложность точности литовского языка, достигнутый результат был адекватным, поскольку интеллектуальные инструменты для взлома паролей, специфичные для литовского языка, пока еще не получили широкого распространения.

В [56] авторы проанализировали рекомендации и требования к паролям, изложенные в международных и национальных стандартах, изучив возможность реализации верификаторов надежности паролей в различных операционных системах. Были представлены результаты эксперимента, проведенного на существующем наборе данных паролей, помеченных уровнями их надежности, и производительность оценивалась с использованием макро-F1-меры.

Однако не совсем ясно, как обнаружить пароль, который является простым, но в тоже время соответствует условиям заданной политики системы и уже используется в ней. Нами рассмотрена задача обнаружения слабых паролей, уже используемых в операционной системе или приложении, и предложено решение на основе машинного обучения. Рассмотрен случай, когда все пароли, проанализированные с помощью этого метода, были явно обнаружены в системных журналах, которые по-прежнему недоступны злоумышленнику. Из-за этого ограничения предлагаемый метод применим только к системам, которые генерируют журналы, содержащие явные данные аутентификации. Поэтому нам потребовалось разработать единое определение для классификации паролей как простых и надежных и создать технологии машинного обучения для их классификации.

1.4.1 Критерии оценки паролей

Было проведено не мало исследований, посвященных решению проблем информационной безопасности с использованием технологий машинного обучения. Dong Q. и др. [57] отметили, что большинство моделей вероятностной проверки паролей учитывают только традиционную угрозу подбора паролей методом перебора, игнорируя более серьезную угрозу целенаправленного подбора паролей с использованием персональных данных (например, имя, дата рождения, номер телефона). Затем они ввели высокочастотную подстроку в качестве нового грамматического тега в ведущие модели вероятностной проверки паролей TarGuess-I и TarMarkov и предложили модели TarGuess-I-H и TarMarkov-H. Аналогичная проблема рассматривалась в [58,59].

В первом случае авторы предложили TarGuess-I+, улучшенную модель подбора паролей, способную идентифицировать популярные пароли путем создания списка из 300 самых популярных паролей на похожих веб-сайтах и распознавать специальные строки путем извлечения непрерывных символов из сгенерированных пользователем персональных данных.

Во втором случае авторы предложили TarGuess-I+KPX, модифицированную модель подбора паролей с тремя семантическими методами,

включая идентификацию популярных паролей путем создания списков из 300 самых популярных паролей на похожих веб-сайтах, распознавание шаблонов клавиатуры по относительному положению и обнаружение специальных строк путем извлечения непрерывных символов из сгенерированных пользователем персональных данных.

Проблема использования персональной информации при формировании паролей и её влияния на устойчивость механизмов аутентификации рассматривается в работе Cui X. и соавторов [60], выполненной в контексте задач повышения информационной безопасности программных и распределённых информационных систем. Авторы применяют методы интеллектуального анализа данных для выявления связей между конфиденциальной пользовательской информацией и структурой паролей, что относится к задачам машинного обучения и обработки данных в системах защиты информации.

В работе используются алгоритм сегментации структуры пароля и метод двунаправленного сопоставления, позволяющие обнаруживать соответствия между личными атрибутами пользователей и элементами паролей. На основе этого предложен метод оценки надёжности паролей, учитывающий степень использования персональной информации и позволяющий формализовать процесс анализа в рамках моделей оценки риска и классификации в системах контроля доступа.

Xu M. и Han W. [61] рассматривают задачу повышения устойчивости механизмов аутентификации в информационных системах за счёт совершенствования моделей оценки сложности паролей в контексте программных средств контроля доступа. В рамках их работы предложено расширение стандартной модели Password Strength Meter (PSM), ориентированное на применение методов интеллектуального анализа текстовых последовательностей и распознавания структурных шаблонов, что относится к задачам машинного обучения и обработки данных в программных системах безопасности.

Разработанное решение обладает объяснимостью, поскольку позволяет не только оценивать слабость пароля, но и интерпретировать причины его уязвимости на основе выявленных текстовых паттернов. Такой подход может быть

задействован в интеллектуальной системе поддержки принятия решений в области информационной безопасности, направленной на повышение качества генерации учетных данных и снижение риска их компрометации в распределённых информационных системах.

Кроме того, Yuan L. и др. проанализировали шаблоны паролей в утекших наборах данных и провели сравнительное исследование двух доминирующих вероятностных моделей (т. е. модели на основе Маркова и модели на основе вероятностных контекстно-свободных грамматик) и модели PassGAN (которая является репрезентативным методом на основе глубокого обучения) [62]. Важную корреляцию между паролями и профилями их владельцев в социальных сетях установили Ji S. и др. [63], которые разработали первый измеритель надежности паролей, учитывающий социальные профили, а именно SociaLShield.

Кроме того, Tran L. и др. провели всесторонний обзор всех исследований по подбору паролей, представленных в литературе с 1979 по 2022 год, и предложили общую методологическую карту для представления обзора существующих методов [64]. Во-первых, необходимо определить, каким условиям должен соответствовать пароль, чтобы считаться надежным. Точных критериев нет, не говоря уже об их значениях. Требования к паролям не регулируются законом; они ограничены только правилами (внутренними документами конкретной организации), установленными администраторами информационных систем, что объясняет большое разнообразие политик паролей [65].

Первым шагом в защите пароля от компрометации является политика паролей [66], установленная системными администраторами. Она включает в себя несколько конкретных минимальных требований, которым пароль должен соответствовать при первоначальной регистрации учетной записи в системе. Стандартные требования на этом уровне часто включают (в той или иной форме) следующие ограничения:

- минимальная длина – 12 символов;
- обязательное использование специальных символов из перечня разрешенных, например (но не ограничиваясь): !, @, #, $, %,^, &, *, (,), -, _, +, =, [,], {, }, |, ~, ;, :, „, <, >, ., /, ?.;

- обязательное использование цифр (0-9);
- обязательное использование букв в верхнем и нижнем регистрах (a–z, A–Z);
- недопустимо использование словарных слов (имена персонажей, марки автомобилей и др.);
- недопустимо использование персональное информации (имя, фамилия, дата рождения, адрес, номер телефона и др.);
- недопустимо использование распространенных легкоугадываемых простых комбинаций и последовательностей (qwerty, zxzxzx и др.)

Согласно Приказу № 31 Федеральной службы технического и экспортного контроля (ФСТЭК) России от 14 марта 2014 г. «Об утверждении требований к информационной безопасности в автоматизированных системах управления производственными и технологическими процессами на критических объектах, потенциально опасных объектах, а также объектах, представляющих высокий риск для жизни и здоровья человека и окружающей среды», выделяются три класса безопасности автоматизированных систем управления в соответствии с тремя уровнями значимости (критичности) обрабатываемой информации. Gufan K.Y. и др. [67] описали количественную оценку надежности пароля на основе вычислений информационной энтропии, которые представляют собой меру неопределенности, измеряемую в битах и описываемую в общем виде уравнением (1).

$$E = Len \cdot \log_2 M \qquad (1)$$

где E — информационная энтропия, M — размер алфавита (количество символов, которые можно использовать для формирования пароля), а Len — длина пароля.

Если для формирования пароля используется несколько алфавитов, то общий размер алфавита будет равен размеру объединения использованных алфавитов; в частном случае, если алфавиты не пересекаются, общий размер алфавита будет равен сумме размеров использованных алфавитов. В той же работе авторы ввели разделение паролей на пять классов безопасности на основе вычисленных значений энтропии.

Однако в качестве основного документа, содержащего рекомендации по

управлению цифровой идентификацией и аутентификацией, была использована специальная публикация документа NIST 800-63B (NIST) [49]. Ее главная цель – повышение безопасности и надежности систем аутентификации за счет снижения рисков кражи и неправомерного использования учетных данных.

Основные обозначения, используемые для описания признаков паролей и параметров парольных политик, представлены в таблице 1.2.

Таблица 1.2 – Обозначения признаков паролей и параметров парольных политик

Обозначение	Описание
SL	Уровень значимости
E	Энтропия
MinL	Минимальная длина
MaxL	Максимальная длина
RecL	Рекомендованная длина
Len	Длина
U	Символы в верхнем регистре
L	Символы в нижнем регистре
N	Цифры
S	Специальные символы
Seq	Последовательность символов
Rep	Повторение символов
Bio	Биографический элемент
Dic	Словарные слова
Uniq	Уникальные символы
Corp	Пароли, обнаруженные в базах компрометированных паролей
NoCC	Количество символов отличия нового и старого паролей
SpN	Учет количества цифровых символов
Freq	Частота смены алфавита
FCI	Интервал блокировки
NIST	National Institute of Standards and Technology
MaxRC	Максимальное количество повторяющихся символов
MaxCRC	Максимальное количество последовательных повторяющихся символов
MaxFC	Максимальное количество неуспешных аутентификаций
Dist	Среднее расстояние между последовательными символами на раскладке стандартной клавиатуры

Продолжение таблицы 1.2

Обозначение	Описание
AC	Количество задействованных алфавитов
VT	Срок действия
M	Совокупная мощность алфавитов
AL	Разрешено
NOTA	Запрещено
NOTR	Не рекомендуется

В таблице 1.3 представлено сравнение требований к параметрам паролей от таких организаций, как NIST, Microsoft [68], Apple [69], IBM [70] и Cisco [71], с требованиями, установленными в стандарте NIST.

Таблица 1.3 – Сравнение парольных политик

Параметр	NIST [49]	Cisco [71]	Microsoft [68]	IBM [70]	Apple [69]
MinL	8	1	1	1	8
MaxL	64	64			
RecL	12	8	8	8	8
AC	4	4	4	4	4
VT	≤90 day	Forever	NOTR	Forever	Forever
Corp	+	-	+	-	
Bio	+	-	+	+	+
NoCC	3-4	4	NOTR	≥0	
SpN	+	+	+	+	+
M	128	72			
MaxFC	≤100	≥0	≥0	≥0	≥0
MaxRC	+	-	-	+	-
MaxCRC	+	-	-	+	-
FCI	>30 s	>60 s	≥0 s	≥0 s	≥0 s
Dic	+	+	+		+
Rep or Seq	+	-	-	+	-

Стоит отметить, что значения некоторых параметров могут отличаться в зависимости от продукта или версии организации. Также важно отметить, что все рассматриваемые производители позволяют администраторам гибко настраивать многие из перечисленных параметров, тем самым допуская отклонения в лучшую

или худшую сторону (по сравнению со стандартом NIST).

Важно отметить, что политика NIST рекомендует запрещать использование последовательностей символов одного алфавита или повторяющихся символов в качестве паролей (например, «123456» или «aaa333»), но не вводить технические ограничения, которые препятствуют использованию таких комбинаций. Это связано с тем, что такое ограничение может значительно сократить количество вариантов паролей.

Например, предположим, что пользователю разрешено выбирать произвольные символы из следующих четырех алфавитов: заглавные латинские буквы U (мощность $M(U)$ = 26 символов), строчные латинские буквы L (мощность $M(L)$ = 26), цифры N (мощность $M(N)$ = 10) и специальные символы S (мощность $M(S)$ = 10). Таким образом, суммарная степень всех алфавитов M составляет 72 единицы. Если нет предела, указанного выше, то общее количество всех возможных вариантов пароля P вычисляется с использованием уравнения (2):

$$P = M^{Len} \tag{2}$$

Таким образом, для пароля длиной всего четыре символа количество комбинаций составит 72^4 или 26 873 856.

Теперь рассмотрим ситуацию, когда это ограничение введено, и оно не позволяет использовать любые два последовательных символа, принадлежащих к одному и тому же алфавиту. Пусть общая форма пароля одинаковой длины имеет форму $ULNS$. Тогда количество вариантов пароля можно рассчитать с помощью уравнения (3):

$$P' = M \cdot \big(M - M(U)\big) \cdot \big(M - M(L)\big) \cdot \big(M - M(N)\big), \tag{3}$$

или в общем виде:

$$P' = M \cdot \prod_{i=2}^{Len} (M - M_i), \tag{4}$$

где M_i - это мощность алфавита, используемая для образования символа в позиции i.

Таким образом, количество комбинаций в этом случае составляет 9 445 824, что в 2,85 раза меньше, чем в первом случае. Следовательно, введение этого

ограничения помогает злоумышленнику скомпрометировать учетную запись в большей степени, чем защищает от него.

Однако все же рекомендуется учитывать этот параметр паролей при их классификации на потенциально скомпрометированные и устойчивые. Рассмотрим ситуацию, в которой злоумышленник может получить доступ к хеш-значению пароля учетной записи, то есть случай, когда ограничение количества попыток ввода не защищает пароль, поскольку попытки сопоставления будут выполняться вне защищаемой системы. Атака с использованием радужных таблиц или с использованием словарей могут считаться самыми быстрыми методами, поскольку нет необходимости перебирать все возможные комбинации символов, что требует значительно больше времени и вычислительных ресурсов [72]. Поэтому, чтобы минимизировать вероятность подбора пароля такими типами атак и оставить злоумышленнику только возможность провести атаку методом полного перебора, все пароли должны как можно лучше соответствовать критериям случайных последовательностей. Колмогоров А.Н. и Успенский А.В. [73] перечислили три основных свойства случайных последовательностей:

- типичность;
- хаотичность;
- стабильность частоты.

Чтобы доказать, что частота изменений алфавита существенно влияет на классификацию паролей на простые и надежные, мы сосредоточились только на третьем свойстве. Оно заключается в том, что при бесконечно большом количестве испытаний частота случайного события будет сходиться к определенному значению — вероятности этого события. Другими словами, чем больше испытаний проводится, тем точнее оценка вероятности события будет соответствовать фактической вероятности. Однако наиболее распространенным критерием для случайной последовательности является значение «хи-квадрат» χ^2, которое обычно определяется уравнением (5):

$$\chi^2 = \sum_{s=1}^{k} \frac{(Y_s - np_s)^2}{np_s}, \tag{5}$$

где s — определенное наблюдаемое значение, n — количество наблюдений, p_s — вероятность наступления события s, np_s — ожидаемое количество раз, когда произойдет событие s, Y_s — фактическое количество раз, когда произошло событие s, и k — количество различных значений, которые может принимать параметр s.

На рисунке 1.4 представлены несколько примеров простых и надежных паролей, использованных при анализе.

Simple	Strong	Simple	Strong
iLOVEmatt2	Dq5Z4X@)AS	Jagger1978	ZO1OF^mgSW
9473Mickey	Q5^pz!AP(q	Utvolsdt14	QYT!1J54rb
BMW318cise	jm#PBz3otw	Welcome187	v!4RSRKICV
Ching24568	!)$j46&Ye7	Computer12	nl1GX*(hPO

Рисунок 1.4 – Примеры простых и стойких паролей

Далее, пароли из сгенерированного образца рассматривались на основе задействованных алфавитов. Например, пароль «AAbb12@#» можно рассматривать как $UULLNNSS$ в ранее введенной нотации. Затем, в рамках задачи, переменные в уравнении (5) можно интерпретировать следующим образом: s — один из четырех ожидаемых алфавитов, p_s — вероятность появления символа из алфавита s, Y_s — фактическое количество встреченных символов из алфавита s, а произведение np_s соответствует ожидаемому количеству вхождений символов из алфавита s. Значение $k = 4$ — общее количество алфавитов. Переменная $n = 20\,000$ соответствует количеству выполненных испытаний (один пароль на испытание). При формировании списка паролей мы ограничили допустимый набор символов заглавными латинскими буквами U, строчными латинскими буквами L, цифрами N и 10 специальными символами S. Суммарная мощность всех алфавитов M также составляла 72 единицы. Затем, используя выражение, было легко вычислить вероятность появления каждой буквы алфавита:

$$p_s = \frac{M_s}{M}, \tag{6}$$

где M_s - мощность рассматриваемого алфавита s.

С учетом этого, были получены средние результаты вычислений для Y_{simple}, np_{simple} и Y_{strong}, $\mathrm{np}_{\mathrm{strong}}$ для простых и надежных паролей соответственно. В таблице 1.4 представлены результаты тестов на случайность последовательностей для простых и надежных паролей соответственно.

Таблица 1.4 – Результаты тестов на случайность простых и надежных паролей

Параметр	U	L	N	S	χ^2
Y_{simple}	0.1246	0.5701	0.2566	0.0487	7.9796
np_{simple}	0.3600	0.3600	0.1400	0.1400	
Y_{strong}	0.3173	0.3164	0.1833	0.1829	2.1536
$\mathrm{np}_{\mathrm{strong}}$	0.3600	0.3600	0.1400	0.1400	

Тесты показали, что в случае простых паролей наблюдались статистически значимые различия между полученными и ожидаемыми результатами для случайных последовательностей, в то время как для надежных паролей эти показатели, наоборот, были близки. Поскольку в нашем случае число степеней свободы $k - 1 = 3$, то, используя таблицу 1.5 с указанием на ней некоторых процентных пунктов распределения хи-квадрат, значение V' для простых паролей приблизительно соответствовало $p = 95\%$, что указывает на значительные отклонения от случайного поведения, поскольку V примет значение 7,815 только в 5% случаев.

В то же время для надежных паролей параметр χ^2 приблизительно соответствует $p = 50\%$. Это значение нельзя считать значительно малым или значительно большим; следовательно, рассматриваемые в данном случае пароли можно считать случайными относительно этого значения. Действительно, на представленном ранее рисунке 1.4 легко увидеть, что большинство простых паролей состоят из различных слов, фраз и других распространенных последовательностей, в то время как невозможно установить какую-либо связь между символами надежных паролей.

Важно отметить, что, несмотря на нормативные ограничения, введенные политикой паролей, установленные требования часто не проверяются технически (за исключением длины пароля, количества задействованных букв и разрешенных

специальных символов), что дает возможность использовать различные простые комбинации.

Таблица 1.5 – Некоторые процентные пункты распределения χ^2

V	p = 1%	p = 5%	p = 25%	p = 50%	p = 75%	p = 95%	p = 99%
1	0.0002	0.0039	0.1015	0.4549	1.323	3.841	6.635
2	0.021	0.1026	0.5754	1.386	2.773	5.991	9.210
3	0.1148	0.3518	1.213	2.366	4.108	7.815	11.34
4	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
5	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
6	0.8721	1.635	3.455	5.348	7.841	12.59	16.81
7	1.239	2.167	4.255	6.346	9.037	14.07	18.48
8	1.646	2.733	5.071	7.344	10.22	15.51	20.09
9	2.088	3.325	5.899	8.343	11.39	16.92	21.67
10	2.558	3.940	6.737	9.342	12.55	18.31	23.21
11	3.053	4.575	7.584	10.34	13.70	19.68	24.72
12	3.571	5.226	8.438	11.34	14.85	21.03	26.22
15	5.229	7.261	11.04	14.34	18.25	25.00	30.58
20	8.260	10.85	15.45	19.34	23.83	31.41	37.57
30	14.95	18.49	24.48	29.34	34.80	43.77	50.89
50	29.71	34.76	42.94	49.33	56.33	67.50	76.15
> 30	$V + \sqrt{2V}x_p + \frac{2}{3}x_p^2 - \frac{2}{3} + O\left(\frac{1}{\sqrt{V}}\right)$						
$xp = 1$	-2.33	-1.64	-0.674	0.00	0.674	1.64	2.33

На основе вышеизложенного были выявлены несколько наиболее значимых параметров паролей, как покажут дальнейшие эксперименты, совместный анализ которых может помочь классифицировать пароли на надежные и простые. Общий список представлен в таблице 1.6.

Здесь мы добавили еще два параметра. Uniq определяет количество уникальных символов в пароле и необходим в качестве дополнительного критерия классификации для тех же случаев, когда пользователь использует различные

повторяющиеся последовательности (например, «zzzz», «qwqwqw» и т. д.).

Таблица 1.6 – Критерии паролей

Тип	MinL	U	L	N	S	Seq	Rep	Uniq	Freq	Dic	Bio	Corp	Dist
Надежный	14	≥4	≥4	≥3	≥3	≤2	≤2	>0.9	0.3	-	-	-	>4
Простой	10	≥3	≥3	≥2	≥2	≤4	≤3	>0.7	0.18	-	-	-	>3.2

Пользователи также часто придумывают пароли-змейки, то есть значения, содержащие последовательности символов, расположенных рядом друг с другом на раскладке клавиатуры, образуя таким образом «змеиную» сигнатуру (например, «qwerty», «zxcasd» и другие) [8]. Для противодействия этому мы ввели параметр Dist, который последовательно вычисляет расстояние между соседними символами пароля и вычисляет среднее значение из полученных значений.

Важное замечание: все параметры, перечисленные в таблице 1.6, должны использоваться одновременно для правильного решения о надежности пароля. Это продемонстрировано в последующих экспериментах. Полученные пороговые значения для каждого уровня надежности также были дополнительно рассчитаны.

1.4.2 Анализ методов поиска простых паролей

Процесс аутентификации включает хранение учетных записей пользователей или служб в базах данных операционных систем, установленных на конечных узлах, предназначенных для входа в систему. Поэтому для защиты паролей от компрометации при несанкционированном доступе злоумышленника выполняется хеширование паролей. Этот механизм предотвращает захват учетных записей злоумышленником для дальнейшего перемещения внутри системы или повышения привилегий на атакованном узле [74]. Однако из-за этого системные администраторы также не могут извлекать пароли в открытом виде для проведения ручного или автоматизированного анализа паролей с целью проверки их соответствия политике паролей.

Для частичного решения этой проблемы они прибегают к методу, который включает в себя выгрузку хеш-значений и установление соответствующих значений паролей из общедоступных баз данных. Организации часто используют свои пароли по умолчанию, которые обычно не встречаются в таких таблицах.

Поэтому администраторы могут самостоятельно вычислять хеш-функции на основе известных простых паролей и паролей по умолчанию, чтобы искать их в своих базах данных. Это позволяет идентифицировать учетные записи с паролями, которые потенциально могут быть скомпрометированы злоумышленниками. Однако этот метод имеет несколько существенных недостатков, среди которых можно выделить следующие:

- предположим, информационная система содержит хосты, которые не находятся под единым управлением какого-либо сервера (например, контроллера домена) или на которых используются локальные учетные записи. В этом случае администраторам придется выгрузить базы данных со всех таких конечных точек, чтобы получить значения хэш-функций всех паролей. В случае, если организация имеет большое количество серверов и автоматизированных рабочих мест, этот процесс может занять значительное количество времени;
- хэш-функции, используемые для преобразования паролей, должны быть надежными [75]. Следовательно, количество коллизий должно быть сведено к минимуму, и не должно быть возможности установить связь между хэшированными данными и результатами преобразований. Это приводит к тому, что такой простой пароль, например, «password», немного измененный на «password_1379», может быть не найден, хотя также остается ненадежным;
- ситуация усугубляется, если используется механизм добавления «соли» [76] для дополнительной защиты учетных записей при их хранении. В этом случае значения хеш-функций без и с их смешиванием значительно различаются, что существенно затрудняет поиск ненадежных паролей или вообще не позволяет использовать этот метод.

Методы подбора паролей во всех их разновидностях используются злоумышленниками и аналитиками безопасности для взлома учетных записей и дальнейшего продвижения или закрепления в системе. Как правило, они включают в себя перебор каждого символа (или некоторых из них, если точно известно о наличии определенных символов в конкретной позиции) пароля до тех пор, пока

не будет найдена истинная комбинация. Специалисты по информационной безопасности могут использовать этот метод для выявления ненадежных паролей, если у них есть определенные навыки работы с соответствующими инструментами. Ситуация усложняется пропорционально увеличению количества учетных записей в системе. Перебор всех возможных комбинаций для одного пользователя может занять значительное время, даже если используется оптимизация на основе словарей паролей или их масок. Поэтому для крупных организаций, где количество учетных записей исчисляется тысячами, использование этого метода может быть неэффективным как с точки зрения времени, так и количества найденных ненадежных паролей.

Метод, называемый «спреем паролей», ускоряет поиск. Он предполагает одновременное использование одной и той же комбинации для нескольких учетных записей. Это значительно сокращает время, необходимое для перебора всех возможных вариантов [77]. Однако этот метод не гарантирует обнаружение всех желаемых простых паролей и имеет недостатки, присущие описанным ранее методам.

Также важно отметить, если система имеет защиту от атак методом полного перебора, например, на установлены ограничения на количество входов в систему под одной учетной записью или с одного устройства, тогда это ограничение значительно усложняет процесс поиска, делая его практически неэффективным.

В [78] исследовалась эффективность словарных атак и атак методом перебора, с использованием параллельных реализаций на Python, C++ и Hashcat. Результаты показали, что среднее время выполнения словарной атаки было примерно в 50 раз быстрее, чем атака методом перебора. Кроме того, в [79] были представлены статистические сравнения этих методов, подчеркивающие эффективность словарных атак. Словарные атаки продемонстрировали в 1,5 раза более высокий процент успеха при взломе паролей, достигаемый при более коротком времени выполнения. Следовательно, словарные атаки быстрее и точнее, чем атаки методом перебора.

1.5 Выводы по первой главе

В первой главе показано, что традиционные сигнатурные и эвристические системы обнаружения атак, а также классические подходы анализа журналов, недостаточно эффективны против современных многоэтапных APT-атак и обфусцированных полезных нагрузок, особенно в условиях ограниченных вычислительных ресурсов. Проведен обзор современных решений на основе глубоких нейронных сетей, трансформеров, графовых моделей и контрастивного обучения для анализа командных строк, системных событий, веб-журналов и исполняемых файлов, продемонстрировавший высокую точность этих методов, но также их высокую ресурсозатратность и зависимость от больших размеченных датасетов.

Отдельно обоснована актуальность задач классификации команд и событий SIEM в привязке к тактикам MITRE ATT&CK и этапам Cyber Kill Chain, что позволяет обнаруживать отдельные шаги APT-атак на ранних стадиях и рассматривать журналы как основной источник признаков вредоносной активности. Показано, что попытки моделировать сложные многоэтапные сценарии с использованием графовых нейронных сетей (например, Digraph-MMB, PWA-GAT) дают лучшие показатели качества, но практически неприменимы на пограничных устройствах и других маломощных средах.

На этом фоне сформулировано ключевое научно-практическое противоречие: с одной стороны, необходима высокая точность и способность обрабатывать сложные и модифицированные атаки, с другой стороны требуется ресурсоэффективность и возможность развертывания на ограниченном оборудовании. В качестве направления решения предложен подход на основе оптимизированных трансформерных моделей семейства BERT с использованием дистилляции знаний и квантизации, что позволяет отказаться от вычислительно тяжелых графовых моделей при сохранении сопоставимых показателей качества и обеспечить масштабируемость для различных сред и инфраструктур.

Показано, что современные методы на основе глубоких нейронных сетей, трансформерных архитектур и графовых моделей обеспечивают высокое качество классификации и выявления сложных сценариев вредоносной активности, однако

характеризуются значительными требованиями к вычислительным ресурсам, памяти и объёму обучающих данных. Установлено, что существующие подходы к ускорению вычислений преимущественно основаны на архитектурном упрощении моделей, применении дистилляции знаний и высокопроизводительных вычислительных платформах. Вопросы повышения производительности потокового вывода событий безопасности в средах с ограниченными ресурсами без внесения изменений в архитектуру используемой модели и без снижения качества классификации исследованы ограниченно.

Дополнительно в главе проанализированы работы по оценке и классификации надежности паролей, включая модели на основе RoBERTa, вероятностные и грамматические подходы TarGuess, TarMarkov и их модификации, а также методы учета личной информации и поведенческих факторов пользователей. На основе этого анализа сформулирована отдельная прикладная задача выявления слабых паролей, регистрируемых в операционных системах или приложениях, и обоснована целесообразность применения технологий машинного обучения для их автоматической классификации.

В совокупности результаты главы формируют теоретическую и методологическую основу диссертационного исследования, уточняют место предлагаемого решения в существующем научном ландшафте и позволяют четко сформулировать требования к разрабатываемой системе обнаружения вредоносной активности и слабых аутентификационных данных на основе журналов событий и трансформерных моделей.

Глава 2 Разработка методов и алгоритмов обнаружения несанкционированного вторжения в операционные системы с использованием технологии машинного обучения

2.1 Разработка метода и алгоритма раннего обнаружения атак

На рисунке 2.1 представлена структурная схема предлагаемой интеллектуальной системы.

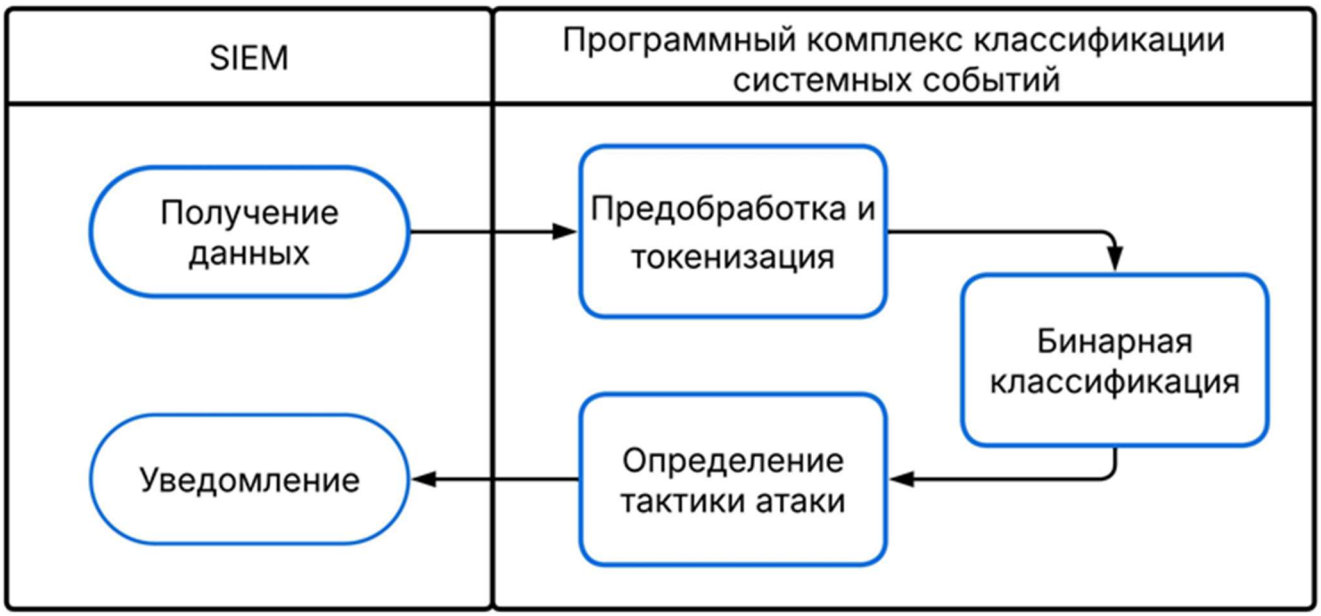


Рисунок 2.1 – Структурная схема предлагаемой интеллектуальной системы

Разрабатываемый программный комплекс реализует последовательную конвейерную архитектуру обработки событий безопасности, характерную для современных распределённых систем мониторинга и анализа информационных потоков. На первом этапе осуществляется централизованный сбор исходных событий из системных журналов операционной системы и прикладных сервисов. В качестве источников данных рассматриваются журналы различного уровня, содержащие информацию о запуске и завершении процессов, доступе к файловой системе, изменении прав и привилегий, а также других действиях, связанных с поведением пользователей и сервисных учетных записей в Linux-среде.

Модуль ввода данных реализует функции агрегации и нормализации событий из разнородных источников, обеспечивая их приведение к единому формату для последующей обработки. В рамках архитектуры SIEM-подобных систем данный компонент выступает как точка первичной консолидации событий

безопасности, после чего сформированный поток данных передается в последующие модули анализа, где применяются методы машинного обучения и интеллектуальной обработки для выявления аномалий и потенциально вредоносной активности.

Второй этап конвейера обработки реализует процедуру токенизации, в рамках которой каждая запись системного или прикладного журнала преобразуется в структурированную последовательность токенов с использованием предварительно обученного токенизатора модели BERT. На этом шаге события предварительно обрабатываются и нормализуются: удаляются избыточные поля и атрибуты, специфичные для окружения, тогда как семантические элементы, такие как домены, пути и IP-адреса, сохраняются в нормализованной форме.

В результате формируется согласованное и структурированное представление данных, уменьшающее уровень шума и повышающее информативность входного потока для последующих этапов анализа. Такая подготовка данных обеспечивает более устойчивую работу моделей машинного обучения, включая трансформерные архитектуры, и повышает качество выявления аномалий в событиях SIEM и журналах операционных систем.

Далее в конвейере обработки применяется бинарный классификатор, реализующий задачу первичной категоризации событий безопасности, поступающих после этапа токенизации и нормализации. В рамках интеллектуальной системы анализа логов данный компонент выполняет функцию предварительного разделения потока событий на два класса: обычные системные операции, характерные для штатного функционирования операционной системы и прикладных сервисов, и потенциально вредоносные действия, которые могут быть связаны с сетевыми атаками, эксплуатацией уязвимостей или несанкционированным доступом.

Такой механизм выступает в роли фильтрационного уровня в архитектуре программного комплекса, снижая нагрузку на последующие этапы глубокой аналитической обработки и позволяя сосредоточить вычислительные ресурсы на событиях с повышенной степенью риска. С точки зрения организации

распределённой обработки данных, это соответствует принципам раннего отсечения малозначимых потоков телеметрии в SIEM-подобных системах, что особенно важно при работе с большими объёмами журналов операционных систем и сетевых сервисов.

Если классификатор определяет событие как вредоносное, оно передается тактическому классификатору, который присваивает одну или несколько тактик MITRE ATT&CK. Это обеспечивает не только обнаружение атак, но и контекстуальное понимание стадии атаки в рамках цепочки действий противника.

Наконец, результаты обоих классификаторов агрегируются в модуле генерации оповещений, который формирует структурированные отчеты или отправляет уведомления в систему мониторинга. Эти оповещения содержат исходный лог, его бинарную метку, обнаруженную тактику и коэффициенты уверенности.

Предлагаемая архитектура использует два классификатора: бинарный классификатор для обнаружения вредоносного содержимого в событии и многометочный классификатор для определения тактики атаки обнаруженного вредоносного события в соответствии с фреймворком MITRE ATT&CK.

На первом этапе токенизация применяется для получения полного текстового представления события, объединяя сообщение аудита и связанные параметры. На втором этапе используется упрощенный процесс токенизации на уровне предобработанных логов, где каждый пример соответствует одному или нескольким бинарным индикаторам тактик. В обоих случаях токенизация обеспечивает согласованное представление данных и служит связующим звеном между текстовыми событиями и вводами нейронной модели, обеспечивая эффективный анализ контекста и семантических зависимостей в логах [11].

Оба классификатора реализованы с использованием дистилляции знаний, при этом токенизация выполнялась как для модели-учителя (BERT), так и для модели-студента (TinyBERT). Чтобы обеспечить идентичность входных представлений, оба токенизатора были инициализированы одним и тем же словарем.

После завершения всех этапов обнаруженные вредоносные события могут

быть переданы специалистам по информационной безопасности для дальнейшего анализа и реализации соответствующих мер противодействия.

2.1.1 Форматирование данных

В качестве основного источника данных был использован набор Linux-APT-Dataset-2024 [80]. Он был собран в тестовой среде, эмулирующей реальную инфраструктуру, что обеспечило захват репрезентативных системных действий и логов. Вредоносное поведение было сымитировано автором набора данных с использованием специализированного инструмента APT (Advanced Persistent Threat). Смоделированные тактики включают полезные нагрузки для повышения привилегий Linux, недавние утечки уязвимостей, эмуляцию клавиатурного шпиона, а также кампании APT, такие как APT41, APT28, APT29 и Turla. Все типы событий были агрегированы в единый набор данных системой безопасности Wazuh [81].

Авторы работы [82] провели исследование, сравнивающее этот набор данных с другими аналогичными наборами данных, используемыми в анализе различных APT-стратегий. Их результаты представлены в таблице 2.1.

Таблица 2.1 – Сравнение наборов данных с точки зрения стратегий APT-атак.

Dataset	Reconnaissance	Persistence	Lateral Movement	Data Exfiltration	Cover Up
Darpa [85]	+	+			
Streamspot [86]	+	+	+		
Provsec [87]	+	+		+	
Atlas [88]	+	+	+	+	
Wget [89]	+	+		+	
SC-2 [89]	+	+		+	
DAPT2020 [90]	+	+	+	+	
Unraveled [81]	+	+	+	+	+
Linux-APT-Dataset-2024 [80]	+	+	+	+	+

Тактики, включенные в набор Linux-MMA-Dataset-2024, являются ключевыми для задачи обнаружения и классификации вредоносных команд,

поскольку охватывают полный спектр атаки – от начальной фазы проникновения до финальной стадии реализации. На ранних этапах компрометации система должна обнаруживать попытки несанкционированного доступа, включая рекогносцировку целевой инфраструктуры, получение начального доступа и выполнение вредоносного кода [83]. Обнаружение повышения привилегий играет критически важную роль, поскольку эта фаза знаменует собой переход от ограниченного доступа к уровню контроля, достаточному для глубокого проникновения в систему и проведения более разрушительных операций [84].

В промежуточных фазах атаки система должна отслеживать попытки закрепления в системе, перемещения между сегментами сети и сбор информации о среде для планирования дальнейших действий. На поздних этапах, когда злоумышленник уже обеспечил себе положение, важно выявлять извлечение учетных данных, установление удаленных каналов управления, сбор данных и попытки нанести ущерб системам или инфраструктуре.

Такой многоуровневый подход к классификации вредоносных событий позволяет модели не только выявлять отдельные инциденты, но и понимать стратегический контекст атак, что особенно важно для эффективного мониторинга безопасности в реальном времени на устройствах с ограниченными ресурсами [11].

В результатах, представленных сравнительной таблицей 2.1, видно, что наиболее разнообразные наборы данных с точки зрения количества тактик атак – Unraveled и Linux-MMA-Dataset-2024. Несмотря на растущую популярность первого, из-за разнообразия сценариев атак он не используется в системах обнаружения вторжений и не считается эталонным набором данных. Важно также отметить, что события, связанные с выполнением команд, описываются лишь небольшим набором атрибутов – идентификатором процесса, рабочей директорией процесса, идентификатором терминала, именем файла и идентификатором пользователя – которые сами по себе недостаточны для решения рассматриваемой задачи. По этой причине в своей работе выбран Linux-MMA-Dataset-2024.

Набор данных доступен в двух форматах: «combine.csv» и «Processed Version.xlsx», каждый из которых содержит все необходимые поля для данной задачи. В данной работе использовалась последняя версия.

Основным источником критериев вредоносности является поле «FullLog», содержащее системные события, указывающие на попытку (успешную или неуспешную) вредоносного действия. Другие важные поля включают «Malicious/General», которое содержит метку вредоносной (value 1) или доброкачественной (value 0) активности, и поле «label», в котором содержится имя сценария атаки и «rule.mitre.tactic», включающее список всех тактик MITRE ATT&CK, связанных с событием (если оно описывает вредоносную активность), а также значение «tactic» для удобства. Используя эти параметры, выполнена дополнительную фильтрацию для получения списка уникальных событий. Небольшое число строк было отброшено из-за пустых значений в полях «FullLog» или «label».

Из-за потенциальной утечки данных между различными машинами набор данных был разделен на пять подмножеств на уровне хостов. Поскольку набор данных не был сбалансирован как по количеству бинарных меток вредоносности, так и по количеству тактик атак вредоносных событий, была применена стратифицированная выборка, чтобы обеспечить пропорциональное представление каждой тактики как в обучающей, так и в тестовой выборках.

После получения исходных данных начинается этап предварительной обработки. Он включает преобразование текста в нижний регистр, удаление лишних пробельных символов (whitespace collapse), обработку ASCII-символов и фильтрацию нерелевантных или семантически дублирующихся параметров и переменных.

Во время обработки ASCII-символов применяется нормализация в форме NFKC (Normalization Form KC). Она включает два процесса: Compatibility Decomposition, при котором каждый символ заменяется совместимым форматом, и Canonical Composition, при котором возможные последовательности символов снова объединяются в составленные формы [91].

На этом этапе блоки с типом = NORMAL, типом = PROCITLE, параметрами msg = audit(), AUID, UID, GID, EUID, SUID, FSUID, EGID, SGID, FSGID, comm=, и exe= удаляются. Идентификаторы процессов, IP-адреса, имена хостов и домены заменяются статическими токенами вида <pid>, <ip>, <domain> и <hash>. Пути к

файлам обрабатываются отдельно: во время обучения кодирование меток применяется к каждому пути в формате <path_N>, где N – его уникальный идентификатор. Таким образом, создается отображение (mapping), которое затем используется повторно во время вывода модели. Если во время вывода модель встречает путь, которого нет в словаре, он преобразуется таким же образом и добавляется в словарь. Описанная процедура предварительной обработки применялась как во время обучения, так и во время тестирования. Этот процесс позволяет уменьшить длину большинства событий более чем вдвое, не теряя основного семантического содержания. В таблице 2.2 представлено статистическое сравнение количества токенов в строке набора данных, с обработкой и без предложенного метода предварительной обработки.

Таблица 2.2 – Влияние предварительной обработки на количество токенов.

Параметр токена	Без предварительной обработки	С предварительной обработкой
Средняя длина	579.3	299.4
Медиана	653.0	327.0
Максимальная длина	4010	3718
90-й процентиль	718.0	362.0
95-й процентиль	729.0	383.0
99-й процентиль	815.0	428.3

Поскольку события представляют собой неструктурированный текст, содержащий большое количество управляющих символов, параметров и идентификаторов, их необходимо преобразовать в унифицированное представление [9]. С этой целью токенизация выполнялась с использованием стандартных токенизаторов семейства BERT, реализованных в библиотеке HuggingFace Transformers (использовалась модель bert-base-uncased). Токенизация производилась по методу WordPiece, при котором текст разбивается на подсловные единицы (токены) на основе словарей [92] размером около 30 000 единиц. Каждый лог стандартизировался до фиксированной длины и усекался до 512 токенов согласно архитектурным ограничениям BERT. Как показано в таблице 2.2, длины 512 токенов достаточно для покрытия более 99% событий. Тем не менее набор данных содержит множество командных аргументов, критичных для анализа

событий, что может привести к превышению максимального лимита токенов. В рамках данной задачи такие строки усекались до значения 512 токенов, что не повлияло на итоговый результат. Модели также смогли классифицировать эти события как вредоносные, а также успешно определили их тактики атак. Это объясняется тем, что в оставшихся после усечения данных также присутствуют характерные данным тактикам атак паттерны.

2.1.2 Бинарная классификация вредоносной активности

Следующим этапом разработанного метода является бинарная классификация системных событий, направленная на разделение входного потока на нормальные и потенциально вредоносные действия.

Использование отдельного бинарного классификатора позволяет реализовать двухэтапную схему анализа, при которой дальнейшая обработка выполняется только для событий, содержащих признаки вредоносной активности. Такой подход уменьшает вычислительную нагрузку на последующие этапы анализа и позволяет сосредоточить ресурсы на наиболее значимых событиях [12].

В качестве основы классификатора используются трансформерные модели семейства BERT, способные учитывать контекст возникновения события и выявлять скрытые зависимости между его элементами. При этом разработанный метод ориентирован не только на обеспечение высокой точности классификации, но и на возможность последующего использования в системах мониторинга информационной безопасности, функционирующих в условиях ограниченных вычислительных ресурсов.

Для решения данной задачи в составе метода применяется перенос знаний от более сложной модели к компактной архитектуре посредством процедуры дистилляции. Использование данного подхода позволяет сохранить способность модели выявлять закономерности, характерные для вредоносной активности, одновременно уменьшая вычислительную сложность последующей обработки событий.

Дистилляция знаний (Knowledge Distillation) – это метод машинного обучения, заключающийся в передаче знаний от большой, сложной модели

(teacher) к меньшей, более эффективной модели (student) [93]. Вместо использования жестких one-hot меток используется распределение вероятностей (soft targets), позволяющее модели-ученику получать дополнительную информацию об уверенности предсказаний модели-учителя [94,95].

Ключевым параметром, контролирующим мягкость предсказаний учителя и влияющим на процесс обучения студента, является температура (t).

Во время дистилляции процесс обучения студента сводится к минимизации комбинированной функции потерь L [96], как показано в формуле (7).

$$L = a \cdot t^2 \cdot L_{KL} + (1 - a) \cdot L_{CE}, \quad (7)$$

где L_{KL} - это расхождение Кульбака–Лейблера между предсказаниями студента и учителя, L_{CE} - функция потерь кросс-энтропии между предсказаниями студента и истинными метками, и a - коэффициент весов.

Предлагаемый метод предполагает использование наиболее распространенной схемы обучения, при которой предварительно обученная модель-учитель остается замороженной в ходе обучения студента [97].

Результатом работы бинарного классификатора является присвоение событию одной из двух меток: нормальное событие или вредоносное, которые далее передаются на следующий этап анализа.

2.1.3 Многометочная классификация обнаруженной вредоносной активности

После выявления вредоносной активности выполняется определение тактики атаки в соответствии с матрицей MITRE ATT&CK.

Необходимость данного этапа обусловлена тем, что факт обнаружения вредоносного события сам по себе не позволяет определить его роль в общей цепочке действий злоумышленника. Для принятия решений специалистам центра мониторинга информационной безопасности требуется информация не только о наличии атаки, но и о ее стадии, целях и предполагаемом развитии.

В разработанном методе для решения данной задачи используется отдельный многометочный классификатор [13], функционирующий независимо от бинарного классификатора. На его вход поступают только события, ранее

отнесенные к вредоносному классу.

Особенностью рассматриваемой задачи является возможность одновременного соответствия одного события нескольким тактикам MITRE ATT&CK. Например, одно действие злоумышленника может одновременно содержать признаки первоначального доступа, повышения привилегий и уклонения от средств защиты. В связи с этим в методе используется многометочная схема классификации, допускающая назначение нескольким тактикам одному событию.

Для формирования тактического описания атаки применяется трансформерная модель, использующая контекстное представление события, сформированное на предыдущих этапах обработки. Полученные результаты позволяют определить место события в цепочке реализации атаки и сформировать дополнительный уровень интерпретации обнаруженной вредоносной активности.

Таким образом, разработанный метод обеспечивает не только обнаружение вредоносных действий в потоке системных событий операционных систем семейства Linux, но и их последующую классификацию по тактикам MITRE ATT&CK, что позволяет повысить информативность формируемых уведомлений и упростить анализ инцидентов специалистами по информационной безопасности.

2.2 Разработка метода и алгоритма идентификации эксплуатируемых паролей

Разработанный интеллектуальный метод оценки пользовательских паролей в операционных системах семейства Linux основан на применении технологий машинного обучения и анализа статистических характеристик случайности. Предлагаемый метод предназначен для выявления потенциально компрометируемых паролей на этапе их создания или изменения пользователем и может использоваться в составе подсистем аутентификации операционных систем семейства Linux.

В отличие от традиционных механизмов контроля сложности паролей, основанных на фиксированных требованиях к минимальной длине, использованию цифр, специальных символов и символов различных регистров, предлагаемый

метод ориентирован на оценку вероятности компрометации пароля с учетом статистических закономерностей, характерных для ранее скомпрометированных учетных данных. Такой подход позволяет учитывать не только формальное соответствие парольной политике, но и фактическую устойчивость пароля к подбору и использованию в атаках на учетные записи пользователей.

Схема предлагаемого метода представлена на рисунке 2.2.

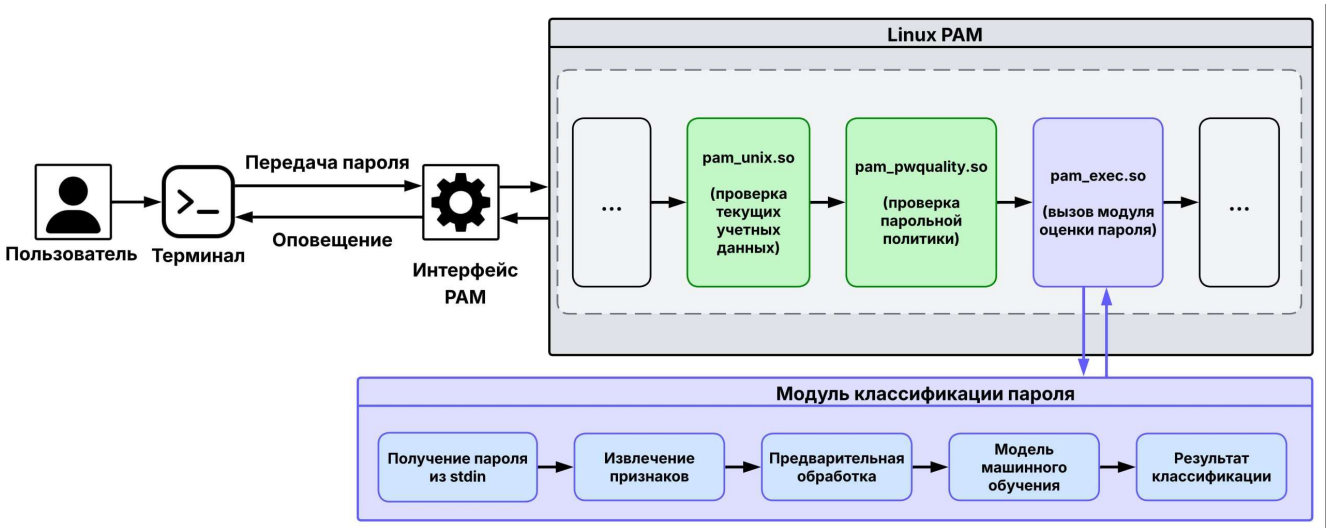


Рисунок 2.2 – Интеграция программного комплекса с модулем PAM

Функционирование метода начинается в момент регистрации новой учетной записи либо изменения существующего пароля. Пользователь вводит пароль стандартными средствами операционной системы, после чего пароль передается в подсистему аутентификации Linux-PAM. На данном этапе вместо непосредственной проверки на соответствие заранее заданным правилам выполняется передача пароля в модуль оценки устойчивости.

Полученный пароль рассматривается как последовательность символов, для которой формируется набор количественных характеристик. Основной целью данного этапа является преобразование символьного представления пароля в набор признаков, пригодных для последующего анализа средствами машинного обучения. Для этого вычисляются характеристики, отражающие структуру пароля, разнообразие используемых символов и степень его случайности.

В частности, определяются длина пароля, количество цифр, строчных и прописных букв, количество специальных символов, доля различных типов символов относительно общей длины пароли, показатели повторяемости символов,

энтропия, частотные показатели распределения символов, коэффициенты разнообразия алфавита, а также признаки, характеризующие наличие повторяющихся шаблонов и закономерностей построения пароля.

Выбор признакового пространства основан на результатах анализа критериев надежности паролей, выполненного в разделе 1.3.1. Проведенное исследование показало, что существенная часть простых паролей характеризуется наличием статистических закономерностей, отсутствием равномерности распределения символов и выраженными структурными шаблонами. Напротив, надежные пароли обладают свойствами, характерными для случайных последовательностей, включая высокое разнообразие символов, отсутствие повторяющихся структур и более равномерное использование различных алфавитов.

По этой причине в состав признакового пространства были включены характеристики, позволяющие косвенно оценивать степень случайности анализируемого пароля. К таким признакам относятся относительное количество уникальных символов, частота смены алфавитов, количество задействованных алфавитов и среднее расстояние между соседними символами на клавиатурной раскладке. Совокупный анализ указанных характеристик позволяет количественно описывать отклонение структуры пароля от поведения, характерного для случайных последовательностей.

В результате выполнения этапа извлечения признаков пароль преобразуется в вектор признакового описания, характеризующего структуру и статистические свойства анализируемого пароля. Полученный вектор используется в качестве входных данных для модели машинного обучения.

На следующем этапе осуществляется интеллектуальная оценка устойчивости пароля. Модель машинного обучения анализирует сформированный набор признаков и определяет вероятность того, что исследуемый пароль обладает характеристиками, свойственными ранее скомпрометированным или легко подбираемым паролям. Результатом работы модели является значение вероятности компрометации, характеризующей степень риска использования анализируемого пароля.

В отличие от традиционных средств оценки сложности паролей,

использующих набор независимых правил и пороговых ограничений, предлагаемый метод анализирует совместное влияние нескольких характеристик случайности. Отдельный признак сам по себе не позволяет достоверно определить надежность пароля. Например, наличие специальных символов или использование нескольких алфавитов не гарантирует устойчивость пароля при наличии повторяющихся шаблонов либо предсказуемых последовательностей. Поэтому решение принимается на основании совокупности признаков, отражающих различные аспекты случайности и структурного разнообразия анализируемой символьной последовательности.

Использование модели машинного обучения позволяет выявлять зависимости между признаками, характерные для ранее скомпрометированных паролей, и формировать итоговую оценку вероятности компрометации без необходимости задания большого количества жестких эвристических правил.

Полученная вероятность интерпретируется как количественная оценка устойчивости пароля. Низкие значения соответствуют паролям, обладающим высоким уровнем случайности и разнообразия символов, тогда как высокие значения свидетельствуют о наличии признаков, характерных для потенциально уязвимых паролей. Использование вероятностной оценки позволяет отказаться от жестких пороговых правил и перейти к более гибкому механизму принятия решений, учитывающему совокупность факторов, влияющих на безопасность учетных данных.

После вычисления вероятности компрометации формируется результат проверки. Если значение риска не превышает установленного порога, пароль признается допустимым и может быть использован пользователем. В противном случае пользователю возвращается уведомление о недостаточной устойчивости выбранного пароля. Дополнительно формируются рекомендации по его улучшению, основанные на анализе наиболее значимых признаков. В качестве причин отклонения могут указываться недостаточная длина пароля, низкое разнообразие используемых символов, наличие повторяющихся последовательностей, использование предсказуемых шаблонов или недостаточный уровень энтропии.

Сформированный результат возвращается в подсистему Linux-PAM, которая принимает окончательное решение о возможности использования пароля. Таким образом, предлагаемый метод интегрируется в стандартный процесс аутентификации операционной системы и обеспечивает интеллектуальную оценку устойчивости учетных данных без изменения существующих механизмов управления пользователями.

Предлагаемый метод позволяет перейти от традиционного контроля соответствия парольной политике к интеллектуальному анализу качества пользовательских паролей. Использование технологий машинного обучения обеспечивает возможность учета сложных зависимостей между признаками пароля и признаками его потенциальной компрометации, что позволяет повысить качество оценки устойчивости учетных данных и обеспечить формирование обоснованных рекомендаций пользователям по повышению уровня защищенности создаваемых паролей.

2.2.1 Обоснование использования машинного обучения

Чтобы обосновать использование моделей машинного обучения для классификации паролей и оценить релевантность выбранных параметров пароля, указанных в таблице 1.6, использован метод многомерной линейной регрессии. На основе ранее сформированного набора данных было установлено несколько зависимостей между сложностью пароля и каждым отдельным параметром.

Для снижения влияния коллинеарности и мультиколлинеарности была построена матрица корреляций (рисунок 2.3) для всех параметров, перечисленных в таблице 1.6. Здесь стоит отметить сильную положительную корреляцию между длиной Len и энтропией E. Действительно, согласно уравнению (1), существует прямая связь между ними, поэтому мы не будем принимать второй параметр дальше в расчет. Кроме того, была наблюдена значительная зависимость между U и L, что можно объяснить тем фактом, что большинство простых паролей состоят из слов, начинающихся с прописной буквы. В то же время можно заметить умеренную отрицательную корреляцию параметра U с параметрами Len, S и N. Примерно такая же умеренная корреляция наблюдалась между Freq и L. Поэтому

мы также решили отказаться от буквенных параметров U и L. На рисунке 2.4 показана матрица корреляций после выбора признаков.

Рисунок 2.3 – Матрица корреляций параметров паролей

Рисунок 2.4 – Матрица корреляций параметров паролей после фильтрации признаков

Статистика оценки параметров с использованием многомерной линейной регрессии представлена в таблице 2.3.

Таблица 2.3 – Статистика оценки параметров

Parameter	R-Square	Standard Error	t-value	$p > \|t\|$	Coefficient
Len	0.1931	0.0002	234.4004	0	0.0424
Uniq	0.2695	0.0053	120.3340	0	0.634
S	0.3777	0.0105	-299.4380	0	-3.1308
N	0.0407	0.0040	-303.2338	0	-1.2096
Dist	0.1442	0.0006	-49.2347	0	-0.0311
Freq	0.5199	0.0040	423.9579	0	1.6857
AC	0.0089	0.0019	-254.3026	0	-0.4795

Из этого видно, что наибольшие значения R-квадрат соответствовали параметрам пароля, таким как S и Freq. Их значения, 0.3777 и 0.5199 соответственно, были явно ниже допустимого значения порога 0.9. Это указывает на то, что зависимости между ними (и, следовательно, любыми другими параметрами пароля, указанными в таблице 1.6) и сложностью пароля не являются линейными, а описываются более сложными функциями. Следовательно, для решения проблемы классификации паролей необходимо использовать алгоритмы машинного обучения, которые отлично справляются с нелинейными зависимостями, в отличие от классических алгоритмических методов. Кроме того, машинное обучение более уместно для многих переменных и большого объема обрабатываемых данных, что соответствует нашему случаю.

Общее значение R-квадрат составило 77%, что указывает на то, что модель объясняет 77% дисперсии зависимых переменных. Полученное значение было выше, чем соответствующее значение для меньшего количества параметров; однако порог 90% также не был достигнут, что не позволило нам утверждать о линейной зависимости между параметрами и сложностью паролей. Значение p-value для каждого параметра приняло значение 0 из-за округления. Однако даже принимая это во внимание, можно сказать, что вероятность получения результатов, когда нулевая гипотеза верна, чрезвычайно мала.

Можно наблюдать высокое значение показателя t-значения для параметров, таких как количество уникальных символов. Это указывает на то, что вероятность

компрометации пароля снижается, если уменьшить количество повторяющихся символов в нем. На рисунке 2.5 показано распределение паролей по количеству уникальных символов.

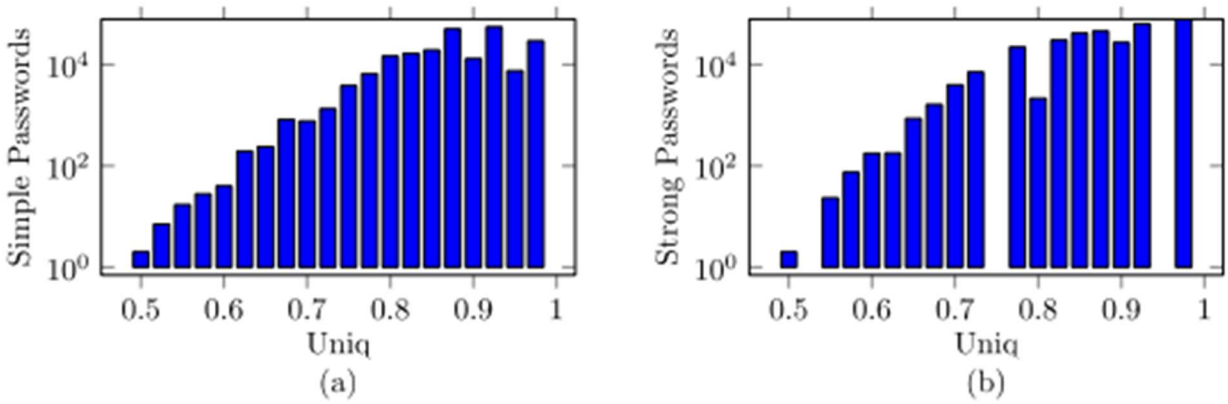


Рисунок 2.5 – Распределение паролей по количеству уникальных символов: (a) простые пароли; (b) надежные пароли

Видно, что этот показатель был значительно выше для надежных паролей (от 90% до 100% случаев), чем для более простых паролей. Средние значения этого параметра для этих массивов составили 92% и 83% соответственно. Это можно объяснить тем фактом, что примерно в 8% всех анализируемых простых паролей частично использовались различные повторяющиеся комбинации символов (например, "123123", "ZXZXZX" и т.д.), что, как было установлено ранее, увеличивает вероятность компрометации учетной записи злоумышленником.

Частота изменения алфавита имеет еще большее влияние. Значение его показателя t-значения составляет 324.1, что указывает на то, что для повышения надежности пароля необходимо как можно чаще менять алфавит при его генерации. Это можно объяснить тем фактом, что примерно каждый четвертый простой пароль состоял либо из слов (латинские буквы в прописном или в основном строчном виде), либо из последовательности чисел, в то время как для надежных паролей наблюдалось равномерное распределение символов между всеми задействованными алфавитами (рисунок 2.6).

Поэтому мы можем заключить, что использование различных слов и числовых комбинаций, особенно связанных с несколькими датами, символами или личной информацией, значительно снижает надежность пароля.

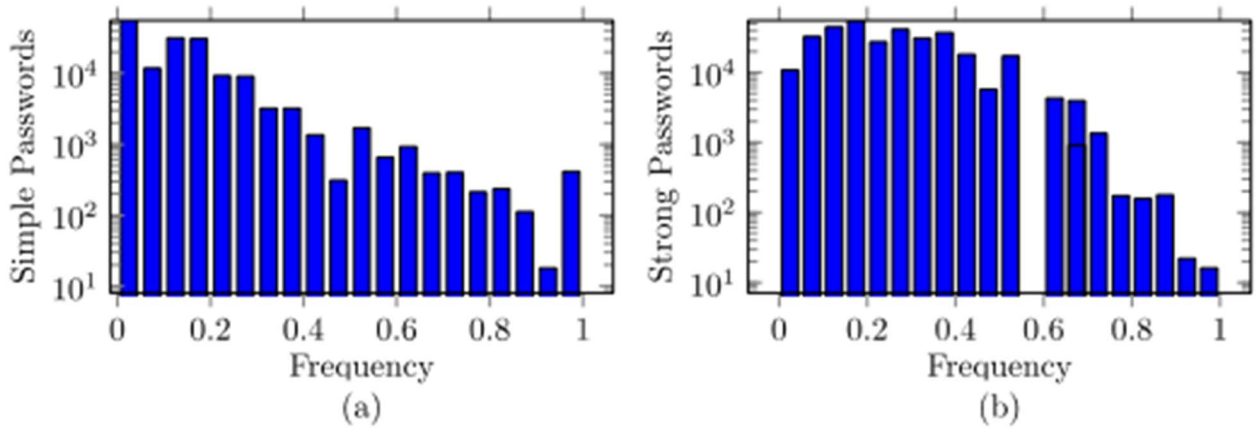


Рисунок 2.6 – Распределение паролей по частоте изменения алфавита: (a) простые пароли; (b) надежные пароли

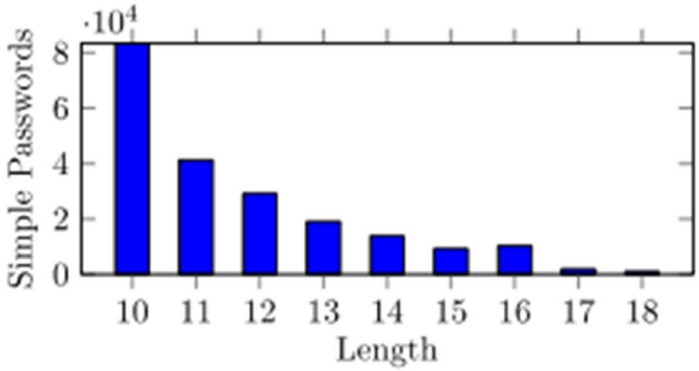


Рисунок 2.7 – Распределение простых паролей по длине

Кроме того, значительный вклад в классификацию внесли параметры, такие как количество используемых алфавитов и длина. Это потому, что анализируемые простые пароли часто использовали три буквы, в то время как обычно использовалось небольшое количество специальных символов, а средняя длина составляла 14–15 символов. Распределение простых паролей по длине показано на рисунке 2.7 (распределение надежных паролей по длине не показано, так как при формировании выборки они были равномерно распределены по этому параметру).

2.3 Разработка метода параллельного вывода

В условиях функционирования систем мониторинга информационной безопасности поток системных событий представляет собой непрерывную последовательность сообщений, поступающих от множества вычислительных узлов, прикладных сервисов и компонентов операционной системы. В средах мониторинга безопасности на базе технологий SIEM поступающие журналы

агрегируются, нормализуются и передаются в интеллектуальные модули анализа, обеспечивающие выявление вредоносной активности и классификацию событий. При этом практическое применение интеллектуальных моделей машинного обучения требует обеспечения минимального времени обработки событий и высокой пропускной способности системы, поскольку задержка анализа приводит к снижению оперативности реагирования на инциденты информационной безопасности. Структурная схема предлагаемого метода обработки представлена на рисунке 2.8.

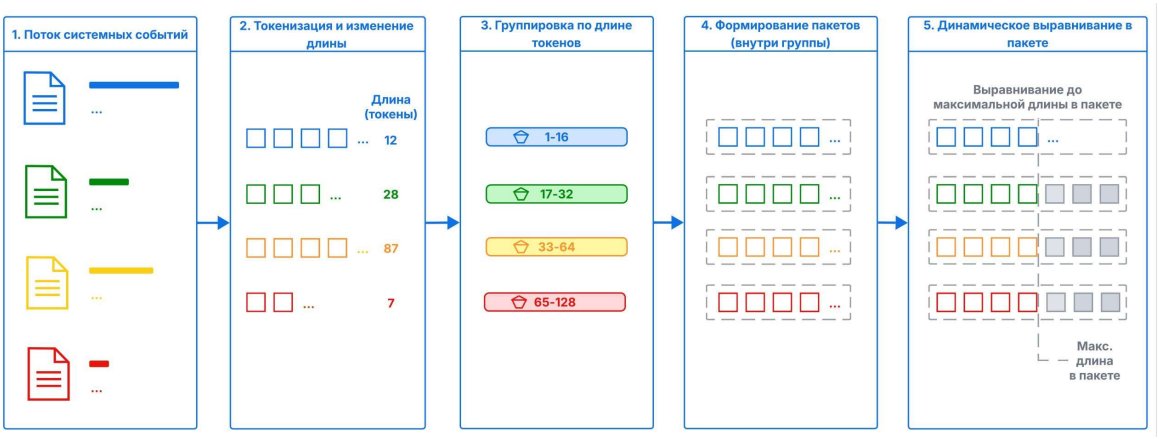


Рисунок 2.8 – Структурная схема предлагаемого метода

В рассматриваемом подходе каждое событие интерпретируется как текстовый объект, анализируемый без построения графовых зависимостей. В качестве основной модели для измерения метрик вывода используется дистиллированная модель семейства BERT. Подобные методы анализа системных событий показали конкурентоспособные метрики качества классификации [9]. Также в работе [98] авторы показывают, что короткие последовательности системных вызовов могут эффективно [11] использоваться как поведенческие признаки, позволяющие различать нормальную и аномальную активность различных UNIX-программ.

После предварительной обработки и токенизации каждое событие преобразуется в последовательность токенов. Поскольку модели семейства Transformer требуют одинаковой размерности входных последовательностей внутри вычислительного пакета, обычно применяется механизм выравнивания длины входных данных. Наиболее распространённым подходом является использование фиксированного максимального размера последовательности, при

котором каждое событие дополняется специальными служебными токенами до заданной длины. В рамках настоящего исследования максимальная длина последовательности ограничивается значением 128 токенов.

Однако в случае фиксированного выравнивания вычислительная нагрузка определяется не фактической длиной событий, а заранее установленным пределом длины последовательности. При использовании фиксированной максимальной длины входной последовательности даже короткие записи журнала обрабатываются так, будто они занимают весь допустимый объём. В результате часть вычислений фактически расходуется на пустые позиции, что особенно заметно при обработке непрерывного потока событий.

Для сокращения таких затрат в работе используется пакетная организация вывода. События передаются в модель не по одному, а группами, то есть объединяются в вычислительный пакет. За счёт этого расходы на подготовку данных и запуск вычислений распределяются между несколькими записями журнала, а не повторяются для каждого события отдельно.

При увеличении размера пакета доля накладных расходов постепенно уменьшается. Однако слишком крупный пакет уже не даёт пропорционального выигрыша, поскольку возрастает нагрузка на центральный процессор и ухудшается использование доступных вычислительных ресурсов. Поэтому размер пакета необходимо подбирать с учётом фактической производительности системы.

Таким образом, возникает задача определения рационального размера пакета, обеспечивающего компромисс между временем обработки и пропускной способностью.

Дополнительным механизмом уменьшения вычислительной нагрузки является динамическое выравнивание последовательностей внутри пакета. В отличие от фиксированного дополнения всех событий до максимального размера 128 токенов предлагается выполнять выравнивание только до максимальной длины последовательности, присутствующей внутри конкретного пакета. Тогда вычислительная сложность обработки определяется уже не глобальным ограничением длины последовательности, а фактической длиной наиболее длинного события внутри пакета.

Использование данного подхода позволяет существенно сократить количество избыточных вычислений в случаях, когда большинство сообщений журналов обладают небольшой длиной. Однако эффективность динамического выравнивания существенно зависит от однородности событий внутри пакета [10,11]. Если пакет одновременно содержит короткие и длинные события, то длина выравнивания определяется наиболее длинным сообщением, вследствие чего часть преимуществ динамического подхода утрачивается.

Для уменьшения неоднородности предлагается введение дополнительного механизма предварительной группировки событий по длине токенизированного представления. В таком случае каждое событие будет относится к одной из групп в зависимости от числа токенов. Формирование пакетов осуществляется только из событий одной группы, что позволяет уменьшить разброс длин последовательностей внутри пакета и снизить число фиктивных вычислений, связанных с дополнением последовательностей.

Таким образом, предлагаемый метод потокового вывода основан на совместном использовании пакетной обработки событий, динамического выравнивания последовательностей и предварительной группировки событий по длине токенизированного представления. В отличие от подходов, предполагающих изменение архитектуры модели или уменьшение числа её параметров, предложенное решение ориентировано на оптимизацию организации вычислительного процесса без модификации структуры интеллектуальной модели.

2.4 Выводы по второй главе

Во второй главе разработана и формально описана архитектура системы раннего обнаружения атак на основе журналов операционных систем и приложений, использующей двухступенчатую схему классификации: бинарное выявление вредоносных событий и последующую многометочную классификацию тактик в терминах MITRE ATT&CK, что обеспечивает не только фиксацию факта атаки, но и понимание ее этапа в kill chain. Показано, что последовательный конвейер сбора, нормализации, токенизации и классификации логов с использованием BERT-совместимых токенизаторов и унифицированного

представления событий позволяет эффективно извлекать семантику из неструктурированных журналов и готовить их к анализу компактными трансформерными моделями.

Обоснован выбор набора данных Linux-MMA-Dataset-2024 как основного источника для обучения и тестирования, поскольку он покрывает полный спектр тактик APT-атак от разведки до утечки и сокрытия данных и превосходит альтернативные датасеты по разнообразию сценариев при достаточной информативности полей для задач бинарной и тактической классификации. Разработана схема предобработки и нормализации журналов (NFKC-нормализация, замена идентификаторов на токены, кодирование путей и фильтрация нерелевантных атрибутов), которая более чем вдвое сокращает длину событий по числу токенов без потери ключевого семантического содержания, что делает возможным использование ограничения в 512 токенов BERT для более 99% записей.

Разработан интеллектуальный метод оценки вероятности компрометации аутентификационных данных, основанный на анализе статистических характеристик случайности паролей и применении моделей машинного обучения. Обоснован выбор признаков, характеризующих структуру и степень случайности символьных последовательностей, показана их связь с вероятностью компрометации паролей и сформировано признаковое описание, используемое для последующей классификации.

Для обоснования применимости ML-подхода проведен анализ параметров паролей (длина, состав, шаблоны и другие) с использованием многомерной линейной регрессии и матрицы корреляций, показавший наличие устойчивых зависимостей между выбранными признаками и сложностью пароля и тем самым подтвердивший целесообразность использования моделей машинного обучения вместо простых пороговых правил.

В третьей части разработан метод потокового вывода интеллектуальных систем анализа системных событий, ориентированный на повышение вычислительной эффективности обработки потоковых журналов без изменения архитектуры модели и снижения качества классификации. Метод основан на

совместном использовании параллельной обработки, динамического выравнивания длины последовательностей и адаптивной группировки событий по вычислительной сложности, что позволяет уменьшить избыточные вычисления при обработке неоднородных входных данных. Формально описаны принципы организации вычислительного процесса, введены критерии формирования пакетов и показана возможность повышения пропускной способности системы.

Таким образом, во второй главе сформирован комплекс взаимосвязанных методов, охватывающий задачи оценки вероятности компрометации аутентификационных данных, обнаружения и классификации вредоносной активности в системных событиях операционных систем семейства Linux, а также повышения вычислительной эффективности интеллектуальных средств анализа информации.

Глава 3 Разработка программного комплекса системы обнаружения атак

3.1 Архитектура и структурная схема программного комплекса обнаружения атак

Для решения задач обнаружения атак с использованием технологий машинного обучения был разработан специальный программный комплекс, реализующий предложенный во второй главе метод обнаружения несанкционированного вторжения на основе журналов событий операционных систем и приложений, а также трансформерных моделей семейства BERT. Программная реализация охватывает полный цикл обработки данных: от предварительной подготовки логов и формирования обучающих выборок до обучения моделей, их сжатия методами дистилляции и квантизации, а также расчёта метрик качества и проведения экспериментального моделирования. Такой подход позволяет связать формальные методы и алгоритмы, описанные во второй главе, с практической реализацией системы, пригодной для последующего развертывания в инфраструктурах с различным уровнем доступных ресурсов.

На рисунке 3.1 представлена типовая схема сети типовой организации, а также показа интеграция предлагаемого программного комплекса.

Одним из ключевых элементов такой сети является SIEM-система, выполняющая функции агрегации и корреляции событий. Обеспечение требуемого уровня защищённости в современных социокиберфизических и распределённых вычислительных средах практически невозможно без применения SIEM-подходов, поскольку именно они обеспечивают централизованный анализ событий безопасности в режиме, близком к реальному времени, а также позволяют выявлять сложные корреляционные зависимости между разнородными событиями, отражающими поведение потенциального нарушителя. В рамках моделей обработки больших данных и технологий глобально распределённой обработки информации SIEM-системы выступают как ключевой компонент инфраструктуры мониторинга, обеспечивая непрерывное наблюдение за состоянием защищаемой среды. Технология SIEM обеспечивает анализ в реальном времени событий безопасности, исходящих от сетевых устройств и приложений, и позволяет

реагировать на них до наступления существенного ущерба [99].

Основными задачами систем данного класса являются сбор, обработка, хранение и анализ событий безопасности, поступающих в систему из множества источников, а также обнаружение в режиме реального времени атак и нарушений критериев и политик безопасности [100]. На основе получаемых событий операционных систем и приложений осуществляется оперативная оценка защищенности информационных, телекоммуникационных и других критически важных ресурсов, анализ и управление рисками безопасности. Во многие современные SIEM системы встроены механизмы принятия решений и инструменты расследования инцидентов.

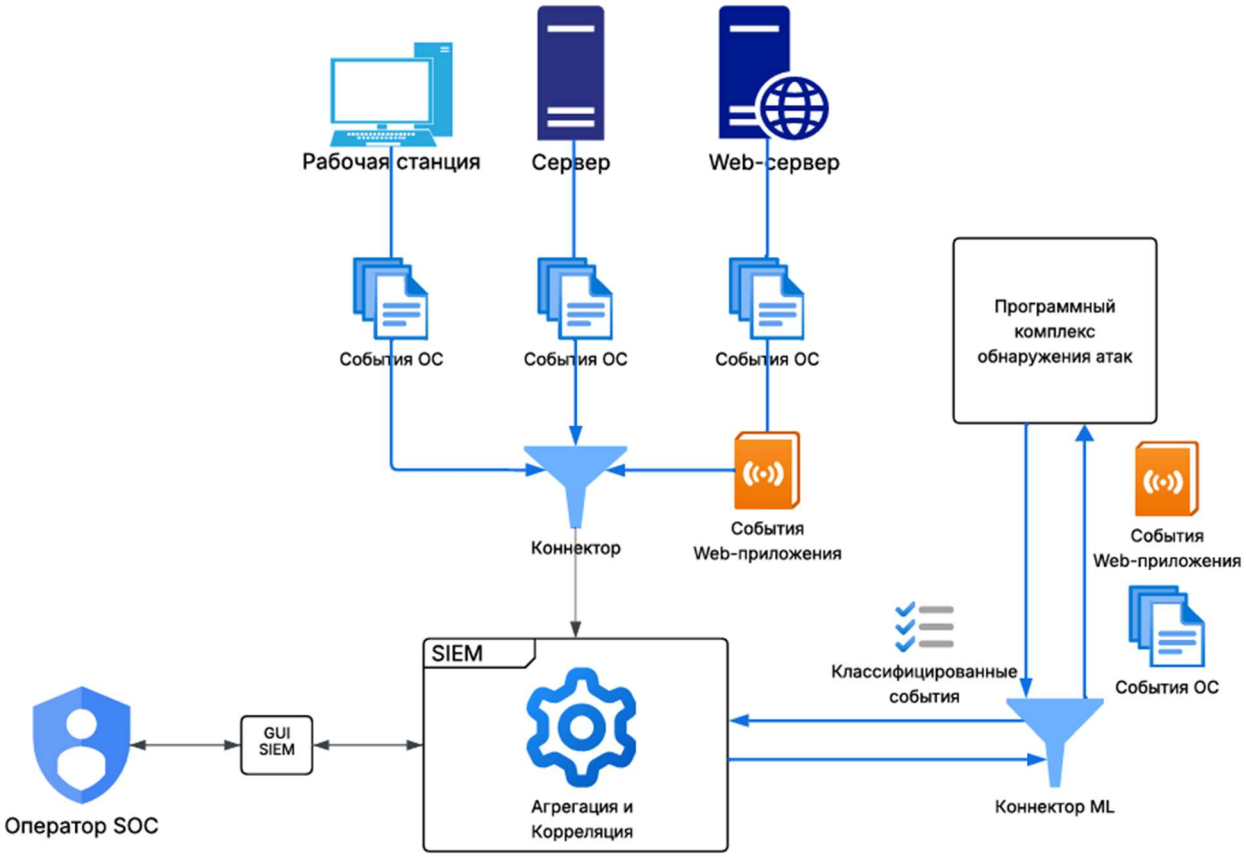


Рисунок 3.1 – Схема интеграции программного комплекса

Ввиду того, что одним из ключевых её элементов являются коннекторы, позволяющие собирать или передавать данные с устройств, нет необходимости в разработке дополнительных механизмов сбора событий со всех потенциально атакуемых устройств и приложений.

Выбор SIEM-системы, как источника событий также объясняется тем, что зачастую специалистам центра информационной безопасности, которые первыми

реагируют на сформированный инцидент, должно выделяться как можно меньше времени для принятия решения, чтобы снизить причиняемый злоумышленником вред к минимуму в случае реальной атаки. Введение же дополнительного интерфейса наоборот способствует увеличению этого времени.

Как видно из рисунка 3.1, события могут быть собраны как с автоматизированных рабочих мест, так и с сервером баз данных, web-приложений или предназначенных для любых других целей. Дополнительный парсинг позволяет привести к общему структурированному виду в человеко-читаемый вид, что позволяет специалистам центра информационной безопасности, используя графический интерфейс (GUI, Graphical User Interface), работать с событиями и инцидентами. Однако для работы программного комплекса этих преобразований недостаточно. Процесс предобработки событий перед работой классификаторов будет описана далее. Передача данных в программный комплекс предполагает введение дополнительного коннектора.

Общая архитектура программного комплекса может быть представлена в виде структурной схемы, включающей несколько логических модулей, последовательно обрабатывающих поток событий (рисунок 3.2).

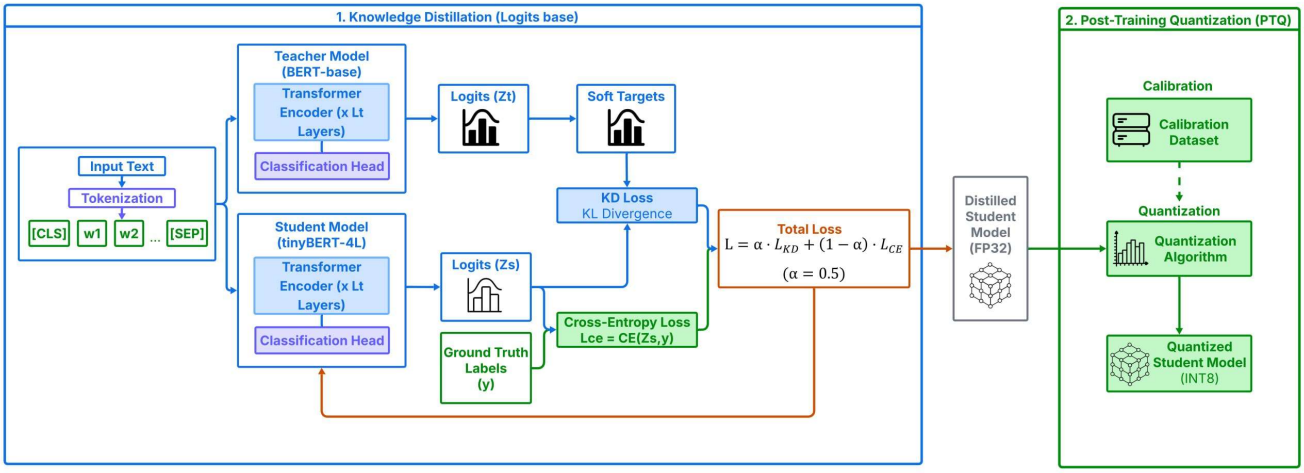


Рисунок 3.2 – Структурная схема программного комплекса классификации системных событий

На вход комплекса поступают журналы аудита и системные события, приведённые к табличному виду с полями, описанными ранее. Далее данные поступают в модуль предварительной обработки, где осуществляется нормализация строк журналов, удаление инфраструктурно-специфичных

атрибутов и приведение текста к унифицированной форме. После этого на основе нормализованных строк формируется текстовое представление событий, которое подаётся на токенизацию с использованием предобученного токенизатора BERT.

Токенизированные последовательности используются для обучения базовой модели бинарной классификации вредоносности, а также для обучения специализированного многометочного классификатора тактик MITRE ATT&CK на подмножестве вредоносных событий. Параллельно реализуются два модуля оптимизации: модуль дистилляции, переносящий знания из полноразмерной модели BERT-base в компактную модель TinyBERT, и модуль квантизации, выполняющий обучение с учётом низкоразрядного представления параметров. На завершающем этапе результаты работы моделей оцениваются средствами вычисления метрик, построения ROC- и PR-кривых, а также визуализации матриц ошибок.

С точки зрения программной реализации архитектура комплекса построена модульно. На рисунке 3.2 можно выделить следующие основные блоки:

- модуль предварительной обработки и нормализации событий;
- модуль обучения базовой модели бинарной классификации;
- модуль обучения многометочного классификатора тактик;
- модуль дистилляции модели и квантизации модели;
- модуль вычисления метрик и анализ качества бинарной модели;
- модуль классификации тактик MITRE ATT&CK и оценка многометочной модели.

Отличительной особенностью структуры комплекса является возможность независимого запуска отдельных сценариев. Это позволяет выполнять предварительную обработку логов один раз и затем многократно переиспользовать подготовленные данные для обучения различных конфигураций моделей, а также повторно оценивать уже обученные модели по новым метрикам без повторного обучения. Аналогично, модуль дистилляции и модуль квантизации могут быть использованы как последовательные этапы оптимизации: сначала осуществляется перенос знаний из базовой модели в компактную архитектуру, а затем к этой

архитектуре применяется обучение с учётом квантизации. Подобная структурная организация упрощает экспериментальное моделирование, позволяя варьировать параметры моделей, архитектуры и режимы оптимизации без изменения основной логики обработки данных.

3.1.1 Модуль предварительной обработки и нормализации событий

Модуль предварительной обработки и нормализации логов является ключевым элементом программного комплекса, обеспечивающим соответствие реальных журналов событий требованиям, предъявляемым к входным данным трансформерных моделей. Общий процесс обработки данных представлен на рисунке 3.3.

Рисунок 3.3 – Модуль предварительной обработки и нормализации событий

Модуль предварительной обработки отвечает за приведение строк аудита к виду, пригодному для подачи в модель. На этом этапе из исходных записей убираются элементы, которые больше описывают особенности конкретной системы, чем само поведение процесса. При этом сохраняются команды, параметры запуска и другие фрагменты, по которым можно судить о характере активности. В программе такая обработка реализована отдельной функцией: она получает строку журнала и возвращает её нормализованное текстовое представление.

Первым шагом проверяется само входное значение. Если запись пустая или не является строкой, она не передаётся на дальнейшую обработку. После этого применяется нормализация Unicode в форме NFKC. Она нужна для того, чтобы одинаковые по смыслу, но по-разному закодированные символы не

воспринимались токенизатором как разные элементы.

Затем из записи исключаются блоки type=CWD, type=NORMAL и type=PROCTITLE. В журналах они обычно содержат сведения о текущем каталоге, типе события и командной строке процесса. Для ручного анализа такие данные могут быть полезны, однако при обучении модели они часто создают лишний шум. Часть этих параметров повторяется от события к событию, а часть слишком сильно зависит от конкретной конфигурации системы.

Далее очищаются служебные поля аудита. К ним относятся идентификаторы пользователей и групп, временные метки и другие похожие атрибуты. Такие данные напрямую зависят от конкретной среды, в которой собирались журналы: состава учётных записей, настроек безопасности и параметров операционной системы. При переносе модели на другую инфраструктуру эти признаки могут потерять смысл или, наоборот, создать ложные зависимости. Поэтому служебные поля исключаются из текста журнала с помощью регулярных выражений, настроенных на UID, GID и сходные по формату параметры.

Числовые значения pid и ppid также не используются в исходном виде. Сам номер процесса не несёт устойчивой информации для классификации, так как он назначается системой динамически. При этом сам факт связи между процессами полезен, поэтому вместо конкретных идентификаторов в строке остаются обобщённые обозначения. Такой вариант сохраняет структуру события, но не заставляет модель опираться на случайные числовые значения.

Отдельная обработка применяется к сетевым и криптографическим данным [10]. В строках аудита могут встречаться IP-адреса, доменные имена, URL и хеши файлов или полезных нагрузок. Такие признаки часто относятся к конкретной инфраструктуре или отдельной кампании атаки и могут быстро изменяться. Если оставить их без изменений, модель способна запомнить отдельные адреса, домены или хеши и использовать их как основное основание для решения. Поэтому сетевые элементы заменяются общими маркерами, а хеш-значения исключаются, поскольку они плохо подходят в качестве стабильных признаков для различения обычной и вредоносной активности.

В конце строка приводится к единому техническому виду. Повторяющиеся пробелы заменяются одним пробелом, лишние пробельные символы по краям удаляются, а все символы переводятся в нижний регистр. Эта часть обработки не меняет смысл события, но упрощает последующую токенизацию, уменьшает словарь и снижает влияние разных вариантов записи одних и тех же команд. В результате работы модуля предобработки каждая исходная запись журнала преобразуется в компактную, нормализованную строку, содержащую только семантически значимые токены команд, параметров и атрибутов, что непосредственно повышает эффективность последующей классификации.

3.1.2 Модуль обучения базовой модели бинарной классификации

Модуль обучения базовой модели бинарной классификации реализует алгоритмы, описанные во второй главе, и отвечает за построение высокоточного классификатора, разделяющего события журналов на нормальные и вредоносные.

Работа модуля для подготовки базовых моделей бинарной и многометочной классификаций представлена на рисунке 3.4.

В качестве основы используется предобученная архитектура BERT-base, адаптированная под задачу двоичной классификации путём добавления выходного полносвязного слоя на два логита. Реализация модуля опирается на использование современной инфраструктуры для обучения трансформеров, что позволяет гибко настраивать параметры обучения и оценивать качество модели.

На первом этапе работы модуля загружается размеченный набор данных, сформированный на основе Linux-APT-Dataset-2024. Данные представлены в табличном виде, где каждая строка соответствует одному событию, а столбцы содержат поля журнала и метку класса. На основе поля, содержащего полное представление события, формируется текстовое представление, которое используется в качестве входа для языковой модели. Для обеспечения корректной оценки качества и предотвращения смещения используется стратифицированное разбиение данных на обучающую и тестовую выборки в фиксированной пропорции, что сохраняет исходное соотношение между нормальными и вредоносными событиями в обеих частях датасета.

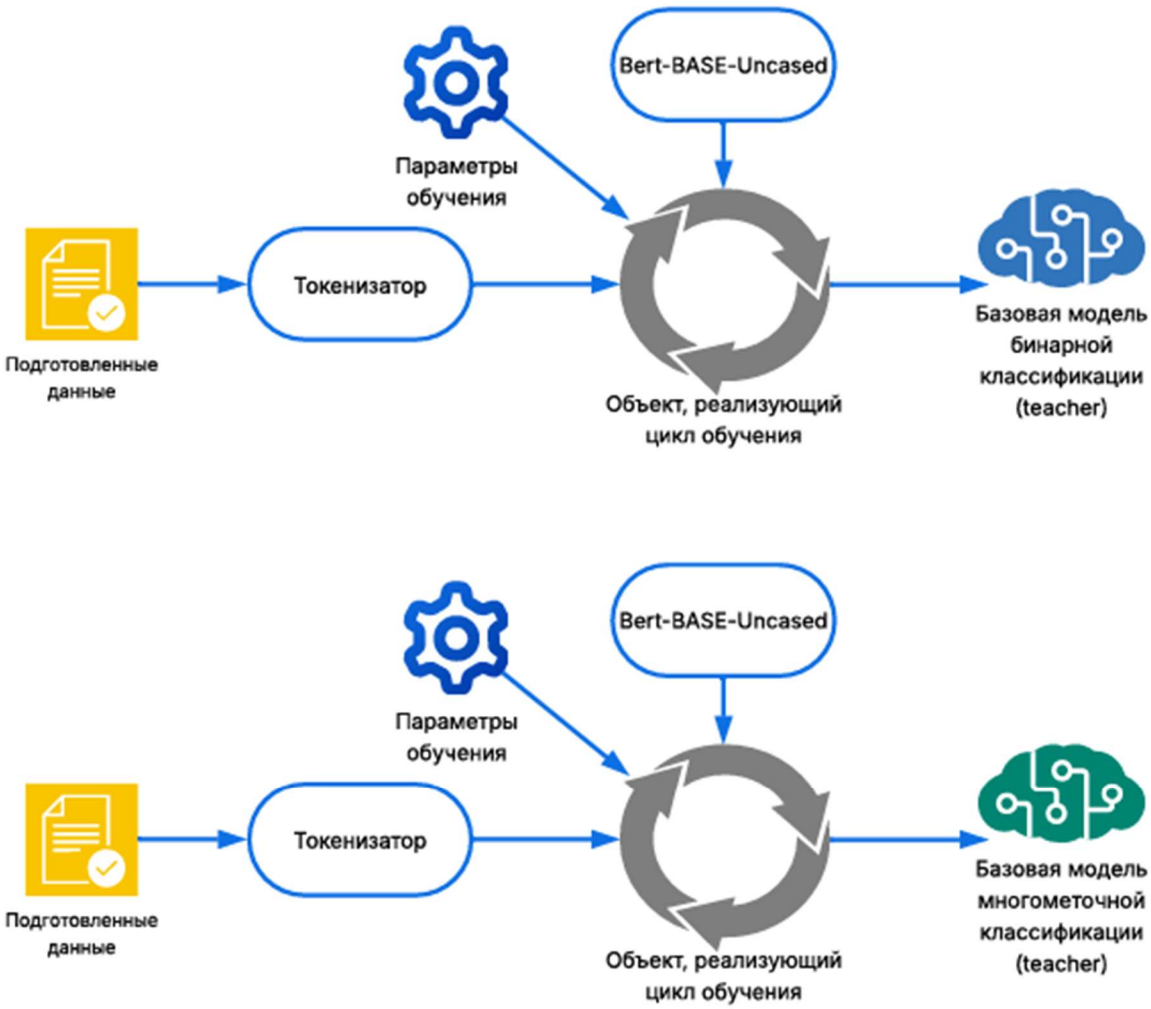


Рисунок 3.4 – Модуль обучения базовой модели бинарной классификации

Следующим этапом является токенизация текстовых представлений с использованием предобученного токенизатора BERT. Для каждой строки текста выполняется разбиение на субсловные токены, добавление специальных токенов начала и конца последовательности, формирование масок внимания и приведение длины последовательности к фиксированному значению за счёт усечения или дополнения паддингом. В базовой конфигурации выбран максимально допустимый размер последовательности, равный 512 токенам, что позволяет не терять информацию для длинных событий и командных строк.

Модуль задаёт набор параметров оптимизации, включающий размер пакета, число эпох, скорость обучения и частоту логирования. С их учётом создаётся объект, реализующий цикл обучения, в котором на каждом шаге рассчитываются функции потерь и обновляются параметры модели. Для оценки качества используется функция вычисления метрик, которая принимает на вход logits и

истинные метки и возвращает значения точности, полноты, F1-меры и матрицы ошибок. Эти метрики фиксируются как для обучающей, так и для тестовой выборки, что позволяет контролировать процесс обучения и своевременно обнаруживать признаки переобучения.

По завершении обучения модуль выполняет сохранение обученной модели и соответствующего токенизатора в файловую систему. Сохранённая модель выступает в дальнейшем в роли учителя в модуле дистилляции и служит базовым эталоном качества при сравнении с компактной и квантованной версиями. Для наглядного анализа работы классификатора строится матрица ошибок, визуализируемая в виде тепловой карты, где по осям отложены истинные и предсказанные классы. Эта визуализация позволяет оценить баланс ошибок первого и второго рода и понять, насколько модель склонна к ложноположительным или ложноотрицательным решениям при выборе порога классификации.

3.1.3 Модуль дистилляции модели

Для обеспечения возможности использования классификатора в ресурсно-ограниченных средах разработан модуль дистилляции, позволяющий перенести знания из полноразмерной модели BERT-base в компактную архитектуру TinyBERT.

Работа модуля по подготовке итоговых моделей бинарной и многометочной классификаций представлена на рисунке 3.5.

Модуль дистилляции реализует подход, при котором обучаемая модель-ученик стремится воспроизвести поведение модели-учителя не только на уровне окончательных решений, но и на уровне распределения вероятностей по классам, что позволяет сохранить информативность выходных представлений при существенном снижении числа параметров.

В основе реализации модуля лежат две модели: учитель, сформированный на предыдущем этапе и загруженный из сохранённого состояния, и ученик, представленный архитектурой TinyBERT. Оба классификатора получают на вход одинаковые токенизированные последовательности, сформированные на основе

нормализованных текстов журналов.

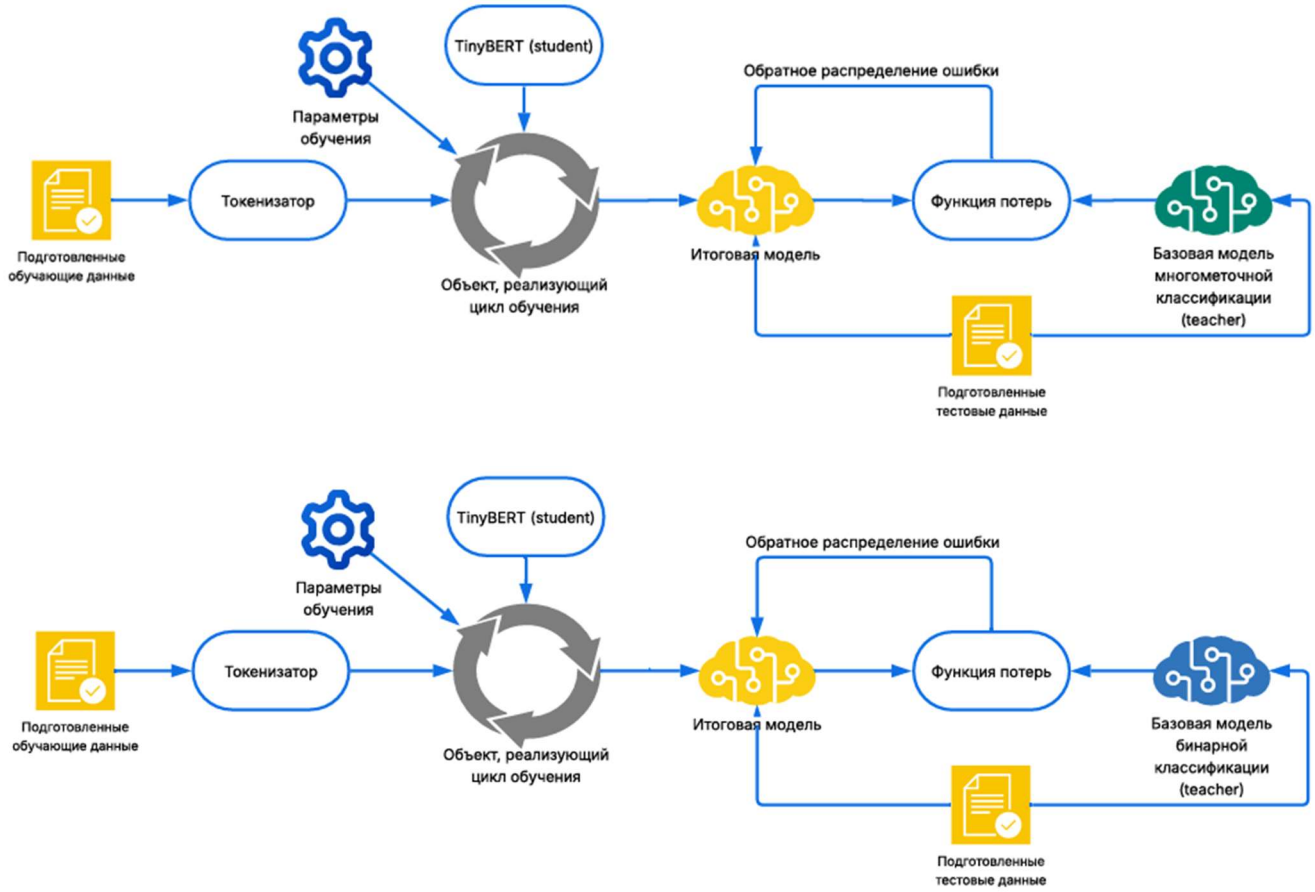


Рисунок 3.5 – Модуль дистилляции модели

Процесс дистилляции организован таким образом, что функция потерь включает два слагаемых. Первое слагаемое представляет собой стандартную кросс-энтропию между логитами модели-ученика и истинными метками, обеспечивая сохранение корректного решения задачи классификации. Второе слагаемое отвечает за приближение распределения вероятностей ученика к распределению учителя и реализуется в виде KL-дивергенции между выходами softmax, вычисленными с использованием повышенной температуры. Температурный параметр отвечает за сглаживание распределения вероятностей, благодаря чему модель-ученик получает более информативные мягкие метки, отражающие различия в уверенности учителя между различными классами.

В ходе обучения веса, соответствующие этим двум слагаемым, подбираются таким образом, чтобы сохранялся баланс между точным воспроизведением истинных меток и переносом знаний из учителя. При этом оптимизация выполняется только для параметров модели-ученика, тогда как модель-учитель

используется в режиме «замороженных» параметров. По окончании обучения с использованием дистилляции компактная модель сохраняется в отдельную директорию, а затем используется в задачах оценки качества и квантизации. В ходе экспериментов было установлено, что при выбранных параметрах температуры и весах отдельных компонентов функции потерь модель TinyBERT сохраняет качество, близкое к исходной модели. При этом её размер меньше, а вывод выполняется быстрее. Для системы мониторинга это важно, поскольку модель должна не только правильно классифицировать события, но и обрабатывать их без заметной задержки.

3.1.4 Модуль квантизации модели

Следующим способом уменьшения вычислительной нагрузки выступает обучение с учётом квантизации. В данном случае компактная модель заранее подготавливается к использованию низкоразрядного представления весов и активаций. Такой режим особенно полезен для устройств с ограниченными ресурсами, например для пограничных узлов и встраиваемых контроллеров, где объём памяти и вычислительная мощность ограничены.

Смысл квантизации с обучением заключается в том, что модель сталкивается с потерей точности представления параметров ещё на этапе оптимизации. Благодаря этому она постепенно адаптируется к будущему низкоразрядному формату, а итоговое снижение качества после перевода в компактное представление оказывается менее заметным.

На уровне реализации для модуля задаётся конфигурация квантизации, совместимая с используемой вычислительной библиотекой. Эта конфигурация применяется ко всем слоям компактной модели, за исключением специальных слоёв встраивания, для которых используются отдельные настройки, обеспечивающие корректное квантование весов при сохранении необходимой точности представления. На этапе подготовки к обучению в модель вставляются дополнительные элементы, имитирующие поведение квантованных операций: наблюдатели, фиксирующие диапазон значений активаций, и вспомогательные преобразования, позволяющие оценивать влияние ограничения разрядности ещё до

окончательного преобразования параметров.

Обучение модели в режиме квантизации с учётом дистилляции организовано аналогично модулю дистилляции. В качестве учителя снова используется полноразмерная модель, обеспечивающая эталонные распределения вероятностей, а модель-ученик представляет собой уже подготовленную к квантованию компактную архитектуру. Функция потерь включает в себя кросс-энтропию по истинным меткам и KL-дивергенцию между распределениями учителя и ученика, взвешенные с заданными коэффициентами. В результате модель учится воспроизводить решения учителя с учётом ограничений, накладываемых квантованием, а встроенные в сеть имитаторы квантованных операций позволяют оценить влияние разрядности на качество уже в процессе обучения.

По завершении заданного числа эпох выполняется окончательное преобразование модели в квантованный формат. После настройки конфигурации операции, имитирующие квантование, заменяются реальными квантованными реализациями, а параметры сети переводятся в низкоразрядный формат. Полученная модель сохраняется отдельно, чтобы затем выполнить её проверку и при необходимости использовать в продуктивной среде.

Качество квантованной модели оценивается на тестовой выборке. Для этого рассчитываются точность, полнота, F1-мера и строится матрица ошибок. Сравнение с базовой и дистиллированной моделями показывает, насколько квантизация влияет на итоговую классификацию и компенсируется ли возможное снижение точности выигрышем в скорости и потреблении ресурсов.

3.1.5 Модуль вычисления метрик и анализ качества бинарной модели

Для бинарной задачи классификации используется расширенный набор метрик. Оценка не ограничивается только общей точностью и F1-мерой, поскольку эти показатели не всегда достаточно полно описывают работу модели в задаче обнаружения атак. Дополнительно анализируются ROC-AUC, PR-AUC, структура ошибок и время вывода. Такой анализ особенно важен в контексте задач информационной безопасности, где разные типы ошибок

имеют разную стоимость для оператора.

В модуле реализован сценарий, который загружает обученную модель и соответствующий токенизатор, подготавливает тестовый набор данных и выполняет вывод без обучения. На вход подаётся часть датасета, не использованная в процессе обучения, причём данные формируются в таком же формате, как и на этапе обучения: для каждого события имеется текстовое поле, нормализованное модулем предобработки, и метка класса. Время выполнения вывода измеряется с использованием стандартных средств, что позволяет оценить задержку обработки при заданном объёме данных и охарактеризовать производительность модели.

На основе полученных logits для каждого элемента тестовой выборки формируется предсказанный класс. Далее рассчитывается отчёт о классификации, включающий точность, полноту и F1-меру для каждого класса, а также средневзвешенные показатели. Матрица ошибок строится как двумерная таблица, где по строкам расположены истинные метки, а по столбцам - предсказанные. Визуализация матрицы ошибок в виде тепловой карты позволяет наглядно оценить, насколько хорошо модель разделяет классы, и увидеть, склонна ли она, например, чаще относить вредоносные события к нормальным или наоборот.

Ключевыми метриками для оценки устойчивости модели к изменению порога классификации являются ROC-AUC и PR-AUC. ROC-AUC и PR-кривые строятся по вероятностям принадлежности события к положительному классу. Эти вероятности получаются из logits с помощью softmax или сигмоиды. ROC-кривая показывает, как меняется соотношение истинноположительных и ложноположительных срабатываний при разных порогах решения, а ROC-AUC отражает площадь под этой кривой.

PR-кривая связывает точность и полноту. Для несбалансированных наборов данных она особенно полезна, поскольку показывает, насколько хорошо модель сохраняет точность при попытке обнаружить больше вредоносных событий. В задаче мониторинга это важнее, чем просто высокая общая точность, так как ошибки разных типов имеют разную практическую цену.

3.1.6 Модуль классификации тактик MITRE ATT&CK и оценка многометочной модели

Помимо бинарной классификации вредоносности отдельное значение имеет задача присвоения вредоносным событиям тактик MITRE ATT&CK, отражающих этапы развития атаки. Для её решения разработан модуль классификации тактик, реализующий многометочную классификацию, при которой одному событию может соответствовать несколько тактик одновременно. Этот модуль позволяет перейти от простого обнаружения вредоносной активности к её тактической интерпретации, что важно для аналитиков и автоматизированных систем реагирования.

В основе модуля лежит специализированная модель на базе трансформера, архитектура которой адаптирована для выдачи набора логитов по числу тактик, включённых в рассматриваемый список. На вход модель получает только те записи журнала, которые были ранее классифицированы как вредоносные, при этом для каждого такого события задан вектор бинарных меток, отражающий наличие или отсутствие каждой тактики. Формирование векторов меток осуществляется на основе строкового поля с перечислением тактик, которое преобразуется в двоичный вектор фиксированной длины.

Обучение многометочной модели реализуется по схеме, аналогичной бинарной классификации, с той разницей, что на выходе модели используется набор сигмоидальных активаций, соответствующих каждому элементу вектора тактик. Функция потерь строится на основе бинарной кросс-энтропии по каждой координате вектора, после чего происходит усреднение по всем тактикам. Для оценки качества используются как общие, так и классовые метрики. Общие метрики включают микро-точность, вычисляемую по всем элементам векторов одновременно, и макро-точность, представляющую собой среднее значение точности по тактикам. Для каждой тактики отдельно рассчитываются точность, полнота, F1-мера и ROC-AUC, что позволяет выявить, какие именно этапы атаки определяются моделью лучше или хуже.

Для визуализации структуры ошибок в многометочной постановке строятся матрицы ошибок по каждой тактике, отражающие, насколько часто модель

правильно и неправильно предсказывает наличие или отсутствие данной тактики. На основе вероятностей принадлежности к тактике для каждой координаты строятся PR-кривые, по которым рассчитывается площадь под кривой. Сопоставление PR-AUC по тактикам позволяет выявить, какие тактики наиболее устойчиво распознаются моделью и где требуется дополнительная доработка датасета или архитектуры.

В совокупности модуль классификации тактик расширяет функциональность системы от обнаружения до аналитического описания поведения атакующих, что повышает ценность решения для практических систем обнаружения и реагирования. Результаты, полученные с его помощью, используются при сравнении предложенного подхода с существующими решениями, основанными на более тяжёлых моделях, и служат основанием для выводов о целесообразности отказа от сложных графовых архитектур в пользу оптимизированных трансформеров.

3.1.7 Исследование архитектур трансформерных моделей

После реализации механизмов предварительной обработки, нормализации и формирования контекстного представления системных событий была выполнена оценка различных архитектур трансформеров, пригодных для решения задачи обнаружения вредоносной активности.

Поскольку для задачи в данной работе, помимо точности модели, важны такие показатели, как время выполнения и потребление ресурсов, были рассмотрены несколько вариантов BERT: базовая модель BERT [101], TinyBERT [102], MobileBERT [103] и DistilBERT [104]. Каждая из этих моделей была обучена и протестирована на подготовленном датасете, после чего были измерены точность классификации и время вывода. Сводные результаты представлены в таблице 3.1.

По данным из таблицы 3.1 видно, что в рамках данной задачи BERT-Base, MobileBERT и DistilBERT демонстрируют практически одинаковую точность. Различия наблюдаются в основном во времени работы моделей. Поскольку MobileBERT оказался самым медленным, он исключен из дальнейшего анализа.

Таблица 3.1 – Сравнение вариантов BERT

Model	Inference time, c	Precision		Recall		F1-score		Accuracy
		normal	malicious	normal	malicious	normal	malicious	
BERT-Base	486.31	0.99	1.00	0.99	0.97	1.00	0.98	0.99
DistilBERT	252.96	0.99	1.00	1.00	0.97	1.00	0.98	0.99
MobileBERT	691.80	0.99	1.00	1.00	0.97	1.00	0.98	0.99
TinyBERT-4L	127.75	0.87	1.00	1.00	0.37	0.93	0.54	0.88

DistilBERT показал почти двукратное ускорение по сравнению с bert-base-uncased при сохранении идентичной точности.

Однако четырехслойная модель TinyBERT показала значительное преимущество по времени вывода: она работает в 3.8 раза быстрее стандартного BERT и в два раза быстрее сжатых версий. Недостатком TinyBERT является значительное падение качества классификации на вредоносных примерах: модель пропускает около 67% вредоносных событий, что недопустимо для реальных СЦБ-операций.

Исследования в данной работе не ограничились только преимуществом в скорости, показанным DistilBERT, в связи с чем была предпринята попытка повысить точность TinyBERT путем дистилляции знаний от более точных моделей при сохранении малого времени работы.

Оптимальные значения коэффициента весов ($\alpha = 0.5$) и числа эпох обучения (3) были определены эмпирически, что позволило сохранить наивысшую точность при одновременном уменьшении времени вывода модели.

Не менее важным фактором при обучении является значение максимальной длины последовательности токенов (π), на которые разбивается входной текст во время токенизации. Максимальное значение π для всех рассматриваемых моделей равно 512 токенам. В рамках анализа длин всех событий, собранных в наборе данных, было установлено, что значения π, равного 128, достаточно для покрытия более 99% данных. Менее 1% данных имеют значительную длину, требующую π = 512 для обработки. Однако было также обнаружено, что увеличение π существенно увеличивает время, необходимое модели для формирования предсказаний.

Представленные в таблице 3.2 данные показывают, что при увеличении π в 4 раза время вывода модели возрастает втрое.

Таблица 3.2 – Влияние π на время вывода

Value of π	Inference time, s	Precision		Recall		F1-score		Accuracy	PR AUC
		normal	malicious	normal	malicious	normal	malicious		
128	93.97	0.986	0.997	0.999	0.944	0.993	0.970	0.970	0.990
512	270.89	0.991	0.997	0.999	0.963	0.995	0.980	0.980	0.996

Также как можно заметить, что несмотря на существенный выигрыш по скорости, метрика Recall (Malicious) для вредоносных команд снижается лишь незначительно – приблизительно на 2%. Поэтому в дальнейшем были ограничены π значением 128, чтобы охватить 99-й перцентиль всех данных.

В качестве обучающей модели была выбрана стандартная модель BERT. Для сравнения, помимо четырехслойной TinyBERT, была также использована ее шестислойная версия TinyBERT-6L [105]. Показатели качества обеих дистиллированных моделей представлены в таблице 3.3.

Таблица 3.3 – Сравнение дистиллированных моделей TinyBERT

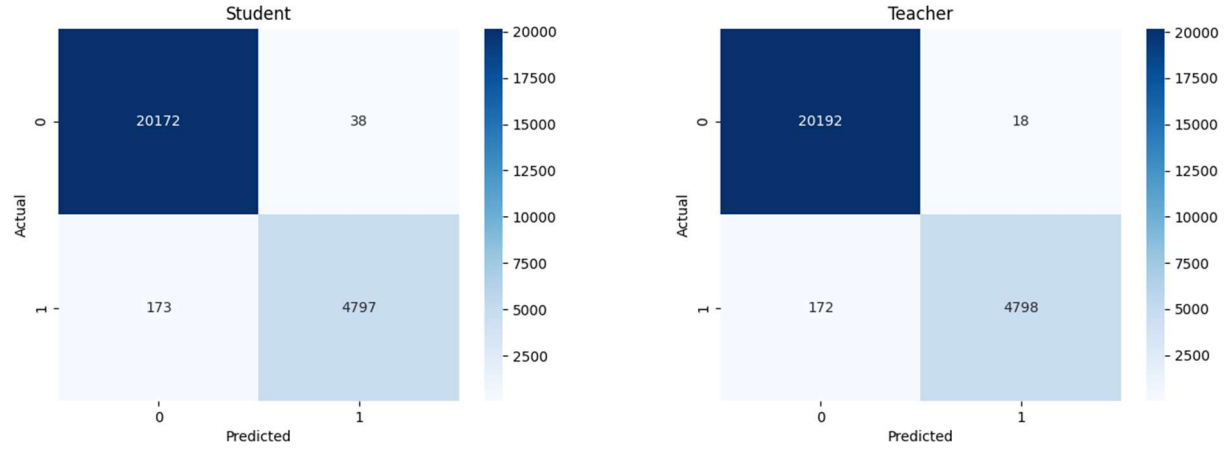
Model	Inference time, s	Precision		Recall		F1-score		Accuracy	PR AUC
		normal	malicious	normal	malicious	normal	malicious		
4L	93.97	0.986	0.997	0.999	0.944	0.993	0.970	0.988	0.990
6L	270.89	0.991	0.997	0.999	0.963	0.995	0.980	0.992	0.996

Значения ROC-AUC и PR-AUC рассматривались отдельно для базовой, дистиллированной и квантованной моделей. Такое сопоставление показывает, насколько сжатие меняет качество обнаружения вредоносных событий. Если после оптимизации метрики остаются на близком уровне, а модель при этом становится быстрее и компактнее, выбранный вариант можно использовать в системе мониторинга без существенной потери качества.

По точности обе модели дают близкий результат, однако усложнение архитектуры влияет на время обработки. Добавление двух дополнительных слоёв увеличило задержку вывода в 1,43 раза. Матрицы ошибок для модели-учителя и модели-студента приведены на рисунке 3.6.

Из-за дисбаланса между доброкачественными и вредоносными событиями особое внимание уделялось PR AUC. В такой задаче ROC-кривая не всегда достаточно наглядна, поскольку может сохранять высокие значения даже при заметном числе ложных срабатываний. PR-кривая лучше отражает практическое качество обнаружения, так как показывает баланс между точностью и полнотой. Значение PR AUC = 0,990 подтверждает, что модель уверенно выделяет вредоносные события и сохраняет низкую долю пропущенных атак.

Таким образом, с помощью дистилляции была получена модель, демонстрирующая высокую точность обнаружения вредоносной активности в событиях при значительном ускорении процесса классификации.



(a) (b)

Рисунок 3.6 – Сравнение дистиллированных моделей TinyBERT: (a) матрица ошибок для модели-студента; (b) матрица ошибок для модели-учителя

Для модели-учителя использовались следующие настройки обучения: максимальная длина токена – 512, размер батча – 8, оптимизатор AdamW с коэффициентом затухания 0.01, параметр warmup – 0.1, а функцией потерь служила CrossEntropyLoss.

Для дистиллированной модели дополнительно применялась 8-битная пост-тренировочная квантизация (PTQ, Post Training Quantization). Только слои типа

torch.nn.Linear были квантизированы, тогда как слои embedding, слои нормализации, функции активации и softmax для attention оставались неизменными. Эти преобразования уменьшили размер модели примерно на 10 МБ при сохранении той же точности классификации и обеспечили примерно двукратное улучшение времени вывода. В таблице 3.4 представлено сравнение моделей с 32-битной и 8-битной PTQ-квантизацией.

Итоговый размер модели бинарной классификации [12] составил 41.48 МБ. При обработке всего набора данных пиковое использование оперативной памяти составило 0.97 МБ, а время вывода – 639.45 с (примерно 11 минут) на машине, работающей только на CPU.

Таблица 3.5 представляет результаты обнаружения стадий атаки на различных машинах. Высокие значения precision, recall и F1-score на всех машинах демонстрируют устойчивость модели независимо от конкретного хоста.

Таблица 3.4 – Сравнение подходов к квантизации модели

Model	Inference Time, s	Precision		Recall		F1-score		Accuracy
		normal	malicious	normal	malicious	normal	malicious	
32-bit PTQ	1167.97	0.989	0.997	0.999	0.955	0.994	0.975	0.990
8-bit PTQ	639.45	0.986	0.997	0.999	0.944	0.993	0.970	0.988

Таблица 3.5 – Точность обнаружения стадий атаки на каждой машине.

Machine	Precision		Recall		F1-score		Accuracy
	normal	malicious	normal	malicious	normal	malicious	
Machine-1-New	0.996	0.995	1.000	0.936	0.998	0.965	0.998
Machine-1	0.996	0.998	1.000	0.937	0.993	0.968	0.997
Machine-2	0.996	1.000	1.000	0.913	0.998	0.967	0.996
Machine-3	0.996	0.867	0.991	0.940	0.993	0.907	0.987

Однако высокая средняя точность модели не устраняет необходимости в дальнейшем снижении доли пропущенных вредоносных событий, поскольку такие ошибки приводят к пропуску скрытых стадий APT-цепочки и могут

способствовать выполнению последующих тактик MITRE ATT&CK без обнаружения. Отмечено, что некоторые вредоносные события классифицируются как нормальные. Это объясняется тем, что определенные команды и последовательности действий в Linux-MMA-Dataset-2024 внешне очень похожи на типичное административное поведение. Дополнительно, операторы APT могут использовать низкопрофильные, медленные тактики, которые трудно отличить от легитимных действий по структуре и параметрам. Эти проблемы могут усугубляться в случаях редких или ранее не встречавшихся шаблонов атак (как показано далее в экспериментах со стадиями Command and Control, Collection и Credential Access), а также высокой вариабельностью параметров команд.

Тем не менее, поскольку злоумышленники часто вынуждены применять уникальные техники и тактики для достижения своих целей, такие действия могут быть обнаружены с высокой вероятностью на других стадиях атаки.

Рассматриваемый набор данных содержит 12 тактик атак, описанных терминологией MITRE: Defense Evasion, Privilege Escalation, Discovery, Initial Access, Execution, Persistence, Lateral Movement, Impact, Collection, Credential Access, Command and Control и Reconnaissance. В таблице 3.6 представлено распределение этих тактик в наборе данных после удаления дубликатов и очистки строк с пустыми значениями.

Таблица 3.6 – Распределение атак по тактикам

Tactic	Count	Train Data	Test Data
Defense Evasion	16,352	13,081	3271
Privilege Escalation	15,416	12,332	3084
Initial Access	14,905	11,923	2982
Discovery	12,938	10,350	2588
Impact	2859	2287	572
Execution	1641	1313	328
Persistence	1097	878	219
Reconnaissance	640	512	128
Lateral Movement	231	185	46
Credential Access	34	27	7
Collection	2	1	1
Command and Control	2	1	1

Полностью обработанный набор данных содержит 114,851 строк, из которых 91,804 – обычные события, а 23,046 – вредоносные. В таблице 3.6 показано, что число тактик значительно выше числа вредоносных событий. Это объясняется тем, что одно вредоносное событие может соответствовать нескольким тактикам одновременно (в диапазоне от 1 до 6).

Для последующей классификации всех событий, идентифицированных как вредоносные, в соответствии с перечисленными тактиками была использована базовая модель BERT с 3 эпохами обучения и максимальной длиной последовательности 128.

Время вывода на всем наборе данных составило 39.02 с (0.0017 с на событие). Поскольку модель работает только с событиями, уже классифицированными как атакующие, это время можно считать приемлемым и не требующим улучшения. Таблица 3.7 представляет результаты классификации вредоносных событий по тактикам.

Таблица 3.7 – Результаты обучения

Tactic	Precision	Recall	F1-Score	AUC
Defense Evasion	0.9988	0.9902	0.9945	0.9993
Privilege Escalation	0.9974	0.9932	0.9953	0.9997
Initial Access	0.9120	0.9977	0.9529	0.9267
Discovery	0.9958	0.9977	0.9967	0.9994
Impact	1.0000	1.0000	1.0000	1.0000
Execution	0.9878	0.9099	0.9476	0.9852
Persistence	1.0000	1.0000	1.0000	1.0000
Reconnaissance	0.9685	0.9690	0.9647	0.9924
Lateral Movement	0.9733	0.9783	0.9758	0.9934
Credential Access	0.7500	0.4286	0.5455	0.9556
Collection	0.0000	0.0000	0.0000	0.0000
Command and Control	0.0000	0.0000	0.0000	0.0000

Модель продемонстрировала высокую надежность почти по всем тактикам. Наивысшие показатели наблюдаются для Defense Evasion, Privilege Escalation, Discovery и Execution, где значения precision, recall и F1-score превышают 0.99. Действительно, из данных, представленных в матрицах ошибок (рисунки 3.7а, 3.8а, 3.9а и 3.10а), видно, что ошибки неверной классификации минимальны. Анализ

ROC и PR также подтверждает высокое качество модели по этим тактикам: площадь под кривой (AUC) находится в пределах ≈0.999 очень близко к идеальному. Также таблица 3.7 показывает высокое значение recall = 0.9977 для тактики Initial Access, что свидетельствует о том, что модель практически не пропускает истинные экземпляры данной тактики. Однако значение precision заметно ниже – 0.9120. Действительно, рисунок 3.13 показывает, что примерно 9% предсказаний были ложноположительными.

На рисунках 3.7b, 3.8b, 3.9b и 3.10b приведены ROC-кривые, показывающие высокую чувствительность и специфичность модели при различных пороговых значениях. На рисунках 3.7c, 3.8c, 3.9c и 3.10c показаны PR-кривые.

Идеальная точность была достигнута для классификации тактик Impact и Persistence: рисунки 3.11 и 3.12 показывают матрицы ошибок, PR-кривые и ROC-кривые. Однако следует отметить, что события тактики Impact имеют специфичную и уникальную форму, присущую только им. Поэтому, после подтверждения факта вредоносности, назначение события этой тактике является прямолинейным.

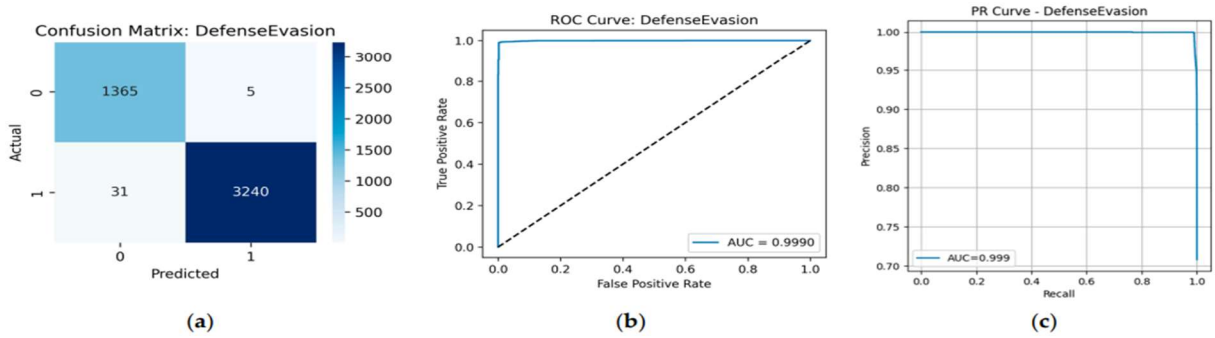


Рисунок 3.7 – Результаты прогнозирования для тактики «Defense Evasion»: (a) матрица ошибок; (b) кривая ROC; (c) кривая PR

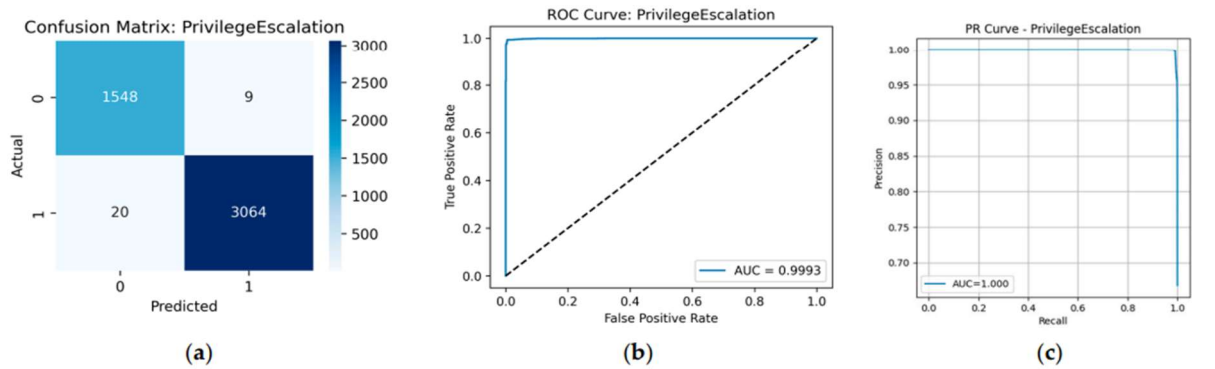


Рисунок 3.8 – Результаты прогнозирования для тактики «Privilege Escalation»: (a) матрица путаницы; (b) ROC-кривая;(c) PR-кривая

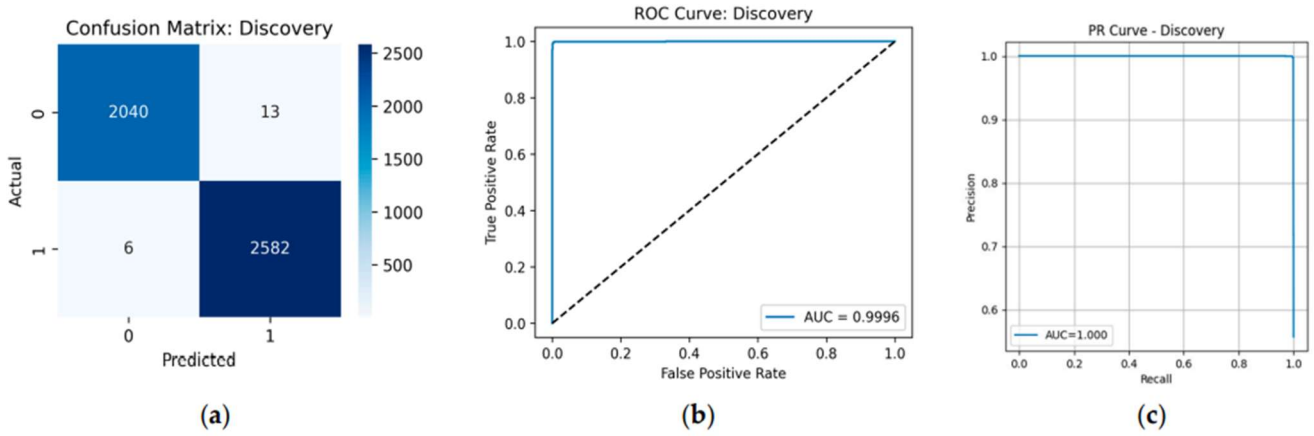


Рисунок 3.9 – Результаты прогнозирования для тактики «Discovery»: (a) матрица ошибок; (b) ROC-кривая; (c) PR-кривая

По данным таблицы 3.6, рассматриваемая тактика встречается примерно в 65 % всех вредоносных событий. Такое преобладание, а также пересечение её признаков с другими тактиками усложняют классификацию и объясняют снижение точности по данному направлению.

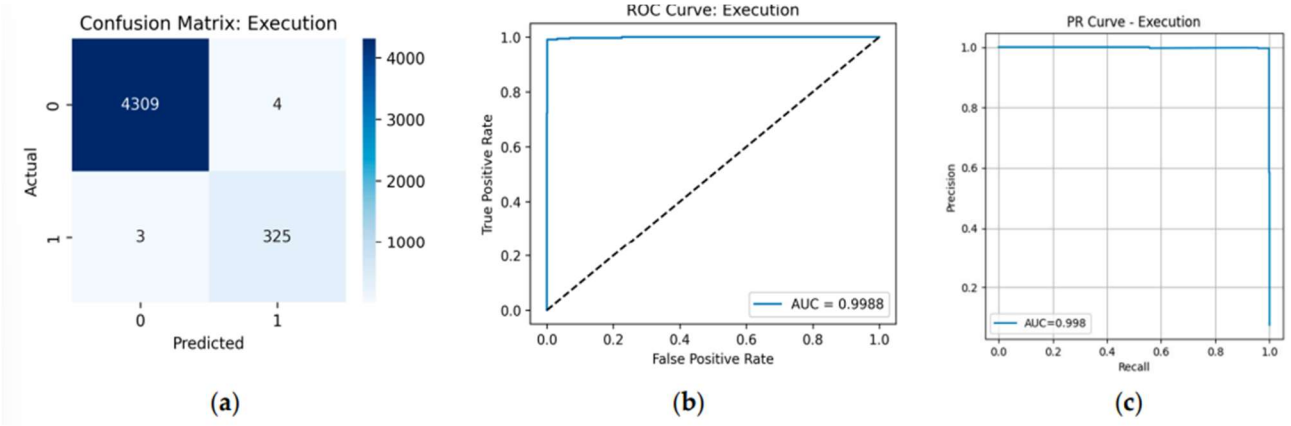


Рисунок 3.10 – Результаты прогнозирования для тактики «Execution»: (a) матрица ошибок; (b) кривая ROC; (c) кривая PR

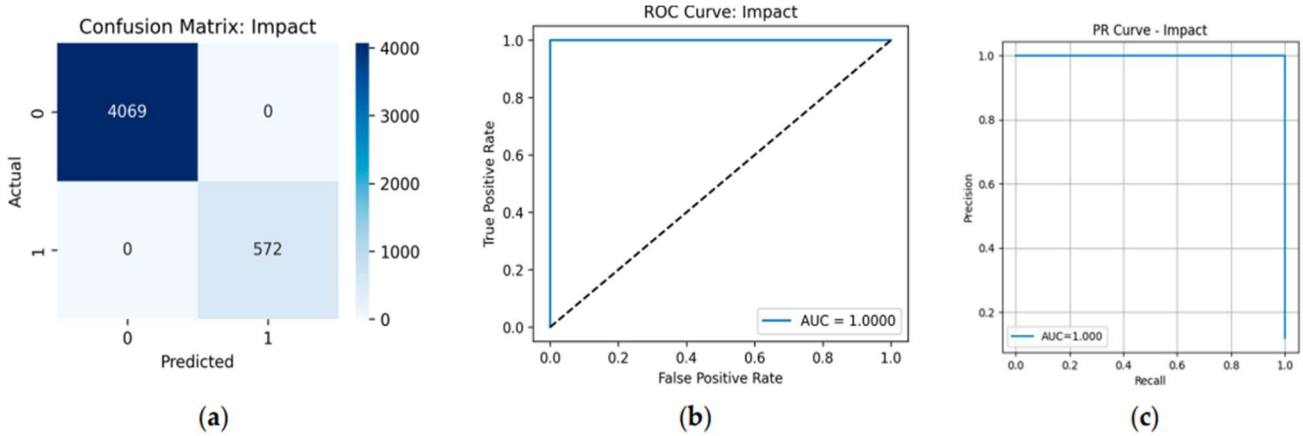


Рисунок 3.11 – Результаты прогнозирования для тактики «Impact»: (a) матрица ошибок; (b) кривая ROC; (c) кривая PR

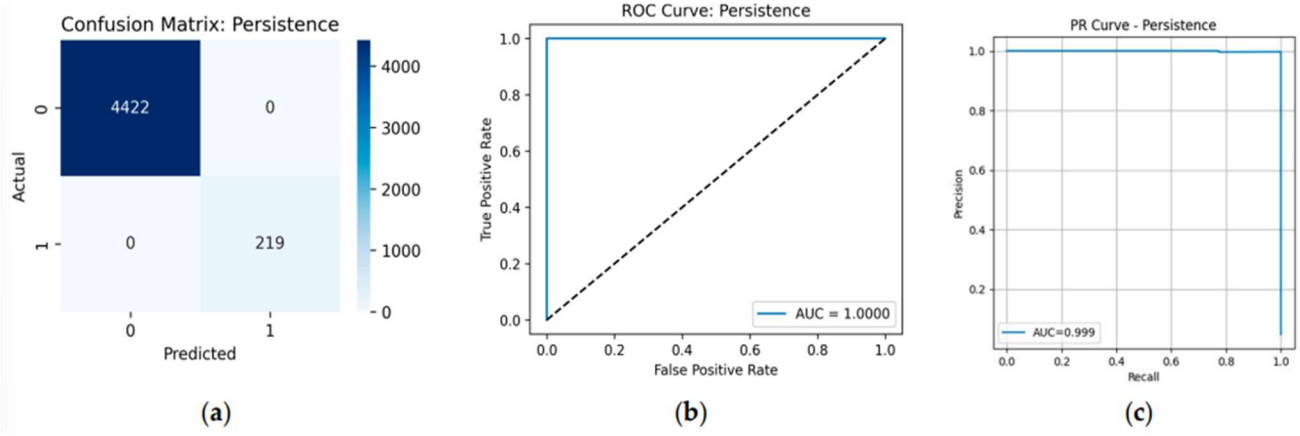


Рисунок 3.12 – Результаты прогнозирования для тактики «Persistence»: (a) матрица путаницы; (b) ROC-кривая; (c) PR-кривая

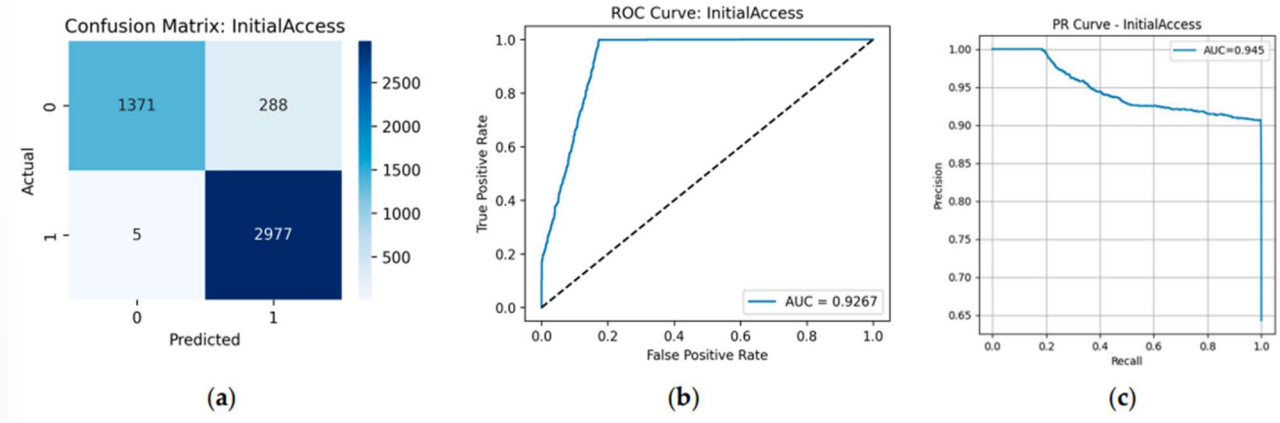


Рисунок 3.13 – Результаты прогнозирования для тактики «Initial Access»: (a) матрица ошибок; (b) ROC-кривая; (c) PR-кривая

Только для 2149 событий, что составляет около 9 %, Initial Access указана как единственная тактика. В остальных 12 756 случаях она встречается совместно с Discovery, Defense Evasion, Persistence, Privilege Escalation или несколькими из этих тактик одновременно. При этом для перечисленных тактик, как отмечалось выше, точность классификации превышает 99 %.

Для Lateral Movement наблюдается обратная ситуация: часть событий других классов была отнесена моделью к этой тактике. Это согласуется со значением precision = 0,98 и указывает на наличие отдельных ложноположительных срабатываний. Однако общее число ошибок для этой тактики составило менее 3% (см. рисунок 3.14), что является хорошим результатом, несмотря на относительно малый размер класса (см. таблицу 3.6).

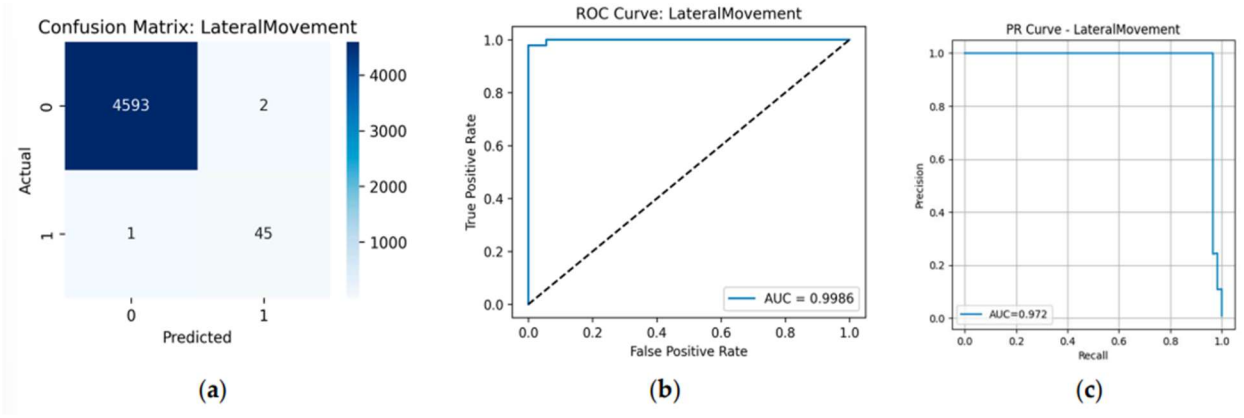


Рисунок 3.14 – Результаты прогнозирования для тактики бокового движения: (а) матрица ошибок; (б) кривая ROC; (в) кривая PR

Стоит подчеркнуть высокую надежность модели при обнаружении атак Reconnaissance. Хотя формат событий, относящихся к этой тактике, очень схож с наиболее распространенными типами атак (представленными в виде веб-логов), общее число событий этой тактики составляет менее 3% (см. таблицу 3.6) без перекрытия с другими тактиками. Тем не менее модель смогла классифицировать этот тип атак с высокой точностью. Рисунок 3.15 показывает, что модель сделала как ложноположительные, так и ложноотрицательные ошибки менее чем в 4% случаев.

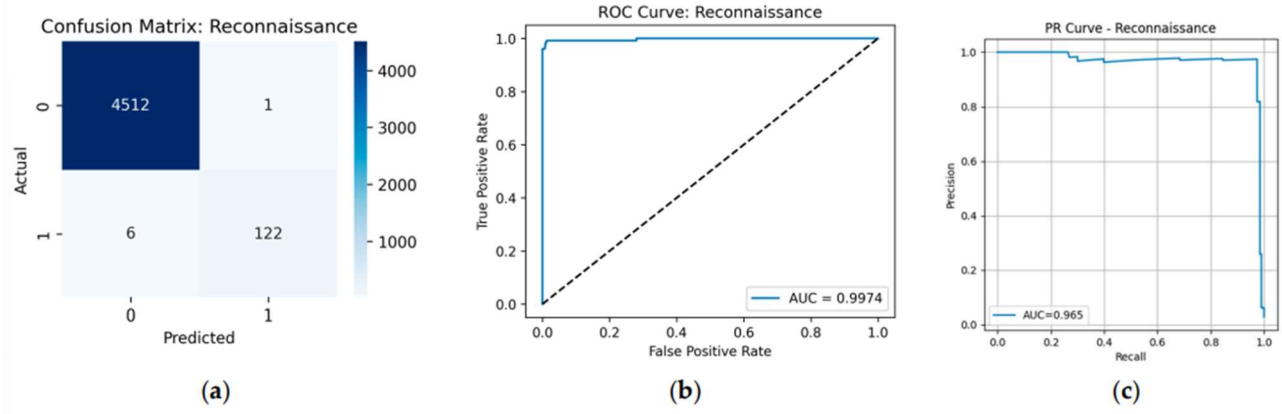


Рисунок 3.15 – Результаты прогнозирования для тактики «Разведка»: (a) матрица ошибок; (b) кривая ROC; (c) кривая PR

Модель не справилась с классификацией типов атак Collection, Credential Access и Command and Control. Это объясняется чрезвычайно малым размером этих классов во всем наборе данных (всего 39 событий, что составляет менее 0.2% от всего набора данных). Этого количества недостаточно для корректного обучения. Тем не менее было установлено, что точность классификации этих событий как

вредоносных составляет примерно 98%, хотя сама тактика не определяется.

Такое низкое качество объясняется природой каждого события: при более детальном рассмотрении становится очевидно, что обнаруженная активность не проявляет ярко выраженных признаков, характерных для других тактик, и может выполняться как злоумышленником, так и администратором или обычным пользователем. Однако поскольку атака не ограничивается только такими действиями, итоговые модели способны на основе других событий в той же последовательности с высокой вероятностью определить отдельные стадии атаки и их тактическую принадлежность согласно MITRE ATT&CK, как показано ранее для более распространенных тактик.

Средние значения метрик по всем тактикам (кроме редко встречающихся Credential Access, Command and Control и Collection) составляют: precision – 0.7991, recall – 0.7781 и F1-score – 0.7848.

Итоговый размер модели многоклассовой классификации [13] составил 41.48 МБ. При обработке всего набора данных пиковое использование оперативной памяти составило 0.99 МБ, а время вывода – 486.25 с (примерно 8 минут) на CPU-машине.

3.2 Архитектура и структурная схема программного комплекса обнаружения потенциально эксплуатируемых злоумышленником паролей

Программная реализация разработанного метода оценки вероятности компрометации аутентификационных данных построена на основе модульной архитектуры и интегрирована в стандартную подсистему управления аутентификацией операционных систем семейства Linux — Pluggable Authentication Modules (PAM). Выбор данного механизма обусловлен тем, что PAM представляет собой унифицированный программный интерфейс, обеспечивающий выполнение операций аутентификации, авторизации и управления учетными данными независимо от конкретных прикладных программ. Использование PAM позволяет внедрить дополнительную процедуру интеллектуального анализа паролей без модификации исходного кода системных утилит, прикладных программ и компонентов ядра операционной системы.

Разработанный программный комплекс состоит из нескольких

взаимодействующих компонентов: стандартной системной утилиты управления учетными записями, подсистемы Linux PAM, модуля интеграции pam_exec.so, программного модуля оценки вероятности компрометации пароля и обученной модели машинного обучения.

Функционирование комплекса начинается при выполнении пользователем операции создания учетной записи либо изменения существующего пароля. Для этого используются стандартные средства операционной системы Linux, такие как passwd, chpasswd, useradd и другие утилиты управления учетными данными. В процессе выполнения указанные программы обращаются к подсистеме PAM через стандартный программный интерфейс операционной системы.

Подсистема PAM организует обработку запроса в виде последовательного вызова модулей, входящих в соответствующий стек обслуживания. В Linux различают стеки auth, account, password и session, каждый из которых отвечает за выполнение определенного класса операций. Поскольку разработанный метод предназначен для анализа нового пароля в момент его установки пользователем, интеграция осуществляется в рамках стека password, предназначенного для обработки операций изменения учетных данных.

Для подключения разработанного программного модуля в конфигурацию PAM используется модуль pam_exec.so. Данный модуль предназначен для запуска пользовательских программ в процессе выполнения PAM-стека и передачи им параметров текущей операции. Использование pam_exec.so позволяет реализовать взаимодействие между стандартной подсистемой управления учетными записями Linux и разработанным модулем оценки компрометации паролей без внесения изменений в существующую программную инфраструктуру операционной системы.

Все компоненты разработанного программного комплекса функционируют на одном вычислительном узле под управлением операционной системы Linux. Передача анализируемого пароля и результатов классификации осуществляется локально средствами операционной системы без использования сетевого взаимодействия и без передачи аутентификационных данных на удаленные серверы. Такой подход исключает появление дополнительных каналов передачи

секретной информации, снижает риск компрометации учетных данных в процессе их обработки и позволяет использовать разработанное решение в изолированных сегментах вычислительной инфраструктуры.

Ключевой особенностью реализации является безопасная передача анализируемого пароля программному модулю оценки компрометации. Для решения данной задачи используется параметр expose_authtok модуля pam_exec.so, обеспечивающий передачу нового пароля через стандартный поток ввода процесса. Такой механизм обладает рядом преимуществ с точки зрения информационной безопасности. Пароль не сохраняется во временных файлах файловой системы, не передается через аргументы командной строки и не размещается в переменных окружения процесса. Соответственно исключается возможность его получения посредством анализа списка выполняемых процессов, истории команд оболочки или содержимого временных файлов.

В процессе функционирования подсистема PAM запускает программный модуль оценки компрометации в виде отдельного локального процесса и передает введенный пользователем пароль через стандартный поток ввода. Взаимодействие между компонентами осуществляется исключительно средствами операционной системы и не требует использования сетевых соединений или удаленных вычислительных ресурсов.

Вычислительный модуль реализован как самостоятельный программный компонент, отделенный от подсистемы PAM. Такое разделение позволяет выполнять обновление моделей машинного обучения и модификацию алгоритмов анализа без изменения конфигурации PAM и без вмешательства в программный код системных компонентов операционной системы. Подсистема PAM в данном случае выполняет только функции интеграции и передачи данных, тогда как вся логика интеллектуального анализа реализована внутри разработанного программного модуля.

После завершения вычислений модуль формирует результат классификации и возвращает код завершения процесса. В случае если вероятность компрометации не превышает установленного порогового значения, программа завершается успешно и возвращает нулевой код завершения. Подсистема PAM интерпретирует

данный результат как успешное выполнение дополнительной проверки и продолжает обработку запроса на изменение пароля. Далее управление передается стандартным модулям Linux, выполняющим обновление хешированного представления пароля в системных файлах учета пользователей /etc/shadow.

Если вычисленная вероятность компрометации превышает установленный порог либо пароль классифицируется как принадлежащий к потенциально компрометируемому классу, программный модуль завершается с кодом ошибки. Полученный результат интерпретируется PAM как отказ в выполнении операции изменения учетных данных. В этом случае процедура смены пароля прерывается до момента обновления учетной записи, а пользователю возвращается сообщение о невозможности использования выбранного значения пароля.

На первом этапе после получения пароля внешний модуль выполняет преобразование пароля в набор признаков, на которых обучена модель машинного обучения. Он начинает свою работу с получения списков паролей. Алгоритм состоит из основного цикла, последовательно обрабатывающего весь полученные массив, и нескольких блоков валидации требованиям установленной парольной политики.

3.2.1 Реализация вычислительного модуля оценки вероятности компрометации аутентификационных данных

Вычислительный модуль оценки вероятности компрометации аутентификационных данных реализует разработанный интеллектуальный метод анализа паролей и выполняет полный цикл их обработки от момента получения нового пароля из подсистемы PAM до формирования итогового решения о возможности его использования. Внутренняя структура модуля состоит из четырех последовательно выполняемых функциональных блоков: блока предварительной обработки, блока извлечения признаков, блока классификации и блока принятия решения.

После запуска программного модуля подсистемой PAM новый пароль поступает на вход блока предварительной обработки. Основной задачей данного этапа является проверка корректности входных данных и контроль их соответствия

минимальным требованиям парольной политики. Блок предварительной проверки пароля используется для подготовки входных данных к дальнейшему интеллектуальному анализу. На этом этапе отсекаются варианты, которые явно не соответствуют минимальным требованиям безопасности, поэтому применение методов машинного обучения к ним не требуется (алг. 1).

Алгоритм 1. Предварительная обработка паролей

Вход: p
Выход: p_{valid}, R, Len, AC
1: $MinL = 8$
2: $MaxL = 18$
3: $Len \leftarrow size(p)$
4: **Если** $Len < MinL$ **или** $Len > MaxL$
5: $R \leftarrow PAM\_REJECT$
6: **Возвратить** R
7: **Конец условия**
8: $AC \leftarrow calculate\_AC(p)$
9: **Если** $AC < 3$
10: $R \leftarrow PAM\_REJECT$
11: **Возвратить** R
12: **Конец условия**
13: $p_{valid} \leftarrow p$
14: $R \leftarrow PAM\_SUCCESS$
15: **Возвратить** p_{valid}, R, Len, AC

Как показано в алгоритме 1, после получения пароля выполняется проверка его основных характеристик. Сначала определяется количество задействованных алфавитов AC. В данной реализации учитываются строчные и прописные латинские буквы, цифры и специальные символы. Если число используемых групп символов меньше установленного минимума, пароль сразу признаётся недостаточно надёжным, и дальнейшая обработка для него не выполняется.

Затем проверяется допустимость использованных символов. Для контроля входных данных применяется регулярное выражение, определяющее множество разрешенных символов. Использование данного этапа позволяет исключить обработку некорректных данных и обеспечить единообразие формирования признакового пространства.

На заключительном этапе предварительной обработки вычисляется длина пароля Len и выполняется ее сравнение с минимально допустимым значением,

установленным парольной политикой. Только после успешного прохождения всех проверок пароль передается в следующий функциональный блок вычислительного модуля.

После завершения этапа предварительной обработки выполняется формирование признакового описания анализируемого пароля. Основной задачей данного этапа является преобразование символьного представления пароля в числовой вектор фиксированной размерности, используемый моделью машинного обучения (алг. 2).

Алгоритм 2. Формирование признакового описания

Вход: p, Len, AC
Выход: F
1: $Uniq \leftarrow calculate\_Uniq(p)$
2: $N \leftarrow calculate\_N(p)$
3: $S \leftarrow calculate\_S(p)$
4: $Dist \leftarrow calculate\_Dist(p)$
5: $Freq \leftarrow calculate\_Freq(p)$
6: $F \leftarrow \{Len, AC, Uniq, N, S, Dist, Freq\}$
7: **Возвратить** F

Как показано в алгоритме 2, после получения валидного пароля последовательно рассчитываются признаки, характеризующие его структуру и статистические свойства. На данном этапе определяются значения признаков, используемых моделью машинного обучения при оценке вероятности компрометации аутентификационных данных.

В качестве признаков используются длина пароля Len, количество задействованных алфавитов AC, относительное количество уникальных символов Uniq, относительное количество специальных символов S, относительное количество цифр N, среднее расстояние между символами по их расположению на клавиатуре Dist, а также частота смены используемых алфавитов Freq.

Признаки Len и AC определяются на этапе предварительной обработки и далее используются при формировании итогового признакового вектора. Относительное количество уникальных символов характеризует степень повторяемости символов внутри пароля. Относительное количество специальных символов и цифр отражает структуру используемых групп символов. Среднее

расстояние между символами по их расположению на клавиатуре позволяет учитывать пространственные закономерности формирования паролей, возникающие в процессе пользовательского ввода. Частота смены алфавитов характеризует интенсивность переходов между различными группами символов и косвенно отражает сложность структуры пароля.

Результатом работы блока извлечения признаков является числовой вектор фиксированной размерности, описывающий анализируемый пароль. Полученный вектор передается в блок классификации.

Обученная модель хранится локально на вычислительном узле и загружается программным модулем при запуске. Результатом работы классификатора является оценка принадлежности анализируемого пароля к одному из двух классов: допустимых либо потенциально компрометируемых паролей.

Полученный результат поступает в блок принятия решения. Основной задачей данного блока является преобразование результата классификации в формат, пригодный для последующего использования подсистемой PAM. Если пароль классифицируется как допустимый и вычисленная вероятность компрометации не превышает установленного порогового значения, формируется положительный результат проверки. В противном случае пароль относится к классу потенциально компрометируемых и формируется отрицательный результат проверки.

На завершающем этапе вычислительный модуль преобразует полученное решение в код завершения процесса операционной системы. В случае успешного прохождения проверки возвращается нулевой код завершения, который интерпретируется подсистемой PAM как успешное выполнение дополнительной проверки. В случае выявления потенциально компрометируемого пароля возвращается ненулевой код завершения, приводящий к прекращению процедуры изменения учетных данных. Таким образом обеспечивается интеграция результатов интеллектуального анализа паролей в стандартный механизм управления учетными записями операционных систем семейства Linux.

Для обеспечения обратной связи пользователю блок принятия решения формирует не только бинарный результат проверки, но и диагностический код

причины отказа. Такой код определяется на основании этапа, на котором пароль был отклонен: недостаточное количество задействованных алфавитов, наличие недопустимых символов, недостаточная длина пароля либо отнесение пароля моделью машинного обучения к классу потенциально компрометируемых.

Основным результатом работы вычислительного модуля для подсистемы PAM является код завершения процесса. Нулевой код завершения соответствует успешному прохождению проверки, ненулевой код — отказу в использовании выбранного пароля. При этом диагностическое сообщение может использоваться как дополнительный результат работы модуля. В рассматриваемой реализации оно формируется локально и может быть передано вызывающему приложению через стандартный поток вывода при использовании параметра stdout модуля pam_exec.so либо зафиксировано в системном журнале для последующего анализа администратором.

Следует учитывать, что отображение пользовательского сообщения зависит не только от разработанного вычислительного модуля, но и от поведения прикладной программы, инициировавшей обращение к PAM. Поэтому гарантированным механизмом управления процедурой смены пароля является возврат кода завершения, тогда как текстовое сообщение рассматривается как вспомогательный механизм информирования пользователя или администратора.

3.2.2 Выбор модели машинного обучения и результаты классификации

Для оценки эффективности разработанного вычислительного модуля и выбора алгоритма машинного обучения, обеспечивающего наилучшее качество классификации паролей, было проведено экспериментальное исследование. Эксперименты выполнялись на персональном компьютере с процессором AMD Ryzen 7 5700X (8 вычислительных ядер, 16 потоков), 32 ГБ оперативной памяти под управлением операционной системы Windows 10. Программная реализация была выполнена на языке Python версии 3.10 в среде разработки PyCharm 2023.3 с использованием библиотек openpyxl, xlwt и стандартных математических модулей языка.

Для формирования обучающей выборки использовались два класса

паролей: потенциально компрометируемые и надежные. В качестве источника потенциально компрометируемых паролей использовалась база данных утекших паролей rockyou.txt. После применения предварительной фильтрации по минимальной длине пароля и минимальному количеству используемых алфавитов было сформировано множество, содержащее около 385 тысяч паролей. Дополнительно из выборки были исключены записи, содержащие символы, не входящие в множество допустимых значений, используемых в разработанном методе.

Для формирования класса надежных паролей использовались специализированные сервисы генерации паролей: Passwordsgenerator [106], Passwords-Generator [107] и Kaspersky [108]. Генерация выполнялась с использованием нескольких независимых источников, что позволило снизить влияние особенностей конкретного алгоритма генерации. С помощью этих сервисов было сгенерировано около четырех миллионов паролей, равномерно распределенных по длине: от 10 до 18 символов и по количеству используемых букв: от 3 до 4 во всех пяти возможных комбинациях. Учитывая, что в процессе генерации могли возникать комбинации, в которых используются только символы одной или двух букв, этот набор данных также был подвергнут фильтрации. Для равномерного распределения по количеству надежных и простых паролей из полученного списка надежных паролей были выбраны те же 385 тысяч комбинаций символов. Суммируя все шесть наборов, мы получили окончательный список, состоящий из около 770 000 паролей.

Полученный набор данных использовался для обучения и тестирования моделей машинного обучения, предназначенных для классификации паролей на допустимые и потенциально компрометируемые.

Для выбора алгоритма классификации было проведено сравнительное исследование нескольких моделей машинного обучения. В эксперименте рассматривались дерево решений (Decision Tree), случайный лес (Random Forest), классификатор Naive Bayes и алгоритм градиентного бустинга XGBoost. Сравнение выполнялось по показателям точности классификации, коэффициенту ошибок, коэффициенту согласия Cohen’s Kappa и времени выполнения. В таблице 3.8

представлены результаты оценки метрик качества рассматриваемых алгоритмов машинного обучения.

Таблица 3.8 – Сравнение алгоритмов машинного обучения для классификации паролей

Algorithm	Password Type	True Positive	False Positive	True Negative	False Negative	Overall Accuracy	Overall, Error	Cohen's Kappa	Work Time, s
XGBoost	0	0.966	0.034	0.978	0.022	0.972	0.028	0.943	6.56
	1	0.978	0.022	0.966	0.034				
Random forest	0	0.965	0.035	0.978	0.022	0.971	0.029	0.942	13.70
	1	0.978	0.022	0.965	0.035				
Naive Bayes	0	0.320	0.611	1.000	0.000	0.694	0.306	0.389	0.58
	1	1.000	0.000	0.320	0.611				
Decision Tree	0	97.40	2.60	98.39	1.61	0.979	0.213	0.957	14.03
	1	98.39	1.61	97.40	2.60				

Результаты эксперимента показывают, что использование классификатора Naive Bayes для решения рассматриваемой задачи является нецелесообразным. Несмотря на высокие показатели распознавания одного из классов, модель продемонстрировала значительное количество ошибок при определении потенциально компрометируемых паролей, что недопустимо для задач обеспечения информационной безопасности.

Модели Random Forest и XGBoost продемонстрировали существенно более высокое качество классификации. Наилучшее время работы показал алгоритм XGBoost, однако максимальная точность была достигнута при использовании дерева решений. Поскольку в рассматриваемой задаче критически важным является снижение вероятности пропуска потенциально компрометируемых паролей, приоритет был отдан точности классификации, а не времени обучения модели. По этой причине в качестве основного алгоритма был выбран классификатор Decision Tree.

После выбора модели была выполнена настройка ее параметров. Для построения дерева решений использовались критерии разделения на основе индекса Gini и информационного выигрыша. Построение модели выполнялось без применения процедуры постобрезки, что позволило сохранить максимальную

выразительную способность классификатора. Для ограничения чрезмерного дробления дерева минимальное количество объектов в листовом узле было установлено равным двум. Дополнительно использовалось усреднение порогов разделения для числовых признаков, а обучение выполнялось с использованием восьми вычислительных потоков.

Качество полученной модели первоначально оценивалось на основе матрицы ошибок, представленной на рисунке 3.16.

	0 (Predicted)	1 (Predicted)	
0 (Target)	107,839	2883	97.40%
1 (Target)	1611	98,285	98.39%
	98.53%	97.15%	

Рисунок 3.16 – Матрица ошибок модуля классификации паролей

Анализ матрицы ошибок показывает, что около 2,6 % потенциально компрометируемых паролей были ошибочно отнесены к допустимым. Для задач контроля качества аутентификационных данных данный тип ошибки является наиболее критичным, поскольку приводит к пропуску потенциально уязвимых учетных записей. Ошибки ложного срабатывания составили около 1,6 %, что приводит лишь к необходимости повторного выбора пользователем нового пароля и не оказывает непосредственного влияния на безопасность системы.

Полученные результаты свидетельствуют о высокой эффективности разработанного классификатора. Общая точность модели составила 97,87 %, что указывает на корректную классификацию подавляющего большинства паролей тестовой выборки. Коэффициент ошибок составил 2,13 %, а значение Cohen’s Kappa достигло 0,957. Такой результат указывает на высокий уровень согласованности между предсказаниями классификатора и фактическими метками классов.

Для дополнительной проверки качества классификатора была построена ROC-кривая, представленная на рисунке 3.17. Площадь под ROC-кривой составила 0,987, что говорит о высокой разделяющей способности модели. Анализ кривой показал, что рациональным является порог 0,708: при нём достигаются TPR = 0,956 и FPR = 0,019. Полученные значения подтверждают, что разработанный модуль

способен различать допустимые и потенциально компрометируемые пароли при низком уровне ошибок классификации.

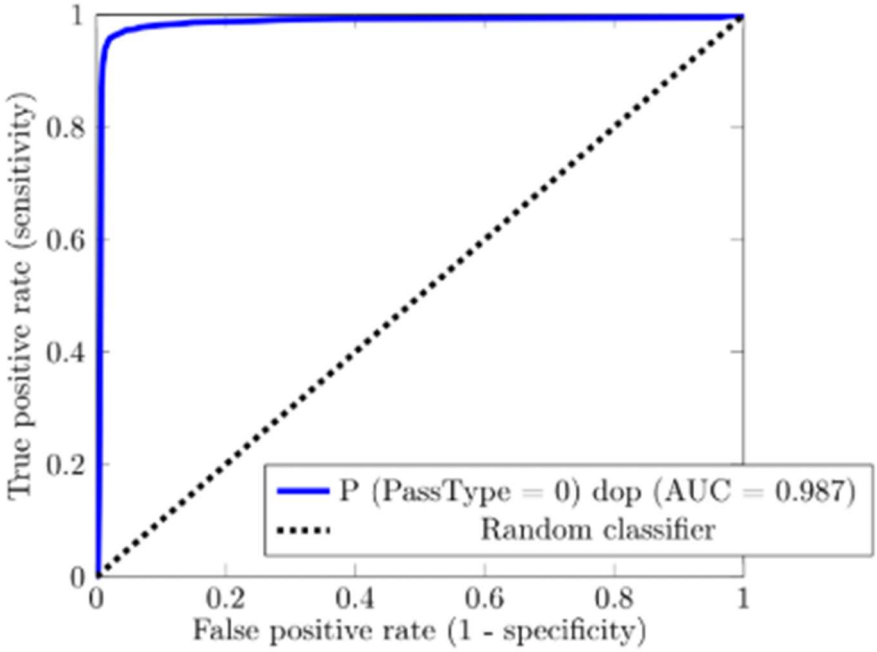


Рисунок 3.17 – ROC-кривая классификатора паролей

Таким образом, проведенное экспериментальное исследование подтвердило эффективность предложенного метода оценки вероятности компрометации аутентификационных данных и обосновало выбор модели Decision Tree в качестве основного классификатора разработанного программного комплекса.

3.3 Программная реализация метода параллельного вывода

Программная реализация метода потокового вывода выполнена на языке Python с использованием библиотек PyTorch и HuggingFace Transformers. Реализованный программный модуль обеспечивает параллельную обработку потоковых системных событий и включает механизмы адаптивного формирования вычислительных пакетов, динамического выравнивания длины последовательностей и группировки событий по вычислительной сложности.

Общая архитектура программной реализации включает этапы загрузки событий, токенизации, оценки длины последовательностей, распределения событий по вычислительным группам, буферизации, формирования пакетов и последующего выполнения вывода модели. В качестве среды исполнения используется CPU-ориентированная конфигурация с управлением числом

вычислительных потоков посредством механизма torch.set_num_threads(), что позволяет использовать возможности параллельного выполнения операций линейной алгебры.

Реализация адаптивной пакетной обработки основана на использовании механизма буферизации событий. После токенизации каждое событие помещается в один из буферов, соответствующих диапазону длины последовательности. Для каждой группы поддерживается отдельная очередь обработки и момент последнего выполнения вывода. Запуск вычислений осуществляется при достижении одного из двух условий: заполнении пакета до заданного размера BATCH_SIZE либо превышении временного порога ожидания TIMEOUT_MS. Такой подход обеспечивает баланс между пропускной способностью системы и задержкой обработки событий (алг. 3).

С целью уменьшения вычислительных затрат реализован механизм динамического выравнивания длины последовательностей. В отличие от фиксированного значения, предполагающего дополнение всех последовательностей до заранее заданного максимума, применяется динамическое выравнивание внутри текущего вычислительного пакета с использованием DataCollatorWithPadding. В результате последовательности дополняются только до максимальной длины внутри текущего пакета, что уменьшает число фиктивных токенов и снижает объём выполняемых вычислений (алг. 4).

Алгоритм 3. Подготовка вычислительного пакета к выводу модели

Вход: $B = \{s_1, s_2, \dots, s_n\}$
Выход: Y
1: $maxLen \leftarrow \max\_length(B)$
2: **Цикл для каждого** $s_i \in B$ **выполнить**
3: $s_i \leftarrow pad(s_i, maxLen)$
4: **Конец цикла**
5: $X \leftarrow create\_tensor(B)$
6: $logits \leftarrow inference(B)$
7: $Y \leftarrow predict(logits)$
8: **Возвратить** Y

С целью уменьшения вычислительных затрат реализован механизм динамического выравнивания длины последовательностей. В отличие от

фиксированного значения, предполагающего дополнение всех последовательностей до заранее заданного максимума, применяется динамическое выравнивание внутри текущего вычислительного пакета с использованием DataCollatorWithPadding. В результате последовательности дополняются только до максимальной длины внутри текущего пакета, что уменьшает число фиктивных токенов и снижает объём выполняемых вычислений (алг. 4).

Алгоритм 4. Обработка вычислительного пакета
Вход: $A = \{s_1, s_2, \dots, s_m\}, BATCH\_SIZE, timeout$
Выход: $B = \{s_1, s_2, \dots, s_n\}$
1: **Цикл для каждого** $s_i \in A$ **выполнить**
2: $Len_i \leftarrow size(s_i)$
3: $G \leftarrow determine\_group(s_i, Len_i)$
4: $Buffer_G \leftarrow Buffer_G \cup \{s_i\}$
5: **Если** $size(Buffer_G) \geq BATCH\_SIZE$
6: $B \leftarrow B \cup Buffer_G$
7: Возвратить B
8: **Иначе если** $elapsed\_time(Buffer_G) \geq timeout$
9: $B \leftarrow B \cup Buffer_G$
10: Возвратить B
11: **Иначе**
12: *Ожидать поступления новых событий*
13: **Конец условия**
14: **Конец цикла**
15: **Возвратить** B

Дополнительным механизмом оптимизации является группировка событий по вычислительной сложности. Для этого после токенизации вычисляется длина последовательности в токенах, а события распределяются по диапазонам длины (1–32, 33–64, 65–128 токенов). Такое разделение уменьшает неоднородность последовательностей внутри вычислительного пакета и позволяет сократить избыточные вычисления, возникающие при совместной обработке коротких и длинных сообщений.

Функция выполнения вывода реализована в виде отдельной процедуры обработки пакета, осуществляющей динамическое формирование тензоров, запуск модели без вычисления градиентов torch.no_grad(), получение логитов и

вычисление итоговых предсказаний. Результаты классификации сохраняются для последующего расчёта характеристик производительности и качества модели.

3.3.1 Оценка эффективности метода параллельного вывода модели

Для проведения эксперимента по предложенным в разделе 2.3 методам была задействована следующая установка: компьютер под управлением 64-разрядной операционной системой Windows 11, процессор AMD Ryzen 7 5700X 8-Core Processor (3.40 GHz) с возможность управления 16 логическими потоками одновременно, 32 Гб оперативной памяти. Программная реализация выполнялась с использованием языка программирования Python v3. Используемые библиотеки включают: scikit-learn v1.7.0, transformers v4.53.2, torch v2.7.1+cu118, numpy v2.3.1, seaborn v0.13.2, tqdm v4.67.1, matplotlib v3.10.3, pandas v2.3.1, datasets v4.0.0, а также iterative-stratification v0.1.9.

Моделью для проведения тестирования была выбрана ранее подготовленная в разделе 3.1.7 дистиллированая модель TinyBERT-4L для бинарной классификации. Тестирование проводилось на том же наборе данных Linux-APT-Dataset-2024 [80].

В качестве основных критериев оценки использовались: общее время вывода, характеризующее длительность обработки потока событий; пропускная способность системы, определяемая числом обработанных событий в секунду; а также число сформированных вычислительных пакетов. Размер пакета варьировался в диапазоне от 1 до 64 событий.

Замер метрик качества оптимизации осуществлялся путем запуска модели на указанном ранее процессоре с использованием всех доступных логических потоков и ядер.

В первом случае исследовалось влияние размера пакета при использовании фиксированного ограничения длины входной последовательности. Для каждого системного события выполнялось выравнивание до максимального значения 128 токенов независимо от фактического размера события. Таким образом, вычислительная нагрузка модели определялась заранее установленным ограничением длины последовательности, а все события внутри пакета обладали

одинаковой размерностью. Полученные результаты представлены в таблице 3.9.

Таблица 3.9 – Метрики качества без введения механизмов оптимизации

Методы оптимизации	Размер пакета	Среднее время обработки набора данных, сек	Пропускная способность, событий/сек
Без оптимизации	1	42.586	179.26
	2	30.529	250.05
	4	24.037	315.60
	6	22.489	339.46
	8	21.619	353.12
	10	21.297	358.46
	12	20.894	365.37
	16	21.348	357.61
	20	21.776	350.58
	26	21.748	351.02
	32	22.740	335.70
	64	25.419	300.33

Видно, что увеличение размера пакета сопровождается снижением общего времени вывода и ростом пропускной способности интеллектуальной системы. Это объясняется уменьшением относительного влияния накладных расходов вычислительного процесса. При обработке каждого отдельного события модель инициирует последовательность операций, связанных с подготовкой входных тензоров, передачей данных и запуском вычислений. При увеличении размера пакета указанные накладные расходы распределяются сразу между несколькими событиями, вследствие чего среднее время обработки одного сообщения уменьшается.

Однако в данном случае фиксированная длина последовательностей приводит к выполнению большого объёма избыточных вычислений, поскольку короткие системные события обрабатываются как последовательности максимальной длины независимо от фактического количества токенов.

Для устранения указанного ограничения во втором случае было применено динамическое выравнивание длины последовательностей внутри пакета. Каждое событие было токенизировано без предварительного дополнения до

фиксированного максимального размера, а выравнивание выполнялось только до длины наиболее длинного события внутри конкретного пакета. Полученные результаты представлены в таблице 3.10.

Таблица 3.10 – Метрики качества с механизмом определения динамического размера пакета

Методы оптимизации	Размер пакета	Среднее время обработки набора данных, сек	Пропускная способность, событий/сек
Динамический размер пакета	1	32.237	236.81
	2	22.030	346.53
	4	16.178	471.87
	6	14.482	527.12
	8	13.720	556.42
	10	13.365	571.20
	12	13.129	581.46
	16	13.172	579.57
	20	13.320	573.11
	26	13.748	555.29
	32	14.158	539.22
	64	16.692	457.35

Полученные зависимости демонстрируют сохранение общей закономерности изменения производительности: увеличение размера пакета сопровождается уменьшением общего времени вывода и ростом пропускной способности системы. Вместе с тем абсолютные показатели оказываются лучше по сравнению с первым экспериментом. Повышение производительности обусловлено уменьшением вычислительной избыточности. В отличие от фиксированного выравнивания, модель перестаёт выполнять операции над фиктивными токенами для коротких событий.

Наиболее выраженный прирост производительности достигается при малых и средних размерах пакетов, в которых влияние избыточных вычислений оказывается наиболее заметным. Одновременно сохраняется характерная область насыщения: после достижения рационального диапазона размеров пакета дальнейшее увеличение числа событий приводит к постепенному замедлению

прироста производительности.

Следует отметить, что эффективность динамического выравнивания ограничивается неоднородностью состава вычислительного пакета. При совместной обработке коротких и длинных последовательностей длина выравнивания определяется наиболее длинным событием, вследствие чего часть преимуществ данного метода утрачивается.

Для устранения данного ограничения был реализован механизм предварительной группировки событий по их длине токенизации. Он заключался в том, что события предварительно разделялись на группы коротких, средних и длинных последовательностей: 1–32, 33–64 и 65–128 токенов соответственно. Формирование пакетов выполнялось только внутри соответствующей группы, что позволяло уменьшить разброс длин последовательностей и снизить объём вычислений, связанных с выравниванием.

Дополнительно было введено ограничение формирования пакета по времени. Оно необходимо для предотвращения чрезмерного рост задержки обработки пакета. В случае если пакет не заполнялся полностью, накопленные события передавались в модель в том количестве, которое уже было добавлено в пакет, после достижения временного порога ожидания, значение которого было установлено в 50 мс. Полученные результаты представлены в таблице 3.11.

Таблица 3.11 – Метрики качества с предлагаемым механизмом оптимизации

Методы оптимизации	Размер пакета	Среднее время обработки набора данных, сек	Пропускная способность, событий/сек
Динамический размер пакета, группировка событий по длине	1	25.398	300.58
	2	17.392	438.93
	4	12.375	616.91
	6	10.759	709.57
	8	9.977	765.15
	10	9.780	780.55
	12	9.384	813.56
	16	9.320	819.19
	20	9.320	819.10
	26	9.389	813.12
	32	9.546	799.68
	64	10.466	729.40

Видно, что использование пакетной обработки позволяет уменьшить относительное влияние накладных расходов запуска модели. Динамическое выравнивание длины последовательностей сокращает объём обязательных вычислений, который возникает при использовании фиксированного размера входных данных. Дополнительная группировка событий по длине уменьшает неоднородность пакетов и делает вычислительный процесс более равномерным.

Совместное применение предложенных методов оптимизации увеличило общую пропускную способность в 2,23 раза относительно первого случая и в 1,4 раза относительно второго. Во всех экспериментах наблюдалась одна и та же зависимость: с ростом размера пакета пропускная способность повышалась, а время обработки уменьшалось до достижения области насыщения. После этого дальнейшее увеличение пакета давало уже меньший эффект из-за ограничений вычислительных ресурсов. Данная зависимость представлена на рисунке 3.18.

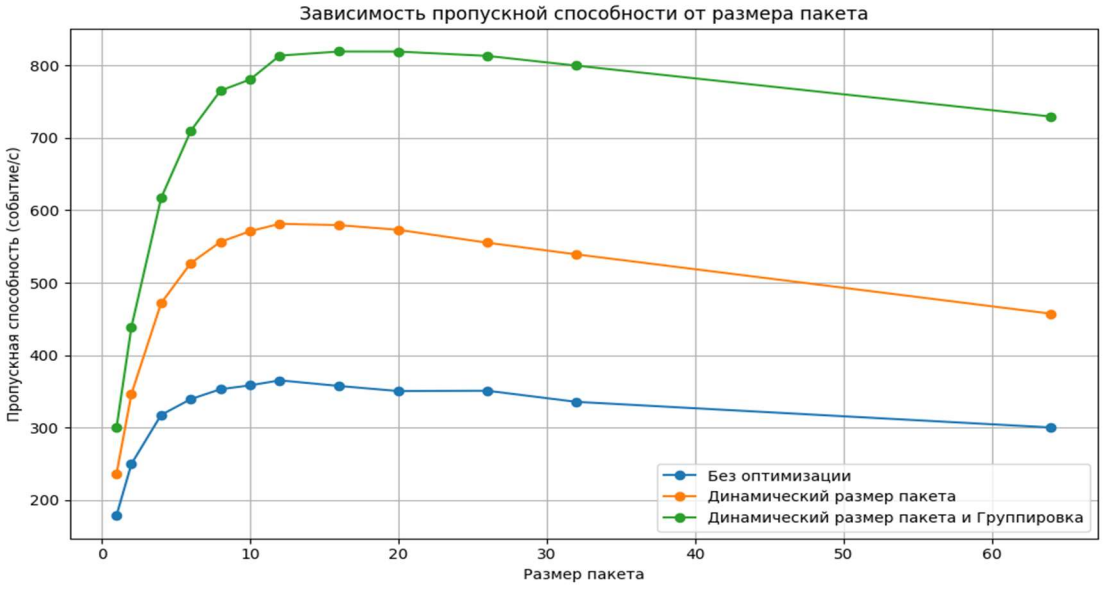


Рисунок 3.18 – Зависимость пропускной способности от размера пакета

Из полученных данных видно, что оптимальный размер пакета – 12 токенизированных событий. После достижения этого значения дальнейшее его увеличение приводит к стабилизации и последующему плавному снижению эффективности. Похожая ситуация наблюдается и в экспериментах, приведенных в работе [37]. В них авторы также подчеркивают, что при постепенном повышении количества используемых каскадов в интервале небольших значений (порядка 1-10) производительность системы сначала временно возрастает, а затем после

преодоления этого интервала постепенно снижается. Это в свою очередь обусловлено ограничением вычислительных ресурсов используемого аппаратного оборудования, а также ростом накладных расходов обработки крупных пакетов.

Полученные результаты подтверждают возможность повышения эффективности потокового вывода интеллектуальных моделей анализа системных событий без изменения архитектуры модели и без снижения качества классификации, что делает предложенный подход применимым в системах мониторинга информационной безопасности, функционирующих в условиях с ограниченными вычислительными ресурсами и непрерывным поступлением потоков событий.

3.4 Выводы по третьей главе

В третьей главе разработан и реализован программный комплекс оценки вероятности компрометации аутентификационных данных, интегрированный в операционные системы семейства Linux посредством подсистемы Pluggable Authentication Modules (PAM). Проведенные экспериментальные исследования подтвердили возможность использования статистических характеристик случайности паролей для решения задачи их классификации. Установлено, что наилучшие результаты демонстрирует алгоритм Decision Tree Learner, обеспечивающий точность классификации 97.87%, значение ROC-AUC 0.987 и низкую долю ложноотрицательных решений, что подтверждает эффективность предложенного подхода для выявления потенциально компрометируемых паролей непосредственно на этапе их создания.

Также в главе разработан и реализован программный комплекс обнаружения и классификации вредоносной активности в системных событиях операционных систем семейства Linux. Проведенное экспериментальное моделирование на наборе данных Linux-MMA-Dataset-2024 подтвердило эффективность предложенного метода анализа событий. Показано, что применение дистилляции знаний и последующего квантования позволяет существенно сократить время обработки событий и требования к вычислительным ресурсам при сохранении высокого качества классификации. Полученные результаты

подтверждают возможность обнаружения вредоносной активности и определения тактик MITRE ATT&CK с высокой точностью для большинства представленных в наборе данных классов атак.

Для повышения эффективности эксплуатации разработанного программного комплекса реализован метод потоковой обработки системных событий. Экспериментально установлено, что совместное использование пакетной обработки, динамического выравнивания длины последовательностей и группировки событий по вычислительной сложности позволяет увеличить пропускную способность системы в 2.23 раза по сравнению с базовой схемой обработки и в 1.4 раза по сравнению с использованием только пакетной обработки. Определен оптимальный размер вычислительного пакета, обеспечивающий максимальную эффективность использования вычислительных ресурсов без снижения качества классификации.

Таким образом, результаты программной реализации и экспериментальных исследований подтвердили работоспособность предложенных методов и возможность их практического применения при создании интеллектуальных подсистем мониторинга информационной безопасности, функционирующих в условиях ограниченных вычислительных ресурсов.

Заключение

В рамках проведенного диссертационного исследования разработаны интеллектуальные методы, модели и программные средства формирования модулей обнаружения и классификации несанкционированных вторжений в операционные системы семейства Linux. Разработанные интеллектуальные методы позволяют не только расширить класс задач, решаемых с использованием языковых моделей, но и уменьшить время обработки данных без снижения уровня точности. Основными результатами диссертационного исследования являются:

1. Разработан метод выявления потенциально компрометируемых аутентификационных данных, позволяющий уточнить вероятность их компрометации, что обеспечивает возможность его применения в подсистемах контроля защищенности, аудита безопасности и управления доступом.

2. Разработан интеллектуальный метод анализа событий операционных систем семейства Linux на основе языковых моделей, позволяющий уменьшить время их обработки в среднем в 5 раз, что обеспечивает возможность его применения при создании программного обеспечения для центров мониторинга информационной безопасности.

3. Разработан метод параллельной потоковой обработки событий операционных систем семейства Linux, позволяющий уменьшить время обработки в среднем в 2 раза и повысить пропускную способность интеллектуальных подсистем мониторинга информационной безопасности на 123% без снижения точности классификации, что обеспечивает возможность его применения при создании программного обеспечения для центров мониторинга информационной безопасности.

Список литературы

1. Buczak A. L., Guven E. A survey of data mining and machine learning methods for cyber security intrusion detection //IEEE Communications surveys & tutorials. – 2015. – Т. 18. – №. 2. – С. 1153-1176.
2. Sommer R., Paxson V. Outside the closed world: On using machine learning for network intrusion detection //2010 IEEE symposium on security and privacy. – IEEE, 2010. – С. 305-316.
3. Vyšniūnas T. et al. Risk-based system-call sequence grouping method for malware intrusion detection //Electronics. – 2024. – Т. 13. – №. 1. – С. 206.
4. Сидоркина И. Г., Михалищев С. В. Совершенствование метода оценки стойкости пароля аутентификации пользователя компьютерных систем на основе использования известных уязвимостей //Программные продукты и системы. – 2024. – Т. 37. – №. 4. – С. 547-553.
5. Исследование утечек паролей в 2024 году: анализ угроз и рекомендации по защите. – Текст : электронный // Newsletter Radensa : [сайт]. – URL: https://newsletter.radensa.ru/archives/8808 (дата обращения: 10.02.2026).
6. Rusanov M. A. et al. Research of machine learning methods for detecting network attacks //Труды Института системного программирования РАН. – 2025. – Т. 37. – №. 4-2. – С. 147-174.
7. Rusanov M. A. et al. Detection of SQL injection attacks through the network logs using machine learning methods //Труды института системного программирования РАН. – 2025. – Т. 37. – №. 5. – С. 81-92.
8. Rusanov M. et al. Identification of exploited unreliable account passwords in the information infrastructure using machine learning methods //Big Data and Cognitive Computing. – 2024. – Т. 8. – №. 11. – С. 159.
9. Rusanov M., Babenko M., Lapina M. Optimization of Machine Learning Algorithms with Distillation and Quantization for Early Detection of Attacks in Resource-Constrained Systems //Big Data and Cognitive Computing. – 2025. – Т. 9. – №. 12. – С. 303.
10. Rusanov M. Криптографическая система на основе хаотической системы Лоренца / М.А. Русанов, С.В. Данилкин, П.И. Карасев, А.А.Л. Алмали

//Приборы и системы. Управление, контроль, диагностика. 2023, – №. 3. – С. 48-53.

11. Русанов М.А. Эффективные модели машинного обучения для раннего выявления угроз / М.А. Русанов, М.Г. Бабенко // Актуальные проблемы информационной безопасности: материалы XIII Всероссийской научно-практической конференции 18 декабря 2025 г. 2026, – С. 126-131.
12. Свидетельство о государственной регистрации программы для ЭВМ №2025696118 Российская Федерация. Программа обнаружения вредоносной активности в системных событиях и приложениях на базе нейронной сети : № 2025694833 : заявл. 01.12.2025 : опубл. 16.12.2025 / М. А. Русанов, М. А. Лапина, М. Г. Бабенко ; заявитель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет».
13. Свидетельство о государственной регистрации программы для ЭВМ №2025695537 Российская Федерация. Программа идентификации тактик вредоносной активности в системных событиях на основе нейронной сети : № 2025694747 : заявл. 01.12.2025 : опубл. 11.12.2025 / М. А. Русанов, М. А. Лапина, М. Г. Бабенко ; заявитель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет».
14. Alma T., Das M. L. Web application attack detection using deep learning //arXiv preprint arXiv:2011.03181. – 2020.
15. Khraisat A. et al. Survey of intrusion detection systems: techniques, datasets and challenges //Cybersecurity. – 2019. – Т. 2. – №. 1. – С. 1-22.
16. Gupta N., Traore I., de Quinan P. M. F. Automated event prioritization for security operation center using deep learning //2019 IEEE International Conference on Big Data (Big Data). – IEEE, 2019. – С. 5864-5872.
17. Pandey V. K. et al. A lightweight framework to secure IoT devices with limited resources in cloud environments //Scientific Reports. – 2025. – Т. 15. – №. 1. – С. 26009.

18. Thompson N. C. et al. The computational limits of deep learning //arXiv preprint arXiv:2007.05558. – 2020. – Т. 10. – №. 2.
19. Amgbara S. I., Akwiwu-Uzoma C., David O. Exploring lightweight machine learning models for personal internet of things (IOT) device security //ResearchGate Preprint. – 2024. – Т. 24.
20. Alasmary H. et al. SHELLCORE: Automating malicious IoT software detection using shell commands representation //IEEE Internet of Things Journal. – 2021. – Т. 9. – №. 4. – С. 2485-2496.
21. Trizna D. Shell language processing: Unix command parsing for machine learning //arXiv preprint arXiv:2107.02438. – 2021.
22. Ongun T. et al. Living-off-the-land command detection using active learning //Proceedings of the 24th international symposium on research in attacks, intrusions and defenses. – 2021. – С. 442-455.
23. Lin J., Guo Y., Chen H. Intrusion detection at scale with the assistance of a command-line language model //2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S). – IEEE, 2024. – С. 73-79.
24. Notaro P. et al. Command-line Risk Classification using Transformer-based Neural Architectures //arXiv preprint arXiv:2412.01655. – 2024.
25. Brodzik A. et al. Ransomware Detection Using Machine Learning in the Linux Kernel //arXiv preprint arXiv:2409.06452. – 2024.
26. Zhang S., Zhao P., An Z. Digraph-mmb: A directed graph-based multimodal model for multi-stage apt attack detection //Available at SSRN 5226068. – 2025.
27. Devlin J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding //Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). – 2019. – С. 4171-4186.
28. Guo H., Yuan S., Wu X. Logbert: Log anomaly detection via bert //2021 international joint conference on neural networks (IJCNN). – IEEE, 2021. – С. 1-8.

29. Hofmeyr S. A., Forrest S., Somayaji A. Intrusion detection using sequences of system calls //Journal of computer security. – 1998. – Т. 6. – №. 3. – С. 151-180.
30. Wang F. et al. LlamaIDS: Real-Time Detection Model of Zero-Day Intrusions Using Large Language Models //2025 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). – IEEE, 2025. – С. 768-769.
31. Xu Y. et al. PWAGAT: Potential Web attacker detection based on graph attention network //Neurocomputing. – 2023. – Т. 557. – С. 126725.
32. Tadhani J. R. et al. Securing web applications against XSS and SQLi attacks using a novel deep learning approach //Scientific Reports. – 2024. – Т. 14. – №. 1. – С. 1803.
33. Guo J. et al. Adaptive contrastive knowledge distillation for BERT compression //Findings of the Association for Computational Linguistics: ACL 2023. – 2023. – С. 8941-8953.
34. Five Steps for Frontier AI Security Readiness. – Текст : электронный // Crowdstrike : [сайт]. – URL: https://www.crowdstrike.com/en-us/ (дата обращения: 02.05.2026).
35. Wu W. et al. CSMC: A secure and efficient visualized malware classification method inspired by compressed sensing //Sensors. – 2024. – Т. 24. – №. 13. – С. 4253.
36. Silfa F., Arnau J. M., González A. E-BATCH: Energy-efficient and high-throughput RNN batching //ACM Transactions on Architecture and Code Optimization (TACO). – 2022. – Т. 19. – №. 1. – С. 1-23.
37. Characterizing and understanding deep neural network batching systems on GPUs / Yu F., Zhang H., Chen A., Wang X., Liang X. //BenchCouncil Transactions on Benchmarks, Standards and Evaluations. – 2023. – Т. 3. – №. 4. – С. 100151.
38. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale / R.Y. Aminabadi, S. Rajbhandari, A.A. Awan //SC22: International Conference for High Performance Computing, Networking, Storage and Analysis. – IEEE, 2022. – С. 1-15.

39. Yadav T., Rao A. M. Technical aspects of cyber kill chain //International symposium on security in computing and communication. – Cham : Springer International Publishing, 2015. – С. 438-452.
40. Матрица MITRE ATT&CK, Cyber Kill Chain, UKC. – Текст : электронный // Blue Team Cookbook : [сайт]. – URL: https://vasilisa-l.gitbook.io/blue-team-cookbook/soc/mitre-attack (дата обращения: 02.05.2026).
41. Assante M. J., Lee R. M. The industrial control system cyber kill chain //SANS Institute InfoSec Reading Room. – 2015. – Т. 1. – №. 1. – С. 2.
42. Kryukov R. et al. Mapping the security events to the MITRE ATT &CK attack patterns to forecast attack propagation //International Workshop on Attacks and Defenses for Internet-of-Things. – Cham : Springer Nature Switzerland, 2022. – С. 165-176.
43. Singh C., Dhanraj M., Huang K. KillChainGraph: ML framework for predicting and mapping ATT&CK techniques //arXiv preprint arXiv:2508.18230. – 2025.
44. Ligh M. H. et al. The art of memory forensics: detecting malware and threats in windows, linux, and Mac memory. – John Wiley & Sons, 2014.
45. Bridges R. A. et al. A survey of intrusion detection systems leveraging host data //ACM computing surveys (CSUR). – 2019. – Т. 52. – №. 6. – С. 1-35.
46. Sen Ö. et al. Simulation of multi-stage attack and defense mechanisms in smart grids //International Journal of Critical Infrastructure Protection. – 2025. – Т. 48. – С. 100727.
47. Zhang Y., Monrose F., Reiter M. K. The security of modern password expiration: An algorithmic framework and empirical analysis //Proceedings of the 17th ACM conference on Computer and communications security. – 2010. – С. 176-186.
48. Iyer K. I. From Logs to Intelligence: Leveraging Data Science for Service Account Monitoring //Computer Fraud and Security. – 2025. – С. 1202-1209.
49. NIST Special Publication 800-63B. – Текст : электронный // NIST : [сайт]. – URL: https://pages.nist.gov/800-63-3/sp800-63b.html (дата обращения: 28.08.2024).
50. AI and Machine Learning in Cybersecurity – How They Will Shape the Future. – Текст : электронный // Kaspersky : [сайт]. – URL:

https://www.kaspersky.co.uk/resource-center/definitions/ai-cybersecurity (дата обращения: 28.08.2024).

51. Musser M., Garriott A. Machine learning and cybersecurity //Center for Security and Emerging Technology: Washington, DC, USA. – 2021.
52. Sarkar S., Nandan M. Password strength analysis and its classification by applying machine learning based techniques //2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA). – IEEE, 2022. – С. 1-5.
53. Farooq U. Real time password strength analysis on a web application using multiple machine learning approaches //Int J Eng Res Technol (IJERT). – 2020. – Т. 9. – №. 12. – С. 359-364.
54. Mo Y. et al. Password complexity prediction based on roberta algorithm //Applied Science and Engineering Journal for Advanced Research. – 2024. – Т. 3. – №. 3. – С. 1-5.
55. Darbutaitė E., Stefanovič P., Ramanauskaitė S. Machine-learning-based password-strength-estimation approach for passwords of Lithuanian context //Applied Sciences. – 2023. – Т. 13. – №. 13. – С. 7811.
56. Belikov V. V., Prokuronov I. A. Password strength verification based on machine learning algorithms and LSTM recurrent neural networks //Russian Technological Journal. – 2023. – Т. 11. – №. 4. – С. 7-15.
57. Dong Q. et al. Pii-psm: a new targeted password strength meter using personally identifiable information //International Conference on Security and Privacy in Communication Systems. – Cham : Springer Nature Switzerland, 2022. – С. 648-669.
58. Xie Z. et al. Modified Password Guessing Methods Based on TarGuess-I //Wireless Communications and Mobile Computing. – 2020. – Т. 2020. – №. 1. – С. 8837210.
59. Xie Z. et al. A new targeted password guessing model //Australasian conference on information security and privacy. – Cham : Springer International Publishing, 2020. – С. 350-368.

60. Cui X. et al. A password strength evaluation algorithm based on sensitive personal information //2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). – IEEE, 2020. – С. 1542-1545.
61. Xu M., Han W. An explainable password strength meter addon via textual pattern recognition //Security and Communication Networks. – 2019. – Т. 2019. – №. 1. – С. 5184643.
62. Linghu Y., Li X., Zhang Z. Deep learning vs. traditional probabilistic models: Case study on short inputs for password guessing //International Conference on Algorithms and Architectures for Parallel Processing. – Cham : Springer International Publishing, 2019. – С. 468-483.
63. Ji S. et al. Password correlation: Quantification, evaluation and application //IEEE INFOCOM 2017-IEEE Conference on Computer Communications. – IEEE, 2017. – С. 1-9.
64. Tran L. et al. A survey on password guessing //arXiv preprint arXiv:2212.08796. – 2022.
65. Lee K., Sjöberg S., Narayanan A. Password policies of most top websites fail to follow best practices //Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022). – 2022. – С. 561-580.
66. Password Policy Best Practices 2023. – Текст : электронный // Metacompliance : [сайт]. – URL: https://www.metacompliance.com/blog/cyber-security-awareness/password-policy-best-practices (дата обращения: 28.08.2024).
67. Гуфан Ю. К. и др. Оценка стойкости парольных фраз к методам подбора //Открытое образование. – 2011. – №. 2-2. – С. 127-130.
68. Password policy recommendations for Microsoft 365 passwords. – Текст : электронный // Microsoft : [сайт]. – URL: https://learn.microsoft.com/en-us/microsoft-365/admin/misc/password-policy-recommendations?view=o365-worldwide (дата обращения: 28.08.2024).
69. Enforce password policies for your devices. – Текст : электронный // Apple : [сайт]. – URL: https://support.apple.com/ru-ru/guide/deployment/depeadc2af0b/web (дата обращения: 28.08.2024).

70. Password policy. – Текст : электронный // IBM : [сайт]. – URL: https://www.ibm.com/docs/en/i/7.3?topic=security-password-policy (дата обращения: 28.08.2024).
71. Cisco IOS XE Catalyst SD-WAN Qualified Command Reference Guide. – Текст : электронный // Cisco : [сайт]. – URL: https://www.cisco.com/c/en/us/td/docs/routers/sdwan/command/iosxe/qualified-cli-command-reference-guide/m-aaa-commands.html (дата обращения: 28.08.2024).
72. Florencio D., Herley C. A large-scale study of web password habits //Proceedings of the 16th international conference on World Wide Web. – 2007. – С. 657-666.
73. Kolmogorov A. N., Uspenskii V. A. Algorithms and randomness //Theory of Probability & Its Applications. – 1988. – Т. 32. – №. 3. – С. 389-412.
74. Ntantogian C., Malliaros S., Xenakis C. Evaluation of password hashing schemes in open source web platforms //Computers & Security. – 2019. – Т. 84. – С. 206-224.
75. Menezes A. J., Van Oorschot P. C., Vanstone S. A. Handbook of applied cryptography. – CRC press, 2018.
76. Kaliski B., Rusch A. RFC 8018: PKCS# 5: Password-based cryptography specification version 2.1. – 2017.
77. Ionaşcu C. C., Treur J., Roelofsma P. H. M. P. Cascade Cyber Risk Management for Password Spray Attacks: Adaptive Network Analysis of AI Coaching //Biologically Inspired Cognitive Architectures Meeting. – Cham : Springer Nature Switzerland, 2025. – С. 265-287.
78. Alkhwaja I. et al. Password cracking with brute force algorithm and dictionary attack using parallel programming //Applied Sciences. – 2023. – Т. 13. – №. 10. – С. 5979.
79. Keneilwe Zuva. “Exploring Password Cracking Techniques: Understanding the Threat Landscape". Acta Scientific Computer Sciences 5.12 (2023): 07-13.
80. Karim S. S. et al. Advanced Persistent Threat (APT) and intrusion detection evaluation dataset for linux systems 2024 //Data in Brief. – 2024. – Т. 54. – С. 110290.

81. Myneni S. et al. Unraveled—A semi-synthetic dataset for Advanced Persistent Threats //Computer Networks. – 2023. – Т. 227. – С. 109688.
82. Athmane M. M. B. et al. Building a novel Graph Neural Networks-based model for efficient detection of Advanced Persistent Threats. – 2023.
83. Pivarníková M., Sokol P., Bajtoš T. Early-stage detection of cyber attacks //Information. – 2020. – Т. 11. – №. 12. – С. 560.
84. Mehmood M. et al. Privilege escalation attack detection and mitigation in cloud using machine learning //IEEe Access. – 2023. – Т. 11. – С. 46561-46576.
85. Thomas C., Sharma V., Balakrishnan N. Usefulness of DARPA dataset for intrusion detection system evaluation //Data Mining, Intrusion detection, information assurance, and data networks security 2008. – SPIE, 2008. – Т. 6973. – С. 164-171.
86. Manzoor E., Milajerdi S. M., Akoglu L. Fast memory-efficient anomaly detection in streaming heterogeneous graphs //Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. – 2016. – С. 1035-1044.
87. Shrestha M. et al. Provsec: Cybersecurity system provenance analysis benchmark dataset //2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA). – IEEE, 2023. – С. 352-357.
88. Alsaheel A. et al. {ATLAS}: A sequence-based learning approach for attack investigation //30th USENIX security symposium (USENIX security 21). – 2021. – С. 3005-3022.
89. Han X. et al. Unicorn: Runtime provenance-based detector for advanced persistent threats //arXiv preprint arXiv:2001.01525. – 2020.
90. Myneni S. et al. DAPT 2020-constructing a benchmark dataset for advanced persistent threats //International workshop on deployable machine learning for security defense. – Cham : Springer International Publishing, 2020. – С. 138-163.
91. Unicode Normalization Forms. – Текст : электронный // Unicode : [сайт]. – URL: https://unicode.org/reports/tr15/ (дата обращения: 02.05.2026).
92. Song X. et al. Fast wordpiece tokenization //Proceedings of the 2021 conference on empirical methods in natural language processing. – 2021. – С. 2089-2103.

93. What is knowledge distillation? – Текст : электронный // IBM : [сайт]. – URL: https://www.ibm.com/think/topics/knowledge-distillation (дата обращения: 30.09.2025).
94. What Is Knowledge Distillation? A Deep Dive. – Текст : электронный // Roboflow : [сайт]. – URL: https://blog.roboflow.com/what-is-knowledge-distillation/ (дата обращения: 30.09.2025).
95. From Teacher to Student: Model Distillation for Cost-Effective LLM Deployment. – Текст : электронный // Marvik : [сайт]. – URL: https://blog.marvik.ai/2025/01/28/from-teacher-to-student-model-distillation-for-cost-effective-llm-deployment/ (дата обращения: 30.09.2025).
96. Xu C. et al. Teacher-student collaborative knowledge distillation for image classification: Teacher-student collaborative knowledge distillation for image classification //Applied Intelligence. – 2023. – Т. 53. – №. 2. – С. 1997-2009.
97. What Is Model Distillation? – Текст : электронный // Builtin : [сайт]. – URL: https://builtin.com/artificial-intelligence/model-distillation (дата обращения: 30.09.2025).
98. Hofmeyr S. A., Forrest S., Somayaji A. Intrusion detection using sequences of system calls //Journal of computer security. – 1998. – Т. 6. – №. 3. – С. 151-180.
99. NIST Special Publication 800-92. – Текст : электронный // NIST : [сайт]. – URL: https://www.academia.edu/33477090/Special_Publication_800_92_Guide_to_Computer_Security_Log_Management_Recommendations_of_the_National_Institute_of_Standards_and_Technology (дата обращения: 02.05.2026).
100. Vielberth M. Security information and event management (SIEM) //Encyclopedia of Cryptography, Security and Privacy. – Cham : Springer Nature Switzerland, 2025. – С. 2304-2306.
101. bert-base-uncased. – Текст : электронный // Hugging Face : [сайт]. – URL: https://huggingface.co/google-bert/bert-base-uncased (дата обращения: 30.09.2025).

102. TinyBERT_General_4L_312D. – Текст : электронный // Hugging Face : [сайт]. – URL: https://huggingface.co/huawei-noah/TinyBERT_General_4L_312D (дата обращения: 30.09.2025).
103. Mobilebert-Uncased. – Текст : электронный // Hugging Face : [сайт]. – URL: https://huggingface.co/google/mobilebert-uncased (дата обращения: 30.09.2025).
104. Distilbert-Base-Uncased. – Текст : электронный // Hugging Face : [сайт]. – URL: https://huggingface.co/distilbert/distilbert-base-uncased (дата обращения: 30.09.2025).
105. TinyBERT_General_6L_768D. – Текст : электронный // Hugging Face : [сайт]. – URL: https://huggingface.co/huawei-noah/TinyBERT_General_6L_768D (дата обращения: 30.09.2025).
106. Password Generator Plus. – Текст : электронный // Passwordsgenerator : [сайт]. – URL: https://passwordsgenerator.net/plus/ (дата обращения: 30.09.2025).
107. Bulk Password Generator. – Текст : электронный // Bulk Password Generator : [сайт]. – URL: https://passwords-generator.org/bulk (дата обращения: 30.09.2025).
108. Check Your Password. – Текст : электронный // Password Kaspersky : [сайт]. – URL: https://password.kaspersky.com/ (дата обращения: 30.09.2025).