МИНИСТЕРСТВО ВНУТРЕННИХ ДЕЛ
РЕСПУБЛИКИ КАЗАХСТАН

АЛМАТИНСКАЯ АКАДЕМИЯ
имени МАКАНА ЕСБУЛАТОВА

ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ МЕТОДОВ ДЕАНОНИМИЗАЦИИ ПОЛЬЗОВАТЕЛЕЙ В СОЦИАЛЬНЫХ СРЕДАХ

Учебное пособие

Алматы 2025

УДК 004
ББК 32.973
К13

Обсуждено и одобрено на заседании научно-методического совета Алматинской академии МВД Республики Казахстан им. М. Есбулатова (протокол № 8 от «16» октября 2025г.).

Рецензенты:

Чукумов Г.Б. – начальник факультета профессиональной подготовки Алматинской академии МВД Республики Казахстан им. М. Есбулатова, PhD, полковник полиции

Байшоланова К.С. – профессор Казахского национального университета им. Аль-Фараби, д.э.н.

К13 Кадырова Р.Т., Ендыбайулы Е., Суюнбай Ж. Исследование современных методов деанонимизации пользователей в социальных средах: Учебное пособие. – Алматы: Алматинская академия им. М. Есболатова МВД Республики Казахстан, 2025. – 136 с.

ISBN 978-601-360-209-7

Учебное пособие предназначено для решения вопросов деанонимизации пользователей в социальных сетях. Работа адресована широкому кругу исследователей, учёных и специалистов в области информационной безопасности, а также курсантам, стажёрам, магистрантам и докторантам высших учебных заведений правоохранительных органов. Учебное пособие сочетает в себе теоретические знания и практические навыки в области кибербезопасности, машинного обучения, OSINT и обработки естественного языка и может быть полезно также лицам, проводящим прикладные исследования.

УДК 004
ББК 32.973

ISBN 978-601-360-209-7

Введение

В современном информационном обществе стремительное развитие цифровых технологий наряду с новыми социальными и экономическими возможностями порождает и сложные угрозы. Широкое использование интернета и социальных сетей стало причиной появления новых форм киберпреступности, в том числе онлайн-мошенничества. Особенно функциональные возможности платформы Telegram и предоставляемая пользователю практически неограниченная анонимность превращают её в удобную среду для совершения преступных действий.

Telegram – это мессенджер, получивший широкое распространение во всём мире, который объединяет такие возможности, как шифрование, автоматизация с помощью ботов, канальные рассылки и групповая коммуникация. Однако именно эти преимущества, с другой стороны, делают его эффективным инструментом и для мошенников, и для киберпреступников. Возможность в Telegram идентифицировать пользователя не по номеру телефона или имени, а исключительно по никнейму (username), а также зашифрованная архитектура платформы существенно усложняют процесс установления личности конкретного человека.

Схемы мошенничества в Telegram проявляются в самых разных формах. Одни обманным путём предлагают пользователям инвестиционный доход, другие используют рассылку фишинговых ссылок, создание поддельных ботов, имитацию служб технической поддержки и другие сложные методы. Участившиеся случаи таких действий и рост числа пользователей поднимают проблему информационной безопасности в Telegram на особо важный уровень. Кроме того, распространение мошенничества зачастую происходит в закрытых каналах и приватных группах, что значительно осложняет для правоохранительных органов процесс мониторинга и сбора доказательств.

В связи с этим автоматическое выявление мошеннических действий в сети Telegram на раннем этапе и деанонимизация подозрительных пользователей становится одной из наиболее актуальных научных и прикладных задач сегодняшнего дня. Для реализации данной задачи необходим комплексный подход на стыке информационной безопасности, анализа данных, обработки естественного языка, OSINT и искусственного интеллекта.

Исследования показывают, что мошеннические сообщения обладают определёнными лингвистическими особенностями. Их структура, словарный запас, содержание и стилистика позволяют автоматически распознавать такие тексты с помощью моделей машинного обучения. В этом плане модель BERT, основанная на трансформерной архитектуре, выделяется как инструмент, показывающий высокую эффективность в глубоком понимании контекста текста и точном определении семантических связей.

Предлагаемое учебное пособие посвящено опыту разработки и реализации комплексной системы, которая автоматически выявляет мошеннические сообщения, распространяемые на платформе Telegram, собирает метаданные об анонимных пользователях, осуществляющих такие действия, и пытается деанонимизировать их на основе открытых источников. Архитектура системы построена на языке программирования Python, данные хранятся в базе MongoDB, а модели основаны на алгоритмах BERT и Random Forest. Кроме того, система развёртывается в Docker-контейнерах и дополняется панелью визуализации (Dashboard) для пользовательского интерфейса.

В системе процессы сбора сообщений через Telegram API, их предварительной обработки, классификации и отправки уведомлений о подозрительных сообщениях организованы модульным образом. Преимущество такой архитектуры заключается в гибкости системы, независимости каждого компонента и простоте технического обслуживания. Каждый модуль выполняет конкретную функцию, взаимо-

действуя с другими и обеспечивая полноценный рабочий процесс. Данные, полученные из Telegram, сохраняются в MongoDB, затем обрабатываются моделями машинного обучения, а результаты отображаются на панели визуализации (Dashboard).

В структуре учебного пособия в теоретической части рассматриваются виды мошенничества в социальных сетях и уязвимые места платформы Telegram, а в практической части подробно описываются конкретные примеры кода на Python, методы сбора сообщений через Telegram API, построение эмбеддингов на основе BERT, а также способы поиска пользователя с помощью OSINT. Это прикладной инструмент, направленный не только на изучение теории, но и на практическую реализацию полноценной системы. Учебное пособие написано на академическом уровне и имеет научно-методическую основу, поэтому может быть напрямую использовано в образовательном процессе, научных исследованиях и профессиональной подготовке.

Цель учебного пособия – объяснить и на практике показать методы автоматического выявления мошеннических сообщений в Telegram, а также сбора дополнительной информации о пользователях, распространяющих их (username, user_id, тип устройства, время регистрации и др.), с последующей деанонимизацией. Кроме того, пособие, объединяя в одной системе технологии Python, NLP, ML и OSINT, позволяет пользователям одновременно развивать практические навыки и системное мышление. Это не только источник теоретических знаний, но и практическая площадка для отработки навыков. Его содержание направлено на развитие профессиональных компетенций в области кибербезопасности, социальной инженерии, информационной разведки, науки о данных и искусственного интеллекта.

1. МОШЕННИЧЕСТВО В СОЦИАЛЬНЫХ СЕТЯХ И ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ЕГО ВЫЯВЛЕНИЯ

1.1. Мошенничество в социальных сетях: распространенность, виды и особенности

За последнее десятилетие социальные сети и мессенджеры стали неотъемлемой частью повседневной жизни людей. Они широко используются для общения, обмена информацией, бизнеса и развлечений. Однако, наряду с таким широким распространением, эти платформы также стали привлекательной средой для киберпреступников [1]. Мошенничество в социальных сетях представляет собой широкий спектр вредоносных действий, направленных на обман пользователей, кражу их персональных данных или причинение финансового ущерба. В данном разделе подробно рассматривается распространённость мошенничества в социальных сетях, его основные виды и характерные особенности.

Число случаев мошенничества в социальных сетях растет во всем мире с каждым годом. Исследования и отчеты в области кибербезопасности подтверждают эту тенденцию. Согласно отчетам Федеральной торговой комиссии (FTC) США, за последние годы расходы на мошенничество, совершаемое через социальные сети, выросли в несколько раз. Согласно данным, опубликованным FTC в 2023 году, расходы на мошенничество в социальных сетях достигли миллиардов долларов США, что делает его одним из самых дорогостоящих видов мошенничества [2]. Особенно распространены инвестиционное мошенничество, мошенничество на романтических связях и мошенничество в сфере онлайн-покупок.

Отчёты Центра по борьбе с киберпреступностью Европола (EC3) также свидетельствуют об увеличении числа случаев мошенничества с использованием социальной инженерии по всему Европейскому союзу [3]. Мессенджеры,

в частности, WhatsApp, Telegram и Facebook Messenger, являются распространёнными инструментами, используемыми мошенниками для установления прямого контакта со своими жертвами и завоевания их доверия.

Эта проблема актуальна и для Казахстана. По данным Центра по борьбе с киберпреступностью МВД Республики Казахстан, растёт число случаев интернет-мошенничества, в том числе совершаемых через социальные сети и мессенджеры. Недостаточная цифровая грамотность граждан и высокое доверие к новым технологиям способствуют достижению мошенниками своих целей. Данная статистика свидетельствует о важности разработки эффективных инструментов и методов, направленных на борьбу с мошенничеством в социальных сетях.

Существует множество видов мошенничества в социальных сетях, которые постоянно меняются и развиваются [4]. Однако наиболее распространёнными являются:

Фишинг – один из самых распространённых видов мошенничества. Мошенники пытаются заманить пользователей на поддельные сайты, отправляя поддельные сообщения или электронные письма от имени известных компаний (банков, почтовых служб, социальных сетей, государственных учреждений). Эти сайты выглядят очень похоже на официальные и требуют от пользователя ввести конфиденциальные данные, такие как логин, пароль, номер банковской карты, CVV-код. Фишинговые ссылки в социальных сетях могут распространяться через личные сообщения, публикации в группах или рекламу. В Telegram это часто делается с помощью ботов или поддельных аккаунтов. Поддельные сообщения ведут на поддельные сайты с целью кражи конфиденциальных данных.

Инвестиционные мошенничества. Мошенники предлагают пользователям «специальные» инвестиционные проекты, обещающие быструю и гарантированно высокую прибыль. Часто это может быть связано с криптовалютами, акциями, недвижимостью или другими активами. Они пытаются убедить своих жертв, используя поддельную

статистику доходности, поддельные положительные отзывы и профессионально выглядящие веб-сайты. Получив деньги, мошенники либо прекращают отношения, либо требуют ещё. Финансовые пирамиды (схемы Понци) также попадают в эту категорию. Финансовые пирамиды – это сомнительные инвестиционные проекты, обещающие высокую и гарантированную прибыль.

Социальная инженерия – это набор методов, представляющих угрозу информационной безопасности, основанных на воздействии на психологию и доверие человека, а не на технических средствах. Эти методы используются злоумышленниками, чтобы обманом заставить пользователей раскрыть конфиденциальную информацию, получить несанкционированный доступ к системе или выполнить желаемые действия [5]. Методы социальной инженерии часто включают в себя методы обмана, такие как фишинг, вишинг, завоевание доверия через незнакомые ссылки или выдача себя за авторитетное лицо. Мошенники завоевывают доверие своих жертв и побуждают их добровольно предоставлять конфиденциальную информацию, переводить деньги или выполнять вредоносные действия. Виды социальной инженерии включают в себя такие методы, как претекстинг (выдумка ложного человека или сценария), запугивание или устрашение жертвы и вызывание сочувствия. Многие виды фишинга также содержат элементы социальной инженерии. Склонение пользователя к выполнению желаемых действий посредством психологической манипуляции.

Мошенничество при трудоустройстве – обман соискателей с помощью ложных объявлений о вакансиях. Для трудоустройства могут потребоваться аванс, копии документов или платные курсы.

Поддельные лотереи и розыгрыши призов – пользователи получают уведомления о выигрыше крупных сумм денег, дорогостоящего оборудования или других ценных призов. Чтобы получить «приз», требуется внести

определённую сумму аванса (например, налоги, комиссия, стоимость доставки) или предоставить персональные данные. Как правило, эти выигрыши фиктивны.

Распространение вредоносного ПО – вредоносные файлы или ссылки могут распространяться через социальные сети. Они маскируются под интересный контент (эксклюзивные видео, бесплатное ПО) и побуждают пользователя загрузить или открыть их. В результате на устройство могут быть установлены вирусы, трояны, шпионские программы или программы-вымогатели.

Поддельная техническая поддержка – мошенники выдают себя за сотрудников технической поддержки известных компаний (Microsoft, Apple, антивирусные компании) и сообщают пользователям о проблемах с их устройствами.

Цифровые технологии. Стремительное развитие цифровых технологий в современном информационном обществе, наряду с новыми социальными и экономическими возможностями, также создаёт серьёзные угрозы. Широкое распространение Интернета и социальных сетей привело к появлению новых видов киберпреступности, включая онлайн-мошенничество. В частности, функциональность платформы Telegram и возможность предоставления неограниченной анонимности пользователю делают её благоприятной средой для совершения преступных деяний.

Telegram – популярный во всем мире мессенджер, сочетающий в себе такие функции, как шифрование, автоматизация ботов, чаты и групповые коммуникации для обмена сообщениями и распространения информации. Однако эти преимущества также делают его эффективным инструментом для мошенников и киберпреступников. Возможность идентифицировать пользователя Telegram не по номеру телефона или имени, а только по никнейму (имени пользователя), а также зашифрованная архитектура платформы затрудняют идентификацию реального человека.

Мошеннические схемы на платформе Telegram бывают самых разных видов. Некоторые обманывают пользователей, обещая возврат инвестиций, другие используют более изощрённые методы, такие как распространение фишинговых ссылок, создание поддельных ботов и имитация служб технической поддержки. Частота подобных действий и рост числа пользователей ставят вопрос информационной безопасности Telegram на особенно серьёзный уровень. Кроме того, тот факт, что мошенничество часто распространяется в закрытых каналах и закрытых группах, затрудняет отслеживание и доказывание мошенничества правоохранительными органами.

Фишинг – это вид мошенничества, при котором мошенники выдают себя за доверенные организации (банки, государственные учреждения, почтовые службы и т.д.) и используют поддельные сообщения, сайты или ботов, чтобы обманным путём заставить пользователей предоставить конфиденциальную информацию (логин, пароль, номер карты, CVV-код и т.д.). Фишинг в Telegram часто распространяется через поддельные аккаунты технической поддержки, поддельные учётные данные и поддельных ботов. Ссылки часто выглядят как ссылки на реальные ресурсы, что вводит пользователя в заблуждение и вынуждает его вводить свои данные.

Инвестиционное мошенничество – это вид мошенничества, обманывающий пользователей, предлагая им финансовые схемы, обещающие высокую прибыль. Этот метод, часто встречающийся в Telegram-каналах, предлагает зарабатывать деньги, инвестируя в криптовалюту, участвуя в международной торговле или реализуя уникальные бизнес-проекты. Мошенники используют поддельную статистику, фейковые отзывы и имена известных людей, чтобы внушить доверие. После того, как жертва отправляет деньги, мошенники либо не выходят на связь, либо требуют дополнительную сумму.

Социальная инженерия – это набор методов обмана, основанных на психологии человека, а не на технических средствах. Мошенники входят в доверие к жертве и убеждают её совершить определённое действие: раскрыть пароль, перевести деньги, перейти по ссылке или установить приложение. В Telegram это часто делается путём общения с незнакомыми пользователями через личные сообщения, принятия срочных решений или отправки ложных предупреждений об угрозах.

Трудоустройство – это форма обмана соискателей, основанная на обещании им вакансий. В Telegram-каналах жертв заманивают рекламой работы за рубежом, удалённой работы или заработка на дому, предлагая заполнить анкету, предоставить документы или оплатить подготовительный курс «обучения». Мошенники используют подобные действия для сбора персональных данных или заработка.

Поддельные призы – пользователю сообщают о выигрыше приза и просят заранее оплатить «налог», «стоимость доставки» или другую плату за его получение. Такие действия создают эмоциональное давление и подталкивают человека к быстрому принятию решения. В Telegram это часто распространяется через ботов или аккаунты, созданные от имени известных брендов. Мошенники могут даже использовать настоящий документ о выигрыше, список победителей или логотипы.

Распространение вредоносного ПО – это попытка установить шпионское ПО, трояны, программы-вымогатели или вирусы на устройство пользователя. В Telegram это маскируется под интересный контент: бесплатное ПО, секретные файлы, эксклюзивный контент. Загружая файл или переходя по ссылке, пользователь рискует стать жертвой слежки за своим устройством или кражи данных.

Поддельная техническая поддержка – сообщение пользователю о проблеме с его устройством или аккаунтом и предложение установить дополнительное программное обеспечение или оплатить её устранение. В Telegram это часто делается через поддельные аккаунты, которые используют логотипы известных компаний для внушения

доверия. Если жертва предоставляет удалённый доступ к своему устройству, мошенник может легко завладеть её личными данными или финансами.

Поскольку каждый из этих видов мошенничества активно используется на платформе Telegram, их раннее выявление и разработка инструментов борьбы с ними является актуальной задачей сегодня. Представленный обучающий материал обеспечивает системный подход к достижению этой цели, помогая разобраться в схемах мошенничества и научиться защищаться от них. Ниже представлена сводная таблица наиболее распространённых видов мошенничества в социальных сетях:

№	Вид мошенничества	Краткое описание	Основная цель
1	Фишинг	Поддельные сообщения, ведущие на поддельные веб-сайты с целью кражи конфиденциаль-ных данных.	Кража логина/пароля, данных банковской карты.
2	Инвестиционное мошенничество	Сомнительные инвестиционные проекты, финансовые пирамиды, обещающие высокую и гарантированную прибыль.	Нецелевое использование средств.
3	Социальная инженерия	Побуждение пользователя к совершению желаемых действий посредством психологическо-го манипулиро-вания.	Получение конфиденциальной информации, перевод денег, злонамеренные действия.
4	Работа	Ложные объявления	Предоплата,

		о вакансиях, требующие предоплаты или предоставления персональных данных для трудоустройства.	сбор персональных данных.
5	Поддельные выигрыши	Объявление о крупных выигрышах или призах и требование внесения аванса или предоставления персональных данных для их получения.	Предоплата, сбор персональных данных.
6	Распространение вредоносного ПО	Распространение вредоносных файлов или ссылок, замаскирован-ных под интересный контент.	Повреждение устройства, кража данных, шантаж.
7	Поддельная техническая поддержка	«Решение» технических проблем от имени известных компаний.	Снимайте деньги, удалённо управляйте устройством.

Таблица 1.1 Основные виды мошенничества в социальных сетях и их краткая характеристика

Также при анализе схем мошенничества на платформе Telegram наблюдается их корреляция с сезонными, географическими и контекстными факторами. Например, схемы, основанные на фиктивных выигрышах и скидках, становятся более распространенными в предпраздничный период. В периоды экономического кризиса или высокой безработицы увеличивается количество мошеннических сообщений, связанных с трудоустройством. Кроме того, в

кризисных ситуациях, таких как пандемия, война или стихийные бедствия, мошенники используют страх и недоверие пользователей для организации фальшивой гуманитарной помощи и фиктивных сборов пожертвований. В Telegram такие схемы часто организованы профессионально и сопровождаются привлекательной графикой, достоверными комментариями и активностью ботов. Использование ботов и автоматизированных систем для распространения мошеннического контента значительно расширяет его сферу применения. Кроме того, некоторые мошенники покупают каналы или группы, предоставляющие незаконные услуги, тем самым получая мгновенный доступ к новой аудитории. Они заранее определяют свою целевую аудиторию и формируют доверие, создавая контент, адаптированный для этой группы. Особенно часто жертвами становятся молодежь, люди с низкой финансовой грамотностью и пользователи, не уделяющие должного внимания технической безопасности. Мошенники используют изображения известных людей, логотипы или фейковые новости для завоевания доверия, создавая впечатление, что их сообщения достоверны и заслуживают доверия. Эффективные меры противодействия столь сложным и масштабным фишинговым экосистемам не должны ограничиваться только технологическими решениями. Формирование культуры кибергигиены, повышение медиаграмотности пользователей и их адаптация к информационной безопасности являются важными составляющими комплексной стратегии защиты. Поэтому более глубокое выявление и изучение мошеннических схем в сети Telegram представляет собой не только научный интерес, но и задачу, имеющую реальное социальное значение. В этом контексте использование методов обработки естественного языка и машинного обучения является эффективным способом автоматического распознавания мошеннических сообщений. При этом проверка результатов, полученных из открытых источников, и выявление дополнительной информации о пользователе является важным аспектом деанонимизации и сбора

доказательств. Разработанная для этой цели система может стать эффективным инструментом противодействия мошенническим действиям в Telegram, реализуя комплексный подход в соответствии с современными требованиями.

Платформа Telegram в настоящее время является одним из самых популярных мессенджеров в мире. Неограниченный доступ к открытым группам и каналам, анонимность и широкий функционал делают эту платформу привлекательной не только для добросовестных пользователей, но и для преступных групп, реализующих различные мошеннические схемы. Глубокий анализ мошеннических схем в Telegram показывает, что они тесно связаны с сезонными, географическими и контекстными факторами.

Например, в предпраздничный период мошенники в Telegram активно продвигают фейковые розыгрыши и скидки. Они легко обманывают пользователей, распространяя фейковые сообщения от имени известных магазинов, сервисных центров и банков. Поскольку в эти периоды люди больше заинтересованы в покупках и скидках, мошенники эффективно используют эту психологическую слабость. Подобные действия особенно распространены в период Нового года, Навруза, «Чёрной пятницы» и других распродаж.

В периоды экономического кризиса, роста инфляции или высокой безработицы мошенники часто распространяют ложные предложения о трудоустройстве или дополнительном заработке. Они обещают «устроиться на работу» или «получить дополнительный доход» за определённую сумму, но как только пользователь переводит деньги, соединение обрывается. Такие схемы особенно распространены в экономически неблагополучных странах и регионах.

Во время чрезвычайных ситуаций, таких как пандемии, войны, стихийные бедствия, психологическое состояние людей и их информационные потребности меняются. В такие моменты в Telegram широко распространяются фейковые

гуманитарные акции, фейковые благотворительные фонды, фейковые кампании по продаже медицинского оборудования и фейковые сборы пожертвований. Страх и недоверие пользователей становятся главным оружием мошенников.

Мошеннические схемы в Telegram стали высокоорганизованными. Они не ограничиваются простыми текстовыми сообщениями, а организуют искусственную активность с использованием профессионально созданной графики, логотипов, реальных отзывов и даже ботов. Мошенники используют изображения известных компаний, финансовых учреждений и даже знаменитостей, чтобы придать своим сообщениям реалистичность. В частности, мошенники открывают поддельные страницы, которые якобы распространяют сообщения от имени известных блогеров и лидеров общественного мнения.

В Telegram боты и автоматизированные системы позволяют мошенническому контенту быстро распространяться среди широкой аудитории. Боты автоматически публикуют сообщения в каналах и группах, генерируя фейковые комментарии и «лайки», создавая видимость популярности. Этот метод ещё больше укрепляет доверие пользователей, и во многих случаях люди реагируют на фейковые сообщения, даже не проверяя их.

Некоторые мошенники покупают существующие каналы или группы в Telegram, предлагающие незаконные услуги, и нацеливают свои мошеннические схемы на уже готовую аудиторию. В таких случаях создание нового канала не занимает много времени, и их действия сразу же начинают распространяться. Мошенники заранее изучают свою целевую аудиторию и разрабатывают контент, учитывающий возраст, интересы и социальный статус аудитории.

Основными жертвами мошенников становятся молодёжь, люди с низкой финансовой грамотностью и пользователи, не соблюдающие правила информационной безопасности. Они не замечают вовремя признаки мошеннических схем и легко раскрывают свои личные данные или средства.

Борьба с мошенничеством в Telegram не должна ограничиваться только техническими методами. Это сложная проблема, требующая комплексного подхода.

Во-первых, крайне важно сформировать культуру кибергигиены. Пользователи должны постоянно получать информацию о том, как защитить свои персональные данные, какие сообщения могут быть недостоверными и по каким ссылкам не следует переходить. Для этого необходимо организовать специальные обучающие курсы, разъяснительную работу в социальных сетях и проводить кампании в СМИ.

Во-вторых, необходимо повышать медиаграмотность – развивать умение отличать фейковые новости от ложных рекомендаций. Пользователи должны научиться критически воспринимать каждую информацию.

В-третьих, Telegram необходимо внедрить современные технологические решения, позволяющие автоматически распознавать мошеннические схемы. Эффективными инструментами в этом направлении могут стать методы обработки естественного языка (NLP) и машинного обучения (ML). Например, разрабатывая ботов и интеллектуальные системы, автоматически определяющие мошеннические сообщения по определённым критериям, можно будет пресекать действия мошенников на ранней стадии.

Также важно проверять данные, полученные из открытых источников, выявлять дополнительную информацию о пользователе и отслеживать следы мошеннической деятельности. Это играет важную роль в процессе деанонимизации и на этапе сбора доказательств.

Изучение мошеннических схем в Telegram — не только теоретическая, но и весьма актуальная с социальной точки зрения задача. Это явление представляет прямую угрозу финансовой и личной безопасности людей. Прогнозируя и оперативно выявляя действия мошенников, мы можем значительно повысить уровень кибербезопасности общества.

В заключение следует отметить, что для выявления и пресечения мошеннических схем на платформе Telegram не-

достаточно полагаться исключительно на техническую поддержку правоохранительных органов или самой платформы. Этот вопрос требует совместных усилий всех пользователей, исследователей, IT-специалистов и представителей медиаиндустрии. Только системное обучение, формирование культуры информационной безопасности и использование интеллектуальных технологий позволят эффективно бороться с мошенничеством.

Предотвращение мошенничества в Telegram – один из важнейших шагов в современном цифровом обществе.

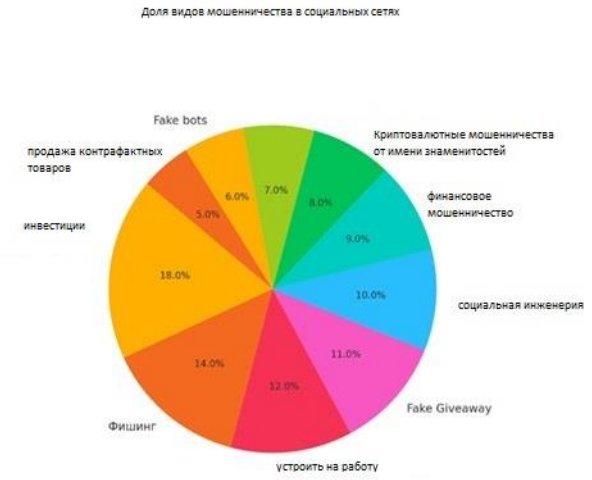

Рисунок 1.1 Распределение распространенных видов мошенничества в социальных сетях
(на основе предварительного анализа или обзора литературы)

На рисунке 1.1 представлена круговая диаграмма, показывающая относительную долю видов мошенничества в социальных сетях. Эта диаграмма основана на фактических данных, собранных в ходе исследования, и показывает частоту использования мошеннических схем.

Как видно из диаграммы, наибольшую долю занимает инвестиционное мошенничество (18,0%). Этот вид

мошенничества эффективно эксплуатирует естественное желание пользователей быстро разбогатеть. В инвестиционном мошенничестве часто участвуют организации, выдающие себя за подставные финансовые компании, криптовалютные платформы или высокодоходные инвестиционные проекты. Они заманивают пользователей, обещая огромную прибыль за короткий срок. Достоверность таких схем подкрепляется профессионально оформленными веб-сайтами, поддельными контрактами и фальшивыми инвестиционными отчётами.

На втором месте находится фишинг (14,0%). Фишинг – это вид мошенничества, направленный на незаконное получение персональных данных пользователей, в частности логинов, паролей, данных банковских карт и персональных идентификационных номеров. Фишинговые сообщения, как правило, составлены таким образом, что очень похожи на официальные сообщения банков, популярных интернет-магазинов или социальных сетей. Когда пользователь переходит по ссылке и вводит свои персональные данные, эта информация попадает в руки мошенников.

Мошенничество с трудоустройством (12,0%) также распространено в социальных сетях. Мошенники публикуют объявления, предлагающие высокую зарплату и лёгкую работу. Такие схемы часто требуют «первоначального взноса», платы за оформление документов или «обучения». Этот вид мошенничества особенно распространён в периоды высокой безработицы или финансовых трудностей.

Поддельные розыгрыши (11,0%) – ещё один распространённый вид мошенничества в социальных сетях. Эти схемы предлагают бесплатные подарки или крупные денежные призы от имени конкретной компании или знаменитости. Пользователю предлагается перейти по ссылке и ввести личные данные или внести небольшой авансовый платёж, чтобы получить приз. В некоторых случаях для участия в розыгрыше его просят поделиться

сообщением с друзьями, что способствует вирусному распространению розыгрыша.

Также широко используются методы общей социальной инженерии (10,0%). В этом виде мошенничества мошенники пытаются достичь своих целей, воздействуя на эмоции, убеждения или социальный статус людей. Иногда они просят о помощи от имени человека, оказавшегося в чрезвычайной ситуации, а иногда действуют под именем известного человека или компании.

Кроме того, довольно распространены случаи финансового мошенничества (9,0%), криптомошенничества (8,0%), выдачи себя за знаменитостей (7,0%), использования поддельных ботов (6,0%) и продажи поддельных товаров (5,0%). Финансовое и криптовалютное мошенничество предлагает пользователям «особые» инвестиционные возможности, в то время как поддельные боты обманывают пользователей с помощью автоматических поддельных сообщений. Поддельные продажи товаров – распространённая схема, с помощью которой мошенники пытаются продать некачественные или несуществующие товары.

Мошенничество в социальных сетях стало серьёзной и растущей угрозой для общества. Основная причина этого – высокое доверие пользователей к социальным сетям и их готовность делиться личной информацией. Схемы мошенничества в социальных сетях постоянно развиваются и становятся всё более изощрёнными, что затрудняет борьбу с ними.

Глубокое изучение различных форм мошенничества и механизмов их распространения – первый и важнейший шаг к совершенствованию систем безопасности и разработке интеллектуальных средств обнаружения мошенничества. Для эффективной борьбы с мошенничеством недостаточно лишь совершенствования технических систем. В связи с этим крайне важно повышать медиаграмотность пользователей, разъяснять им правила кибергигиены и формировать культуру защиты персональных данных.

Кроме того, социальные сети сами должны взять на себя ответственность за борьбу с мошенническими схемами. Совершенствование алгоритмов автоматического выявления фейковых аккаунтов, подозрительной рекламы и мошеннических ссылок должно стать одним из ключевых требований для стабильной работы этих платформ.

В заключение следует отметить, что выявление, прогнозирование и предотвращение мошенничества в социальных сетях – общая задача для всего общества. Только совместные действия государственных органов, правоохранительных органов, IT-компаний и обычных пользователей позволят существенно сократить масштабы мошенничества в социальных сетях. Комплексная стратегия, сочетающая технологические, юридические и образовательные направления, – основной способ эффективного противодействия этой современной угрозе.

1.2. Анализ возможностей платформы Telegram и ее потенциала для мошенничества

Среди социальных сетей и мессенджеров платформа Telegram выделяется своими уникальными возможностями. Эти особенности делают её привлекательным инструментом общения для миллионов пользователей, но в то же время создают благоприятную среду для мошенников и киберпреступников. В этом разделе анализируются основные характеристики платформы Telegram, её аудитория, возможности обеспечения анонимности, принципы работы групп и каналов, а также рассматриваются причины, по которым мошенники часто выбирают именно эту платформу.

Telegram – облачный сервис мгновенного обмена сообщениями, основанный Павлом Дуровым в 2013 году, фокусирующийся на скорости, безопасности и кроссплатформенной функциональности. Он имеет большую и растущую глобальную аудиторию. По состоянию на начало 2023 года у Telegram было более 700 миллионов активных

пользователей в месяц [6]. Хотя его пользовательская база разнообразна, он особенно популярен в Восточной Европе, на Ближнем Востоке, в Юго-Восточной Азии и Латинской Америке. Аудитория также разнообразна: от молодежи до профессионалов и деловых людей. Telegram известен как платформа, заботящаяся о конфиденциальности. Пользователи могут скрывать свои номера телефонов и быть идентифицированными только по уникальному имени пользователя, что обеспечивает определенную степень анонимности. Они также могут контролировать, кто может видеть их номера телефонов (никто, только контакты или все). Сообщения, отправляемые через секретные чаты, защищены сквозным шифрованием, то есть они расшифровываются только на устройствах отправителя и получателя и не могут быть доступны на серверах Telegram [7]. Однако обычные облачные чаты используют шифрование типа «клиент-сервер/сервер-клиент», то есть данные хранятся на сервере. В секретных чатах, а в некоторых случаях и в обычных чатах, сообщения также можно настроить на автоматическое удаление по истечении определённого периода времени.

Группы в Telegram – это чаты для обмена сообщениями с большим количеством участников (до 200000 человек). Они могут быть публичными (к ним может присоединиться любой желающий) или закрытыми (по приглашению). Администраторы обладают широкими возможностями управления группой, удаления сообщений и блокировки участников. Каналы же служат инструментом одностороннего общения. Владельцы каналов (администраторы) могут отправлять сообщения, новости и медиафайлы своим подписчикам (неограниченному количеству), в то время как подписчики могут только читать сообщения в канале, но не писать их (если не включена функция комментирования). Каналы, как и группы, могут быть публичными или закрытыми. Боты – это сторонние приложения, созданные с использованием API Telegram,

которые автоматизируют различные функции, такие как предоставление информации, развлечения, оказание услуг, прием платежей и т.д. Они могут вступать в группы и интегрироваться с каналами.

К сожалению, вышеупомянутые особенности платформы Telegram также делают её привлекательной для мошенников. Главной причиной этого является её относительная анонимность: возможность скрыть свой номер телефона и работать только под своим именем пользователя помогает мошенникам скрывать свою настоящую личность и уклоняться от правоохранительных органов, а лёгкий процесс регистрации новых учётных записей также способствует этому. Кроме того, платформа с её миллионами пользователей предоставляет мошенникам доступ к большой аудитории, что позволяет им широко распространять вредоносные сообщения, фишинговые ссылки или фейковые предложения. Благодаря обширным группам и каналам связи они могут охватить множество потенциальных жертв за короткое время.

Особенности групп и каналов также благоприятствуют мошенникам. Наличие у каждой группы и канала собственных администраторов, а также более слабая централизованная модерация платформы, чем в других социальных сетях (особенно в закрытых группах), позволяют мошенникам бесконтрольно распространять вредоносный контент. Через каналы они могут мгновенно доставлять свою рекламу или фейковые предложения тысячам, а то и миллионам подписчиков. Кроме того, мошенники активно используют ботов для автоматизации своих вредоносных действий. Боты могут массово рассылать фишинговые сообщения, вести поддельные диалоги с пользователями, имитировать работу фейковых служб поддержки или реализовывать сложные схемы, такие как мошенничество с криптовалютой.

Возможности Telegram, ориентированные на конфиденциальность, в плане изображения и шифрования,

также могут быть использованы не по назначению. Это может создать ложное чувство безопасности у некоторых пользователей, позволяя мошенникам убедить их в том, что их действия останутся анонимными. Стоит также отметить, что сквозное шифрование (в приватных чатах) затрудняет расследование мошеннических действий правоохранительными органами. Глобальный, трансграничный характер платформы позволяет мошенникам проводить операции, нацеленные на жертв в разных странах, и уклоняться от юридической ответственности. Кроме того, мошенники эксплуатируют интерес части аудитории Telegram к новым технологиям, включая криптовалюты и онлайн-инвестиции, предлагая мошеннические схемы, связанные с этими отраслями [8].

Совокупность этих факторов делает платформу Telegram одной из самых благоприятных площадок для реализации злонамеренных целей мошенников. Поэтому крайне важно разработать и внедрить эффективные методы и технологии, направленные на борьбу с мошенничеством на этой платформе. Это позволит обеспечить безопасность пользователей и повысить надёжность платформы.

1.3. Обзор методов обработки естественного языка (NLP) и машинного обучения (ML) для обнаружения мошеннических сообщений

Задача автоматического обнаружения мошеннических сообщений часто основана на анализе текстовых данных. Этот процесс требует использования различных методов и технологий в области обработки естественного языка (NLP) и машинного обучения (ML) [9]. В то время как методы NLP помогают преобразовать текст в форму, понятную компьютеру, модели машинного обучения обучаются обнаруживать признаки мошенничества и классифицировать сообщения по соответствующим категориям на основе обработанных данных. В этом разделе рассматриваются

основные концепции NLP в этой области, традиционные модели машинного обучения и архитектуры глубокого обучения, а также формулировки многих известных задач классификации.

Основная цель обработки естественного языка – сделать человеческий язык пригодным для компьютерного анализа и понимания. В контексте анализа мошеннических сообщений часто используются базовые этапы и методы обработки естественного языка, такие как токенизация, лемматизация/стемминг и векторное представление. Токенизация – это процесс разделения текста на отдельные значимые единицы, то есть токены (обычно слова, фразы или знаки препинания). Это первый и самый важный шаг в подготовке текста к дальнейшему анализу. Предложение «Удвойте ваши деньги!» можно разделить на токены («Удвойте ваши деньги», «удвойте», «удвойте», «!»). Методы лемматизации и стемминга направлены на приведение слов к их базовой или исходной форме, что помогает сократить размер пространства токенов и повысить обобщающую способность модели. Стемминг пытается найти корень слова, удаляя суффиксы в конце слова («students», «textbook» – «study»), и, хотя это обычно происходит быстро, результат может быть грамматически неверным. Лемматизация, с другой стороны, приводит слово к его исходной форме (лемме) в словаре («was», «is going» – «going»), что является более сложным процессом, но даёт более точные результаты. Векторное представление, или встраивание слов, позволяет моделям машинного обучения работать непосредственно с текстовыми данными, преобразуя слова или предложения в числовые векторы. Основные методы, используемые в этой области, – это TF-IDF (Term Frequency-Inverse Document Frequency), Word2Vec (Skip-gram, CBOW) и GloVe (Global Vectors for Word Representation). TF-IDF – это статистический метод, который присваивает каждому слову вес на основе частоты слова в документе и его обратной частоты в документе по всему

корпусу. Word2Vec – это модель на основе нейронной сети, которая представляет слова в плотном векторном пространстве с учётом их контекстного значения, где слова со схожим значением расположены близко друг к другу в векторном пространстве. GloVe – ещё один популярный метод, генерирующий векторные предложения на основе статистики совместной встречаемости слов.

Для классификации векторизованных текстовых данных используются различные традиционные модели машинного обучения. К ним относятся наивный байесовский алгоритм, логистическая регрессия, опорные векторные машины (SVM) и случайный лес. Наивный байесовский алгоритм – это вероятностный классификатор, основанный на теореме Байеса, которая предполагает простое предположение о независимости признаков друг от друга. Он часто применяется к небольшим наборам данных, особенно при фильтрации спама, благодаря высокой скорости вычислений. Логистическая регрессия – это линейная модель, которая предсказывает вероятность принадлежности к определённому классу путём пропускания линейной комбинации входных признаков через логистическую функцию и обладает относительно хорошей производительностью благодаря простоте интерпретации. SVM – это мощный классификатор, который пытается найти гиперплоскость, разделяющую точки данных, принадлежащие разным классам, на максимальном расстоянии друг от друга, и может эффективно работать в многомерных пространствах. Случайный лес – это модель, основанная на ансамбле нескольких деревьев решений, каждое из которых обучается на случайной выборке данных и случайном подмножестве признаков, что приводит к созданию стабильной модели, устойчивой к переобучению и обладающей высокой точностью. Хотя эти модели широко используются для обнаружения мошеннических сообщений и показали определенную эффективность, они могут не полностью понимать глубокий семантический смысл текста или контекстные связи между словами.

В последние годы методы глубокого обучения совершили значительный прорыв в области обработки естественного языка (NLP). В частности, классификация текстов, то есть автоматическое определение категории текста, стала одной из наиболее эффективных задач, решаемых с помощью моделей глубокого обучения. Традиционные методы (например, наивный байесовский анализ, логистическая регрессия или модели машинного обучения, основанные на рисованных признаках) не могли изучить контекстные связи в тексте на достаточно глубоком уровне. Модели глубокого обучения, с другой стороны, позволяют эффективно изучать сложные языковые структуры и скрытые зависимости [10]. К таким моделям относятся рекуррентные нейронные сети (RNN), долговременная кратковременная память (LSTM) / рекуррентные блоки с гейтированием (GRU), сверточные нейронные сети (CNN) и преобразователи, особенно BERT (Bidirection Encoder Representations from Transformers). В то время как RNN – это сети, разработанные для учёта последовательной природы слов, LSTM/GRU – это их особые типы, предназначенные для лучшего запоминания долговременных зависимостей и решения проблемы «исчезающего градиента». Хотя CNN изначально была разработана для анализа изображений, она также успешно применялась для извлечения локальных признаков из текстовых данных. Архитектура трансформатора и её основа, BERT, считаются одним из революционных инноваций в области обработки естественного языка (NLP). BERT – это предобученная языковая модель, разработанная Google на основе кодирующей части трансформатора. Её основные преимущества включают двунаправленное понимание контекста, предобучение на очень больших текстовых корпусах (с использованием задач Masked Language Model и Next Sentence Prediction), а также способность легко адаптироваться к конкретной задаче путём дополнительного обучения на небольшом объёме фиксированных данных (тонкая настройка) [11]. BERT и его варианты (RoBERTa, ALBERT, DistilBERT и т.д.) в настоящее

время показывают наилучшие результаты при решении многих задач обработки естественного языка (NLP).

Основные модели, используемые для классификации текста в глубоком обучении, включают рекуррентные нейронные сети (RNN), усовершенствованные рекуррентные архитектуры, такие как долговременная кратковременная память (LSTM) и управляемые рекуррентные блоки (GRU), а также сверточные нейронные сети (CNN) и трансформаторы, которые получили широкое распространение в последние годы.

Архитектура рекуррентных нейронных сетей (РНС) была одним из первых широко используемых методов обработки естественного языка. Эта модель позволяет учитывать последовательный характер текста, то есть при обработке каждого слова она сохраняет информацию из предыдущих слов в памяти и вычисляет следующий шаг. Однако традиционные РНС часто страдают от проблемы «исчезающего градиента» при передаче информации в длинных текстах, то есть важная контекстная информация постепенно исчезает.

Для решения этой проблемы были разработаны модели LSTM и GRU. Эти сети отличаются способностью хранить долгосрочные зависимости. LSTM может хранить важную информацию в течение длительного времени, используя ячейки памяти, и отфильтровывать ненужную информацию. GRU – это упрощённая версия LSTM, которая благодаря меньшему количеству параметров обладает более высокой скоростью обучения, но при этом схожа по эффективности. Эти модели лучше справляются с выявлением важных связей в тексте и значительно повышают качество классификации текста.

Хотя сверточные нейронные сети изначально были разработаны для компьютерного зрения, они также успешно применяются в обработке текстовых данных. СНС эффективно извлекают локальные признаки (n-граммы, фразы) из коротких текстов или предложений и преобразуют их в полезные векторные представления для последующей класси-

фикации. Главным преимуществом моделей СНС является их способность к параллельной обработке и высокая скорость обучения.

Модель Transformer, предложенная исследователями Google в 2017 году, стала инициатором революционных изменений в области обработки естественного языка (NLP). Эта модель принципиально отличается от предыдущих методов, основанных на цепной обработке, таких как RNN и LSTM. Архитектура Transformer позволяет одновременно выполнять параллельный процесс обработки, учитывая зависимости между всеми словами. Это позволило в несколько раз увеличить скорость работы с большими объёмами текста.

Одной из самых успешных моделей на основе трансформеров является BERT (Bidirection Encoder Representations from Transformers). Эта модель была представлена Google в 2018 году и сразу же приобрела огромный успех в сообществе специалистов по обработке естественного языка (NLP). Главной особенностью BERT является способность одновременно читать текст в обоих направлениях (слева направо и справа налево) и глубже понимать контекст. Предыдущие модели, такие как RNN или LSTM, часто читали текст только в одном направлении (слева направо), что не позволяло им учитывать весь смысл текста.

Важнейшим преимуществом BERT является тонкая настройка (Fine-tuning), то есть предобученная модель может быть легко адаптирована к конкретной задаче, используя небольшой объём реальных данных. Этот подход проложил путь к достижению высоких результатов во многих задачах обработки естественного языка, таких как классификация текстов, построение вопросов и ответов, аннотирование текстов и распознавание именованных сущностей (NER).

В традиционной многоклассовой классификации каждый образец принадлежит только одному классу, тогда как в случае обнаружения мошеннических сообщений одно сообщение может содержать признаки нескольких видов мошенничества одновременно. Одно сообщение может

содержать как фишинговую ссылку, так и призыв к пользователю к немедленному действию. Это приводит к задаче многоклассовой классификации. Целью многоклассовой классификации является назначение каждому образцу (сообщению) одной или нескольких подходящих меток из заданного набора меток ({фишинг, спам, мошенничество, вредоносное ПО, социальная инженерия, нормальное}). Для решения этой задачи используются методы преобразования задачи (Binary Relevance, Classifier Chains, Label Powerset) или адаптации алгоритма. В данной диссертации задача многоклассовой классификации решается с использованием модели случайного леса со стратегией OneVsRestClassifier вместе с встраиванием BERT. Этот выбор основан на объединении способности BERT к глубокому пониманию текста и стабильности Random Forest с простотой и эффективностью OneVsRestClassifier при решении многометковых задач.

1.4. Роль методов разведки с открытыми источниками (OSINT) в деанонимизации: теоретические основы

Анонимность в Интернете создаёт благоприятные условия для киберпреступников, осуществляющих свою противоправную деятельность. В борьбе с мошенничеством важно не только выявлять вредоносные сообщения, но и пытаться деанонимизировать лиц или группы, стоящие за ними. Методы разведки на основе открытых источников (OSINT) являются одним из основных инструментов в этом процессе. В данном разделе будут рассмотрены понятие OSINT, его цели и задачи, а также основные методы OSINT, используемые в процессе деанонимизации, а также его правовые и этические аспекты.

Разведка с открытыми источниками (OSINT) – это процесс и результат получения разведывательной информации путем сбора, обработки, анализа и распространения информации из общедоступных открытых

источников [12]. К открытым источникам относятся Интернет (веб-сайты, форумы, блоги, социальные сети, поисковые системы), средства массовой информации (газеты, журналы, телевидение, радио), научные публикации, правительственные и коммерческие отчеты, географические данные и другие общедоступные материалы. Этот подход широко используется в сферах кибербезопасности, журналистики, правоохранительной деятельности, разведки и социальных исследований, поскольку позволяет получать важную информацию о конкретных лицах или организациях, не прибегая к секретным или неавторизованным источникам. Основные цели OSINT могут быть разнообразными, но в контексте кибербезопасности и борьбы с мошенничеством они включают сбор информации, выявление связей, деанонимизацию, оценку угроз и сбор доказательств. Методы OSINT могут быть использованы для получения ценной информации о конкретном лице, организации или событии и особенно эффективны для деанонимизации анонимных или скрытых пользователей и разоблачения схем онлайн-мошенничества. Задачи OSINT включают в себя конкретные действия, направленные на достижение этих целей: выявление источников, сбор данных в автоматическом или ручном режиме, фильтрацию, структурирование, анализ, интерпретацию и представление результатов полученной информации.

Деанонимизация – сложный и многоуровневый процесс, который обычно включает сбор, анализ и идентификацию «цифровых следов» человека. Цифровые следы – это данные, оставленные пользователем в интернете: IP-адреса, имена пользователей, фотографии профилей, стиль письма, геолокационные метки и многое другое.

Процесс деанонимизации особенно затруднен на платформах, где анонимность играет первостепенную роль, таких как Telegram, поскольку такие системы обеспечивают лишь ограниченную видимость персональных данных пользователей. Однако можно частично или полностью деанонимизировать аккаунты Telegram, эффективно

используя определённые методы OSINT (разведки с открытым исходным кодом).

Основные методы OSINT, обычно используемые для деанонимизации в Telegram, можно суммировать в таблице ниже:

№	Метод OSINT	Краткое описание	Возможные результаты
1	Поиск по имени пользователя Подсчет и анализ имен пользователей/ никнеймов	Проверьте имя пользователя Telegram на различных онлайн-платформах (социальных сетях, форумах, поисковиках). Инструменты: WhatsMyName.app, Sherlock, Maigret	Другие профили, имена, фотографии, интересы, контактные данные
2	Анализ номера телефона	Проверка известного номера телефона с помощью специальных сервисов или поисковых систем, с соблюдением законов о конфиденциальности	Связанные аккаунты, рекламные сообщения, записи в скомпрометированных базах данных
3	Анализ адреса электронной почты	Проверьте известный адрес электронной почты в поисковых системах, социальных сетях и сервисах обнаружения утечек данных (Меня взломали?)	Связанные учетные записи, сообщения на форуме, пароли в скомпрометированных данных (если они раскрыты)
4	Анализ следов в социальных сетях	Сравнение профилей, анализ контента (посты,	Подтверждение принадлежности профилей

	(SOCMINT)	комментарии, лайки), анализ связей (друзья, подписчики) (методы SNA), обратный поиск изображений (Google Images, TinEye)	одному и тому же человеку, интересов, взглядов, места жительства, круга общения, оригинальной фотографии
5	Анализ метаданных	Анализ метаданных (EXIF) опубликованных файлов (изображений, документов)	Модель устройства, время съемки, географические координаты
6	Анализ утечки данных	Поиск известных адресов электронной почты, имен пользователей и номеров телефонов во взломанных базах данных (Have I Been Pwned?, DeHashed, IntelX.io)	Украденные логины, пароли (хэшированные или раскрытые), персональные данные и другая сопутствующая информация

Таблица 1.2 Основные методы OSINT, используемые при деанонимизации

Используя эти методы по отдельности или в сочетании, можно постепенно собрать информацию об анонимном пользователе и создать его «цифровой портрет». Однако процесс деанонимизации не всегда успешен и требует тщательной проверки достоверности полученной информации.

При использовании методов OSINT крайне важно соблюдать правовые и этические нормы [13]. Хотя OSINT в основном работает с открытой и общедоступной информацией, существуют определённые границы. С юридической точки зрения, необходимо избегать таких действий, как сбор информации, которую можно получить только законным путём, несанкционированный доступ к

чужим учётным записям, а также незаконное получение и распространение персональных данных. В каждой стране действует своё законодательство о защите персональных данных (GDPR в ЕС, Закон «О персональных данных и их защите» в Казахстане), и его требования должны неукоснительно соблюдаться. Также важно помнить, что использование методов OSINT в злонамеренных целях (преследование, шантаж) карается законом [14].

Методы OSINT основаны не только на технических инструментах, но и на аналитических навыках эксперта. Правильный отбор информации и её распознавание от фейковых данных – один из важнейших этапов процесса OSINT. В настоящее время такие методы, как сбор информации с таких платформ, как Telegram, Twitter, Instagram, изучение профилей пользователей и выявление связей между аккаунтами, являются реальными примерами эффективного использования открытой информации. Кроме того, инструменты OSINT – Maltego, SpiderFoot, Recon-ng и др. – позволяют следователям и исследователям визуально отображать сложные данные и проводить структурный анализ. Методы OSINT должны осуществляться с соблюдением правовых и этических норм, поскольку открытая информация сама по себе может привести к вторжению в личную жизнь. По этой причине исследователям необходимо тщательно оценивать источник, точность и цели использования информации.

Существует множество коммерческих и открытых инструментов для сбора и анализа информации в сфере кибербезопасности и OSINT. Наряду с популярными зарубежными решениями, стоит обратить внимание и на отечественные разработки. В частности, Kazdream Group предлагает ряд специализированных программных комплексов. Среди них можно отметить систему «Аргус» для оперативного поиска и анализа преступной деятельности в интернете. Этот комплекс может стать важным инструментом для правоохранительных органов. Кроме того, компания разработала систему «Айдис» для мониторинга и

сбора утечек зарубежных данных и инструмент «Тритон», позволяющий эффективно визуализировать результаты анализа больших объемов данных. Наличие этих решений свидетельствует о растущем отечественном потенциале в области информационной безопасности и анализа данных. Однако задача выявления и деанонимизации мошенничества в мессенджере Telegram, рассматриваемая в данной диссертационной работе, имеет свою специфику, и разработка комплексных систем, специально адаптированных для этой цели, сочетающих методы обработки естественного языка и машинного обучения с возможностями OSINT, еще требует дальнейших исследований.

2. РАЗРАБОТКА СИСТЕМЫ ОБНАРУЖЕНИЯ И ДЕАНОМИНАЦИИ МОШЕННИЧЕСТВА В TELEGRAM

2.1. Архитектура системы и взаимодействие модулей

При разработке любого сложного программного обеспечения тщательное проектирование архитектуры системы является ключевым условием обеспечения её надёжности, масштабируемости, поддерживаемости и общей эффективности [15]. Качество архитектурных решений особенно важно для систем, работающих с большими данными, выполняющих анализ в реальном времени и интегрирующих множество технологических компонентов. В связи с этим архитектура разрабатываемой системы обнаружения мошенничества и деанонимизации проектировалась как структура, отвечающая современным требованиям, обладающая высокой гибкостью и устойчивостью.

В данном разделе подробно рассматривается общая архитектура системы, её основные модули и механизмы их взаимодействия. Структура системы построена по модуль-

ному принципу, то есть каждый модуль выполняет определённую задачу и логически отделён от других модулей. Такой подход позволяет легко модифицировать, обновлять и расширять программный комплекс, поскольку каждый модуль может тестироваться и совершенствоваться независимо.

Разработанная система состоит из нескольких важных модулей, каждый из которых выполняет свои специфические функции. Прежде всего, важную роль играет модуль-сборщик, отвечающий за сбор данных с платформы Telegram. Этот модуль автоматизирует сбор сообщений, данных пользователей и других метаданных из определенных каналов и групп Telegram. Собранные данные передаются в следующие модули системы для дальнейшей обработки и анализа.

Собранные данные проходят первичную обработку с помощью модуля предварительной обработки и конструирования признаков. На этом этапе удаляются избыточные, ненужные фрагменты данных, тексты преобразуются в векторный формат и выбираются ключевые признаки, способствующие эффективному выявлению мошенничества. Подготовленные данные отправляются в модуль выявления мошенничества.

Модуль обнаружения мошенничества является основным аналитическим центром системы. Он использует предобученные BERT-внедрения и классификатор Random Forest для автоматического выявления признаков мошенничества. Основываясь на принципе многосимвольной классификации, этот модуль может одновременно выявлять несколько категорий мошенничества в одном сообщении. Эта функция особенно важна при анализе сложных, многоаспектных мошеннических схем на платформе Telegram.

При обнаружении подозрительной активности активируется модуль деанонимизации. Этот модуль использует инструменты OSINT для сбора дополнительной информации из открытых источников. Популярные платформы OSINT, такие как WhatsMyName, Sherlock и Maigret, используются для по-

иска каналов связи пользователей Telegram, профилей в других социальных сетях и дополнительных цифровых следов. Эта информация поддерживает процесс деанонимизации, то есть позволяет определить истинную личность пользователя Telegram.

Все собранные и обработанные данные, а также результаты работы системы хранятся в NoSQL-базе данных MongoDB. Эта структура базы данных гибкая и подходит для работы с большими объемами неструктурированных данных. Кроме того, MongoDB обеспечивает быстрый поиск и обработку данных системы.

Заключительным компонентом системы является удобный веб-интерфейс. Этот компонент разработан с использованием фреймворка Flask и позволяет визуализировать полученные результаты, создавать отчёты и просматривать подозрительные сообщения. Веб-интерфейс предоставляет пользователю удобную среду для управления системой, мониторинга отдельных каналов и принятия решений на основе полученных данных.

Еще одной важной особенностью архитектуры системы является реализация всех модулей в Docker-контейнерах. Это решение позволяет управлять каждой частью системы в изолированной среде, снижает влияние на всю систему в случае сбоя и обеспечивает легкое масштабирование. С помощью инструмента Docker Compose можно запускать и управлять всеми микросервисами одной командой.

В целом, разработанная система представляет собой эффективное сочетание комплексных архитектурных решений. Модульная, микросервисная, контейнерная и многоуровневая архитектура повышает стабильность и гибкость системы, позволяя точно, быстро и эффективно выявлять и пресекать мошеннические действия на платформе Telegram. Система спроектирована с учётом масштабируемости для лёгкого дальнейшего усовершенствования и адаптации к другим платформам.

Основная цель системы – сбор данных из мессенджера Telegram, их анализ для выявления мошеннических действий, деанонимизация подозрительных аккаунтов и визуализация результатов в удобном виде. Для достижения этих целей система основана на архитектуре, состоящей из нескольких взаимосвязанных микросервисов. Организация системы по микросервисному принципу позволяет повысить её стабильность и гибкость, а также обеспечить независимую работу каждого компонента. Микросервисная архитектура в настоящее время является одним из наиболее эффективных подходов, широко применяемых в сложных программных комплексах. Её главное преимущество заключается в том, что благодаря модульному построению системы каждый сервис может быть реализован отдельно и обновляться независимо при необходимости.

Ещё одной важной особенностью микросервисного подхода является возможность масштабирования системы. Объём данных, собираемых из Telegram, может со временем увеличиваться. В таком случае управление всеми сервисами в единой монолитной системе, вероятно, негативно скажется на её производительности. В случае микросервисной архитектуры можно распределить нагрузку только на один конкретный сервис и масштабировать его на отдельном сервере или контейнере. Например, если процесс сбора данных из Telegram усложнится и появится большой поток сообщений, достаточно будет просто увеличить сервис-сборщик и повысить его производительность.

Кроме того, микросервисная архитектура позволяет выбирать независимый технологический стек для каждого сервиса. Это означает, что каждый компонент может быть реализован на наиболее подходящем для него языке программирования или платформе. В результате, например, сервисы, обрабатывающие текстовые данные, могут быть написаны на Python, а для управления веб-интерфейсом могут использоваться Flask или другие веб-фреймворки. Такая гибкость расширяет технические возможности системы и от-

крывает возможность эффективного использования современных инструментов программирования.

Для управления и координации микросервисов использовалась технология контейнеров Docker. Система Docker позволяет каждому микросервису иметь собственный контейнер и запускать его в среде, независимой от внешней среды. Контейнеры Docker обеспечивают быстрый запуск системы и лёгкое портирование на другие серверы или рабочие среды. Кроме того, для обеспечения корректной работы системы использовался инструмент Docker Compose. Docker Compose – это удобное решение для определения, настройки и управления приложениями, состоящими из нескольких контейнеров. С помощью этого инструмента можно запустить всю систему одной командой и предварительно определить необходимые параметры для каждого сервиса.

Контейнерная архитектура системы также повышает её устойчивость к техническим сбоям. Если какой-либо сервис системы перестаёт работать, остальные сервисы продолжают работать. Это обеспечивает стабильность и надёжность системы. Кроме того, поскольку каждый сервис независим, для его обновления или улучшения не требуется останавливать всю систему. Достаточно остановить только нужный контейнер и перезапустить обновлённую версию.

Эти архитектурные решения значительно упрощают процессы разработки, тестирования и развертывания. Особенно на этапе разработки они позволяют сократить количество ошибок и ускорить тестирование, создавая платформенно-независимую среду. Преимущества контейнеров Docker также особенно заметны при миграции системы на новые серверы или облачные платформы. Контейнеры содержат все зависимости приложения и гарантируют одинаковую производительность в любой точке.

Таким образом, микросервисная архитектура и контейнерная организация системы обеспечивают её высокую надёжность, гибкость, масштабируемость и простоту технической поддержки. Такой подход не только обеспечивает

долгосрочную и стабильную работу системы, но и позволяет добавлять новые сервисы или легко обновлять существующие в будущем. Данная архитектура считается одним из наиболее эффективных решений для обнаружения, анализа и противодействия мошенническим действиям на платформе Telegram.

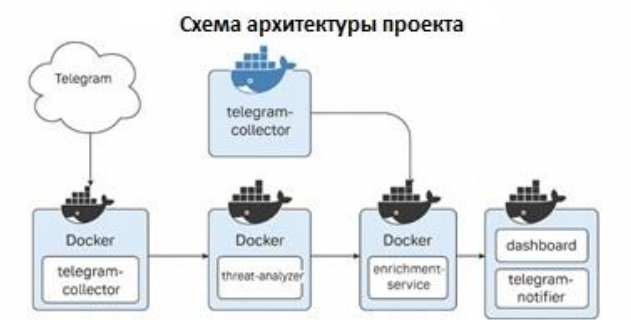

Рисунок 2.1 Общая архитектура разработанной системы

На рисунке 2.1 представлена схема общей архитектуры разработанной системы. На этой схеме наглядно представлены основные компоненты системы, реализованные с использованием Docker-контейнеров, и их взаимодействие. Как показано, система состоит из Docker-контейнеров, разделённых на несколько логических блоков. К ним относятся сбор данных (collector), анализ данных и обнаружение угроз (threat_analyzer), обогащение данных (enrichment_service), веб-приложение (API) (app), служба уведомлений (notifier) и база данных MongoDB для хранения данных. Использование технологии Docker позволяет запускать каждый компонент системы в изолированной среде, управлять зависимостями между ними и легко развертывать систему на различных платформах [17]. Каждый контейнер выполняет определённую функцию и взаимодействует с другими контейнерами через сетевые

интерфейсы. Такое архитектурное решение повышает стабильность системы и упрощает процесс добавления новой функциональности или изменения существующей. Служба сбора данных непрерывно собирает данные с помощью API Telegram и отправляет их в базу данных MongoDB после предварительной обработки. Служба threat_analyzer принимает эти данные и использует модели машинного обучения для поиска признаков мошенничества. Результаты записываются обратно в базу данных и отображаются в пользовательском интерфейсе через службу приложения. Служба уведомлений отправляет уведомления соответствующим сторонам при обнаружении критических событий или ситуаций высокого риска. Служба обогащения, в свою очередь, предназначена для сбора дополнительной информации о выявленных подозрительных объектах и содействия процессу деанонимизации.

Архитектура системы, представленная на рисунке 2.1, характеризуется как сложная структура, полностью отвечающая основным требованиям современных микросервисных подходов. Данная архитектура предусматривает разделение системы на несколько независимых, но взаимосвязанных функциональных модулей. Такой подход существенно повышает гибкость, надежность и масштабируемость системы. Реализация каждого модуля в Docker-контейнере позволяет ему работать в собственной операционной среде, изолированной от других контейнеров. Это предотвращает полную остановку системы при отказе одного из модулей или необходимости обновления. Кроме того, каждый контейнер системы специализирован для решения конкретной задачи, что, в свою очередь, упрощает управление и настройку компонентов.

Важной частью системы является модуль Collector, который собирает данные из Telegram. Этот сервис непрерывно собирает сообщения из необходимых каналов, групп и пользователей через открытый API-интерфейс Telegram. Модуль Collector выполняет первичную обработку собранных сооб-

щений, преобразует их структуру в удобный для системы формат и отправляет эти данные в базу данных MongoDB. Выбор MongoDB не случаен, поскольку она позволяет быстро хранить и обрабатывать большие объёмы полуструктурированных данных. Сообщения в Telegram могут содержать различную информацию, такую как изображения, ссылки, идентификаторы пользователей, помимо текста, поэтому гибкая архитектура базы данных очень важна.

Поступившие в базу данных сообщения обрабатываются модулем threat_analyzer. Этот модуль анализирует содержимое каждого сообщения с помощью предобученных моделей на основе BERT и эффективных алгоритмов классификации, таких как Random Forest, выявляя признаки потенциального мошенничества. Threat_analyzer предназначен для выявления конкретных сценариев мошенничества (инвестиционные мошенничества, фишинг, фальшивые выигрыши и т.д.) и позволяет одновременно выявлять несколько признаков опасности в одном сообщении, используя метод многосимвольной классификации. Это значительно повышает способность системы распознавать сложные схемы мошенничества в Telegram.

После обнаружения угроз активируется модуль enrichment_service. Этот сервис использует методы OSINT для поиска информации о подозрительных пользователях Telegram из дополнительных открытых источников. Например, имя пользователя в Telegram может совпадать с учётными записями, зарегистрированными на других платформах. Этот модуль собирает потенциальные профили пользователей, использующих WhatsMyName, Sherlock и другие инструменты OSINT, расширяя их онлайн-след и предоставляя системе дополнительный контекст. Такая дополнительная информация очень важна для процесса деанонимизации, поскольку позволяет определить истинную личность скрытого пользователя.

Другим важным компонентом системы является модуль оповещения. Этот сервис позволяет отправлять уведомления

в режиме реального времени о событиях повышенного риска ответственным лицам или специальным группам через ботов в Telegram. Такая система оповещения помогает оперативно реагировать на мошеннические действия и повышать эффективность системы. Данный модуль подходит для мониторинга системы в режиме реального времени и удобен для использования дополнительных каналов на платформе Telegram в качестве системы оповещения.

Компонент, представляющий результаты пользователю, представляет собой веб-приложение. Этот модуль разработан на основе веб-фреймворка Flask и предоставляет панель управления системой и инструменты для визуализации аналитической информации. Через веб-интерфейс пользователь может отслеживать собранные сообщения, обнаруженные угрозы, дополнительную информацию о пользователях и общую работу системы. Такой интерфейс идеально подходит для специалистов и исследователей в области кибербезопасности, поскольку позволяет им получать прямой доступ к данным в режиме реального времени и быстро принимать необходимые меры.

В системе все компоненты связаны между собой посредством Docker и образуют единую сетевую архитектуру. Технология Docker Compose позволяет централизованно запускать, управлять и настраивать все контейнеры системы. Это способствует быстрой установке, тестированию и миграции системы на различные серверы. Кроме того, модульная структура системы позволяет легко расширять её в будущем. Например, можно добавить модуль сбора данных из других мессенджеров, выявить новые сценарии мошенничества или внедрить дополнительные инструменты визуализации.

В целом, архитектура системы, представленной на рисунке 2.1, продумана на высоком уровне в соответствии с современными требованиями. Здесь эффективно используются все преимущества микросервисного подхода: гибкость, масштабируемость, работа в изолированной среде и стабильность. Выполнение каждым модулем определённых функций

делает систему удобной для управления, совершенствования и надёжного использования с технической и прикладной точки зрения. Такая сложная архитектура является важнейшим инструментом для эффективного мониторинга, обнаружения и анализа мошеннических схем на платформе Telegram. Данная система считается важным проектом в области кибербезопасности, имеющим практическую ценность и требующим дальнейшего развития.

Рисунок 2.2 Обнаружение мошенничества и деанонимизация. Концептуальная диаграмма

Концептуальная схема процесса обнаружения мошенничества и деанонимизации представлена на рисунке 2.2. Эта схема описывает полную цепочку работы системы от этапа сбора начальных данных до их анализа, обнаружения мошенничества, попытки деанонимизации и представления результатов пользователю. Процесс начинается со сбора данных из мессенджера Telegram (источник данных: Telegram). Эти данные могут включать сообщения, профили пользователей, активность в группах и каналах, а также другие метаданные. Модуль Collector собирает эти данные и выполняет первичную обработку (форматирование, удаление избыточной информации). Затем данные поступают на этап Preprocessing & Feature Engineering, где текстовые данные преобразуются в векторную форму, выявляются значимые признаки и подготавливаются для моделей машинного

обучения. Подготовленные данные отправляются в блок Fraud Detection Engine, где предобученные модели классификации и кластеризации (на основе scikit-learn, Transformers) выявляют признаки мошенничества. Обнаруженные подозрительные действия или учетные записи поступают на этап De-anonymization Attempt. На этом этапе собирается дополнительная информация из открытых источников (OSINT) с помощью сервиса enrichment_service, что позволяет связать подозреваемого с реальным человеком или другими онлайн-идентификаторами. Все обработанные данные, результаты анализа и прогнозы деанонимизации сохраняются в хранилище данных (MongoDB). Сохранённая информация визуализируется, формируются отчёты и представляются в удобном интерфейсе с помощью компонента «Панель мониторинга/Отчётность». Эта концептуальная схема помогает понять логику работы системы и основные направления потоков данных.

Концептуальная схема процесса обнаружения мошенничества и деанонимизации описывается как сложная система, состоящая из многоэтапных, взаимосвязанных этапов, как показано на рисунке 2.2. Эта схема подробно объясняет базовую структуру системы, роль каждого модуля и направления движения данных. Основная цель системы – своевременное выявление мошеннических действий на платформе Telegram, их анализ и, по возможности, сбор дополнительной информации для идентификации лиц, стоящих за подозрительными аккаунтами.

Начальным этапом процесса является сбор данных из мессенджера Telegram. В настоящее время Telegram является широко используемым средством коммуникации во всем мире, а некоторые архитектурные особенности, в частности, шифрование сообщений, анонимность и большое количество открытых групп, делают его подходящей платформой для организации мошеннических действий. Система использует специальный модуль Collector для сбора данных из Telegram. Модуль Collector автоматически собирает информацию из

открытых каналов, групп и чатов пользователей, включая сообщения, данные профилей пользователей и активность в каналах, используя API Telegram. Эти данные изначально неструктурированы и представлены в различных форматах.

Собранные данные отправляются на этап предварительной обработки и проектирования признаков. На этом этапе данные подвергаются систематической обработке. В первую очередь из сообщений удаляются такие элементы, как избыточные символы, ненужные HTML-теги или ссылки. Затем тексты преобразуются в строчные буквы, удаляются стоп-слова и при необходимости применяются методы лемматизации или стемминга. Для преобразования текстовых данных в числовые векторы используются методы BERT-внедрения. Кроме того, анализируются дополнительные признаки, такие как время регистрации пользователя, частота активности, наличие номера телефона и признаки имени пользователя. Этот этап обеспечивает приведение данных к качественному и релевантному формату для использования в моделях машинного обучения.

Следующий основной этап – это модуль обнаружения мошенничества (Fraud Detection Engine). В этом блоке реализуются предобученные модели классификации. Система использует ансамблевый классификатор Random Forest и дополнительные методы кластеризации для обработки текстовых векторов на основе векторов BERT. Эти модели выявляют лингвистические и структурные особенности, характерные для мошенничества в сообщениях Telegram. Система использует принцип многосимвольной классификации, то есть одно сообщение может быть отнесено к нескольким типам мошенничества. Например, одно сообщение может содержать поддельное инвестиционное предложение, фишинговую ссылку и поддельный приз. На этом этапе каждому сообщению присваивается потенциальный уровень риска и прогноз, к какому типу мошенничества оно относится.

После обнаружения признаков мошенничества процесс переходит к этапу «Попытка деанонимизации». На этом эта-

пе методы OSINT используются для полного исследования пользователя Telegram и выявления других следов его пребывания в интернете. С помощью модуля enrichment_service выполняется поиск основных данных, таких как никнейм пользователя (имя пользователя), идентификационный номер, номер телефона в Telegram на других открытых платформах. Для этого используются инструменты OSINT WhatsMyName, Maigret, Sherlock и другие. Цель – связать пользователя, анонимного в Telegram, с другими аккаунтами в интернете, определить его возможную реальную личность или активность в других сетях. Этот этап очень важен для правоохранительных органов, поскольку открывает возможность для точной идентификации мошенников.

Все собранные и обработанные данные, включая сообщения Telegram, сделанные на их основе прогнозы, выявленные виды мошенничества, результаты OSINT-анализа и операции деанонимизации, хранятся в разделе «Хранилище данных». Для этой цели была выбрана база данных MongoDB. MongoDB – широко используемая NoSQL-база данных для быстрого хранения и анализа полуструктурированных и неструктурированных данных. Учитывая разнообразие данных, получаемых из Telegram, такая база данных полностью удовлетворяет потребности системы.

Заключительный этап – панель мониторинга/отчётность. В этом разделе собранные и обработанные данные наглядно представлены пользователю. Веб-интерфейс, построенный на веб-фреймворке Flask, отображает отчёты об угрозах, статистические данные, схемы мошенничества, расширенную информацию о пользователе и результаты деанонимизации. Веб-интерфейс позволяет операторам системы быстро работать с данными и оперативно реагировать на мошеннические действия. Кроме того, этот интерфейс является удобным инструментом для специалистов по кибербезопасности для проведения ежедневного мониторинга.

В целом, концептуальная схема, представленная на рисунке 2.2, описывает полный и логически связный поток данных для обнаружения и деанонимизации мошеннических действий на платформе Telegram. Эта схема наглядно показывает, как данные собираются, обрабатываются, анализируются по каким моделям и, наконец, визуализируются. Такая комплексная система полностью отвечает требованиям современной кибербезопасности и является эффективным инструментом противодействия мошенническим схемам в мессенджере Telegram. Кроме того, данная схема обеспечивает гибкую структуру, подходящую для будущей интеграции с другими мессенджерами или добавления новых OSINT-платформ.

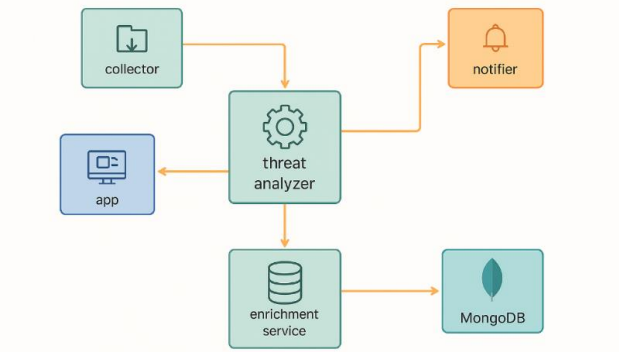

Рисунок 2.3 Схема взаимодействия компонентов системы

Взаимодействие основных функциональных компонентов системы более подробно проиллюстрировано на рисунке 2.3. На этой схеме показаны пути обмена данными и управляющие сигналы между сборщиком, анализатором угроз, приложением, уведомителем, сервисом обогащения и базой данных MongoDB.

Модуль сбора данных напрямую взаимодействует с API Telegram и собирает новые сообщения, действия пользователей и другие релевантные данные. Эти данные сначала сохраняются в специальных коллекциях в базе данных MongoDB. Модуль сбора данных имеет параметры, позволяющие настраивать частоту и объем сбора данных.

Модуль threat_analyzer считывает данные из MongoDB, как новые, так и отсортированные по определённым критериям, периодически или с помощью триггеров. Он выполняет предварительную обработку данных (очистку, токенизацию, векторизацию) и использует модели машинного обучения, основанные на библиотеках scikit-learn и Transformers. Обнаруженные факты мошенничества, уровни подозрения и другие результаты анализа записываются обратно в MongoDB, часто со ссылками на исходные данные или дополнениями к ним.

Модуль enrichment_service срабатывает при получении уведомления о высокорисковых сущностях или событиях, выявленных threat_analyzer. Он собирает дополнительную информацию, анализируя открытые источники (веб-сайты, социальные сети, форумы), и связывает её с соответствующими записями в MongoDB. Этот процесс способствует деанонимизации и более глубокому пониманию природы угрозы.

Компонент приложения представляет собой RESTful API, построенный на веб-фреймворке Flask. Он принимает запросы из пользовательского интерфейса (Dashboard), извлекает необходимые данные из MongoDB, при необходимости вызывает сервисы threat_analyzer или richment_service и возвращает результаты в формате JSON. Приложение также может предоставлять интерфейс настройки и управления системой.

Модуль уведомлений генерирует уведомления о важных событиях (обнаружение нового высокорискового мошенничества, системная ошибка), поступающие от threat_analyzer или других компонентов системы. Эти уведомления могут быть отправлены соответствующим пользователям или администраторам по электронной почте, через мессенджеры или по другим каналам.

База данных MongoDB служит централизованным и постоянным хранилищем данных для всех этих компонентов. Она хранит необработанные и обработанные данные, результаты моделей машинного обучения, информацию о деаноним-

мизации, пользовательские данные и системные журналы. Гибкая схема MongoDB (без схемы) позволяет хранить данные в различных структурах и легко изменять модель данных по мере развития системы.

Размещение всех компонентов в Docker-контейнерах обеспечивает их изоляцию друг от друга и общую стабильность системы. Взаимодействие между компонентами осуществляется преимущественно посредством сетевых запросов (HTTP REST API) и обмена данными через MongoDB.

Выбранный для реализации проекта стек технологий играет важную роль в достижении поставленных перед системой целей. Ниже перечислены основные используемые технологии и библиотеки, а также причины их выбора:

Python – в качестве основного языка программирования для системы был выбран Python. Его основными преимуществами являются простой синтаксис, наличие множества библиотек (особенно в областях анализа данных, машинного обучения и веб-разработки), а также поддержка активного сообщества. Язык Python позволяет решать все основные задачи системы, такие как сбор, обработка, анализ данных и создание API.

Telethon – для взаимодействия с мессенджером Telegram использовалась асинхронная Python-библиотека Telethon. Эта библиотека позволяет полноценно работать с API Telegram, читать сообщения, получать информацию о пользователях, работать с группами и каналами, а также выполнять другие необходимые операции. Её асинхронность способствует эффективному и быстрому сбору данных.

PyMongo – официальный драйвер PyMongo, используемый для работы с базой данных MongoDB на Python. Эта библиотека обеспечивает подключение к MongoDB, выполняя операции чтения, записи, обновления и удаления данных, а также построение сложных запросов.

Scikit-learn – для решения задач машинного обучения была выбрана широко известная библиотека scikit-learn. Она предоставляет широкий спектр алгоритмов и инструментов,

таких как классификация, регрессия, кластеризация, снижение размерности, выбор моделей и предварительная обработка. Функциональность этой библиотеки активно используется при построении и оценке моделей обнаружения мошенничества.

Transformers (Hugging Face) – библиотека Transformers использовалась для анализа текстовых данных и обработки естественного языка (NLP). Эта библиотека предоставляет доступ к большому количеству предобученных моделей Transformers (BERT, GPT), которые помогают эффективно решать такие задачи, как встраивание текста, классификация текста и распознавание именованных объектов (NER). Она играет важную роль в анализе содержания мошеннических сообщений и выявлении подозрительных лингвистических особенностей.

Для разработки прикладного компонента системы был выбран Flask – микрофреймворк для создания лёгких и гибких веб-приложений и API. Он предоставляет базовую функциональность, необходимую для создания RESTful API, обработки пользовательских запросов и отправки данных на панель управления. Минималистичный подход и расширяемость Flask сделали его оптимальным выбором для этого проекта.

Docker – это мощная платформа для контейнеризации и изоляции всех компонентов системы. Docker позволяет упаковать каждую службу (collector, threat_analyzer, app, notifier, richment_service, MongoDB) вместе с её зависимостями в отдельный контейнер. Это помогает стандартизировать процессы разработки, тестирования и развертывания, избегать проблем, связанных с окружением, и легко масштабировать систему. Docker Compose используется для определения и управления приложениями, состоящими из нескольких контейнеров. Этот технологический стек призван обеспечить надёжность, гибкость, масштабируемость и простоту обслуживания системы.

Используемые методы контейнеризации позволяют поддерживать модульную структуру системы и независимо обновлять каждый компонент. Каждый сервис логически обособлен, что упрощает локализацию системных сбоев. Кроме того, благодаря установлению сетевых соединений между контейнерами обмен данными осуществляется безопасно и контролируемо.

2.2. Реализация модуля сбора данных с использованием API Telegram

Для эффективного функционирования системы обнаружения мошенничества и деанонимизации большое значение имеют качество и полнота исходных данных. Поскольку в данной системе в качестве основного источника был выбран мессенджер Telegram, потребовалось разработать специальный модуль, который бы непрерывно и надежно собирал из него необходимую информацию. Для этого был реализован модуль сбора данных, называемый Telegram-collector. Этот модуль напрямую взаимодействует с API-интерфейсом Telegram и отвечает за сбор новых сообщений, активности пользователей и других важных метаданных [18]. В данном разделе рассматриваются основные аспекты разработки модуля Telegram-collector, в частности, выбор технологического стека, процесс авторизации, логика сбора данных, структура собираемых данных, а также практические примеры работы модуля.

Хотя для взаимодействия с API Telegram на Python доступно несколько библиотек, для данного проекта была выбрана библиотека Telethon [19]. Этот выбор был обусловлен несколькими важными факторами. Во-первых, Telethon полностью асинхронен и тесно интегрирован с модулем Python asyncio. Это обеспечивает высокую производительность при работе с сетевыми операциями, особенно когда необходимо собирать данные из множества чатов одновременно. Асинхронность позволяет эффективно использовать ресурсы и

сокращает время отклика системы. Во-вторых, Telethon предоставляет доступ к низкоуровневым функциям API Telegram, которые могут потребоваться для выполнения нестандартных или сложных запросов. Он поддерживает все основные объекты Telegram (сообщения, чаты, пользователей, медиафайлы) и предоставляет удобный интерфейс для работы с ними. В-третьих, Telethon – активно развивающийся и поддерживаемый проект. Он имеет обширную документацию, множество примеров и активное сообщество, что упрощает процесс разработки и помогает быстро решать любые возникающие вопросы. Кроме того, совместимость Telethon с экосистемой Python позволяет легко интегрировать его с другими библиотеками (PyMongo, scikit-learn), обеспечивая целостность всей системы.

Для работы с API Telegram каждое приложение должно иметь свои уникальные идентификаторы. Эти идентификаторы предназначены для обеспечения безопасности платформы Telegram и предотвращения злоупотреблений API. Процесс авторизации включает следующие ключевые элементы:

API ID и API Hash – это значения, которые получаются при регистрации нового приложения через учётную запись разработчика на официальном сайте Telegram (my.telegram.org). API ID – это числовой идентификатор приложения, а API Hash – секретный ключ, принадлежащий этому приложению. Эти два значения используются для аутентификации клиента при каждом подключении к API. Крайне важно сохранять их конфиденциальность.

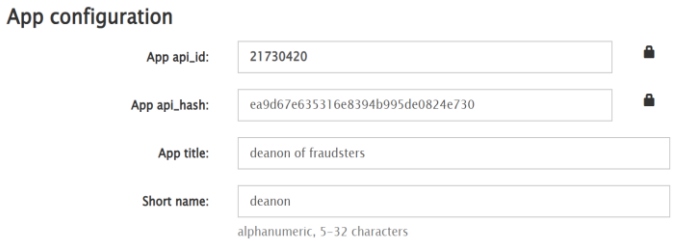

Рисунок 2.4 Конфигурация приложения Telegram API

На рисунке 2.4 показан пример интерфейса настройки приложения Telegram API на сайте my.telegram.org. Здесь после заполнения разработчиком таких полей, как «Название приложения» и «Краткое название», и выбора платформы ему предоставляются уникальные значения api_id и api_hash. Эти значения вносятся в конфигурационный файл модуля Telegram-collector и используются для установления связи с серверами Telegram.

Номер телефона – для подключения к API через библиотеку Telethon требуется действительный номер телефона. При первом подключении Telegram отправляет на этот номер код подтверждения. После ввода кода Telethon сохраняет файл сессии (с расширением session) на локальном диске. Этот файл сессии устраняет необходимость повторной авторизации при последующих подключениях, если сессия действительна. В целях безопасности рекомендуется использовать отдельный номер телефона для сбора данных, не связанный с основной учётной записью.

Авторизация выполняется автоматически при запуске модуля Telegram-сборщика. Если корректный файл сессии не найден, модуль запрашивает у пользователя номер телефона, а затем проверочный код, полученный от Telegram. После успешной авторизации модуль начинает сбор данных.

Модуль сбора телеграмм может собирать данные в двух основных режимах: получение новых сообщений в реальном времени и загрузка исторических данных из ранее отправленных сообщений [20].

В режиме сбора новых сообщений модуль использует обработчик событий Telethon events.NewMessage. Этот обработчик срабатывает при поступлении нового сообщения в указанные чаты (группы, каналы) или от отдельных пользователей. Конфигурация модуля определяет, какие чаты прослушивать. При получении каждого нового сообщения его содержимое (текст, медиафайлы, информация об отправителе и т.д.) извлекается и сохраняется в базе данных MongoDB в соответствии с предопределённой

структурой. Такой подход позволяет системе быстро реагировать на мошеннические действия.

Сбор исторических данных используется, когда необходимо собрать предыдущие сообщения из конкретного чата или пользователя. Для этого используется функция client.iter_messages() сервиса Telethon. Эта функция позволяет перебирать сообщения из конкретного чата. Вы можете указать различные параметры, такие как идентификатор сообщения, с которого начать сбор (offset_id), количество сообщений для сбора (limit), сбор с определенной даты (offset_date) или в обратном направлении (reverse=True). При сборе исторических данных важно учитывать ограничения по скорости, установленные API Telegram. Слишком частая отправка запросов может привести к временной блокировке, поэтому необходимо устанавливать задержки между запросами или разбивать процесс сбора на этапы.

В процессе сбора данных предусмотрены механизмы обработки ошибок. В случае обрыва сетевого соединения или получения некорректного ответа от API модуль пытается переподключиться или повторить запрос через определённое время. Также ведутся логи процесса сбора данных, что помогает выявлять и исправлять ошибки.

Каждое сообщение, полученное через API Telegram, содержит большой объём информации. Модуль сбора Telegram хранит эту информацию в структурированном формате в базе данных MongoDB. Основные поля, хранящиеся в каждом сообщении, включают:

– message_id: уникальный числовой идентификатор сообщения в чате.

– chat_id: уникальный идентификатор чата (группы, канала или личной беседы), в который было отправлено сообщение.

– Текст: Текст сообщения. Это поле может быть пустым, если сообщение не содержит текста.

– sender_id: уникальный идентификатор пользователя, отправившего сообщение.

– временная метка (или поле даты в объекте Telethon): время отправки сообщения (в формате временной метки Unix или объекта datetime).

– reply_to_msg_id: Если сообщение было отправлено в ответ на другое сообщение, то значение message_id сообщения, на которое был дан ответ.

– медиа: Информация о медиаконтенте в сообщении (фото, видео, аудио, документ, наклейка). Это поле может включать тип медиа, идентификатор файла и другие метаданные.

– сущности: информация о конкретных элементах (ссылках, хэштегах, адресах электронной почты, командах бота, упоминаниях) в теле сообщения. Сохраняются тип, начальная позиция и длина каждого элемента.

– просмотры: Если сообщение было опубликовано в канале, количество просмотров, которые оно получило.

– пересланные: счетчик того, сколько раз сообщение было переслано в другие чаты.

– grouped_id: Если сообщение является частью медиа-коллекции (альбома), идентификатор этой коллекции.

Кроме того, можно сохранять дополнительную информацию, например, было ли сообщение отредактировано (edited), является ли отправитель ботом (via_bot_id) или было ли сообщение закреплено (pinned). Формат BSON (Binary JSON), аналогичный JSON, используется для хранения данных в MongoDB. Этот формат поддерживает гибкую схему, то есть разные сообщения могут содержать разные наборы полей, что хорошо подходит для разнообразных данных, поступающих из API Telegram.

Журналирование играет важную роль в мониторинге работы модуля сбора телеграмм и выявлении ошибок. Модуль записывает информацию о своих основных операциях в стандартный поток вывода или в специальный файл журнала.

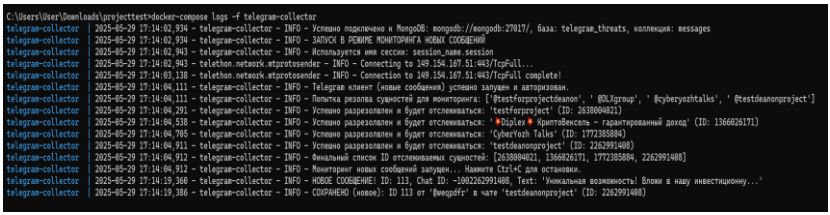

Рисунок 2.5 Журнал приема и хранения сообщений модуля сбора телеграмм

На рисунке 2.5 показан фрагмент файла журнала модуля Telegram-Collector во время работы. В этом журнале отображается информация о получении нового сообщения, его основных атрибутах (message_id, chat_id, sender_id) и его успешном сохранении в базе данных MongoDB. Этот фрагмент журнала показывает, что компонент Telegram_collector.collector получил новое сообщение (id=113) из чата – 1002262991408 от отправителя 2262991408. Затем компонент Telegram_collector.database сообщает о подготовке этого сообщения к добавлению в базу данных и его успешном сохранении. Уровни журнала (INFO, DEBUG) позволяют отображать информацию с различной степенью детализации. Такие журналы помогают контролировать производительность системы, подтверждать сбор данных и при необходимости облегчают процесс устранения неполадок.

В заключение отметим, что модуль сбора телеграмм является важным звеном в сборе первичных данных для системы обнаружения мошенничества. Эффективное использование библиотеки Telethon, корректно настроенный процесс авторизации, надежная логика сбора новых и исторических данных, а также полная структура собираемых данных создают основу для достижения качественных результатов на последующих этапах работы системы (анализ, моделирование, деанонимизация). Стабильная работа модуля и механизмы логирования повышают его надежность.

2.3. Конвейер предварительной обработки собранных текстовых данных

Необработанные текстовые данные, собранные через API Telegram, обычно не подходят для прямого анализа. Они могут содержать избыточные элементы, шум и различную информацию, снижающую эффективность моделей машинного обучения. Поэтому предварительная обработка текстовых данных является неотъемлемой частью любой задачи в области обработки естественного языка (NLP) и машинного обучения (ML) [21]. Основная цель этого процесса – удалить из необработанных текстовых данных лишний шум, извлечь важную информацию и преобразовать данные в формат, удобный для восприятия моделями. Для этой цели был разработан специальный модуль text_processor.py. Этот модуль реализует конвейер, последовательно обрабатывающий тексты собранных сообщений Telegram. Основные этапы этого конвейера и причины их выбора подробно обсуждаются ниже.

Очистка текста – первый и самый важный этап предобработки. Она направлена на удаление из текста элементов, мешающих анализу или имеющих незначительную семантическую нагрузку. Модуль text_processor.py реализует основные этапы очистки, такие как уменьшение регистра, удаление URL-адресов, удаление имени пользователя, удаление знаков препинания и удаление чисел. Объединение текстовых данных в единый, обычно строчный, набор символов предотвращает обработку одинаковых слов, таких как «Word», «word» и «WORD», как разных токенов, что сокращает размер символьного пространства и упрощает процесс обучения модели. Текст «Осторожно, мошенники!» преобразуется в «Осторожно, мошенники!». URL-адреса, часто встречающиеся в сообщениях Telegram (http://example.com, t.me/channel_name), могут быть важны для анализа домена в контексте выявления мошенничества, но добавляют ненужный «шум» в общий семантический анализ текста. В этом проекте URL-адреса обнаруживаются и удаляются с помо-

щью регулярных выражений, поскольку больше внимания уделяется самому текстовому содержимому. Имена пользователей в Telegram в формате @username, как и URL-адреса, считаются избыточными при общем анализе текста и излишне увеличивают пространство символов, поэтому они также обнаруживаются с помощью регулярных выражений и удаляются или заменяются специальным токеном. Знаки препинания (точки, запятые, вопросительные знаки, восклицательные знаки, кавычки, скобки и т.д.) помогают людям понять грамматическую структуру и смысл текста, но считаются избыточными элементами для многих моделей обработки естественного языка. Удаление знаков препинания помогает разделять слова в их чистом виде и повышать согласованность токенов, но в некоторых случаях (для выявления сарказма или эмоциональной окраски) стоит учитывать, что пунктуация может быть важна. Поскольку основной целью этого проекта является обнаружение мошеннического контента, большая часть знаков препинания удаляется. Роль цифр в тексте зависит от контекста; в некоторых случаях они могут нести важную информацию (номера телефонов, номера банковских карт, суммы), но при общем анализе текста они часто оказываются избыточными, значительно увеличивая количество признаков и снижая обобщающую способность модели. В данном проекте предусмотрено удаление цифр или их замена специальным токеном <NUMBER>, выбор зависит от конкретной модели и задачи.

Стоп-слова – это слова, которые очень распространены в языке, но обычно несут небольшую семантическую нагрузку. В казахском языке к ним относятся такие союзы, как «и», «мне», «сен», «ол», «бир», «бар», «но», «для», «о», наречия, причастия, некоторые местоимения и часто используемые глаголы. Удаление стоп-слов имеет ряд преимуществ, таких как сокращение объема данных, повышение эффективности вычислений и фокусировка на более важных словах [22]. В этом проекте используется специальный список стоп-слов для казахского языка, который можно дополнять или изме-

нять по мере необходимости. Следует отметить, что в некоторых случаях, особенно когда важно понимание контекста (при использовании трансформационных моделей, таких как BERT), удаление стоп-слов может быть излишним или даже вредным, поскольку они могут повлиять на структуру и смысл предложения.

Для дальнейшей нормализации текстовых данных можно использовать методы лемматизации или стемминга. Лемматизация – это процесс приведения слов к их исходной, словарной форме (лемме) (слова «окып эрыг», «окыды», «окитын» приводятся к лемме «оку»). Этот процесс требует морфологического анализа слова и обычно сложнее стемминга, но даёт более точные результаты. Для агглютинативных языков, таких как казахский, создание качественного лемматизатора – сложная задача. Стемминг же – более агрессивный процесс, направленный на поиск корня (основы) слова, но не обязательно приводящий к осмысленному слову (слова «окушылы», «укулык», «окучытычы» можно привести к корню «оку»). Алгоритмы стемминга обычно работают быстрее, но могут быть подвержены ошибкам. Для традиционных моделей машинного обучения лемматизация или стемминг могут дополнительно сузить пространство признаков и повысить обобщающую способность модели. Однако, поскольку данная диссертация посвящена преобразующим моделям, таким как BERT, и ограниченному количеству надежных и доступных лемматизаторов/стеммеров для казахского языка, эти шаги не были включены в проект, поскольку модели BERT могут учитывать различные формы слов и их контекстные значения, поэтому лемматизация/стемминг не требуются.

Выбор стратегии предобработки текста часто зависит от типа используемой модели машинного обучения. В BERT (и других моделях преобразования) подход к предобработке текста несколько иной. Модель BERT имеет собственный специальный токенизатор (WordPiece, SentencePiece или BPE), который разбивает текст на подслова и может выпол-

нять множество внутренних операций очистки. Кроме того, поскольку BERT ориентирован на глубокое понимание контекста, некоторые элементы, считающиеся «шумом» (некоторые виды пунктуации, регистр слов и даже некоторые стоп-слова), могут быть полезными семантическими сигналами для модели, которые могут указывать на структуру предложения, эмоциональную окраску или стиль автора. Поэтому обработка текста в BERT обычно минимальна: она может ограничиваться в основном удалением специальных токенов или их заменой спецсимволами, а также корректировкой в соответствии с максимальной длиной входных данных модели. Поскольку в этом проекте используется модель BERT, модуль text_processor.py предназначен для удаления таких элементов, как URL-адреса и имена пользователей, а также для коррекции регистра текста с учётом этих особенностей. Удаление стоп-слов и активное удаление знаков препинания необязательно для BERT, а иногда может даже быть излишним [23].

Напротив, если используется метод Feature Engineering (традиционные модели машинного обучения), то есть традиционные модели машинного обучения (логистическая регрессия, опорные векторные машины, наивный байесовский алгоритм), и их метки создаются вручную (с использованием TF-IDF, Bag-of-Words, n-грамм), то более тщательная и глубокая очистка текста крайне важна. Это связано с тем, что эти модели не способны понимать контекст, как BERT, и очень чувствительны к «шуму». В таком случае все вышеперечисленные этапы очистки (преобразование регистра, удаление URL/имен/знаков препинания/цифр), а также удаление стоп-слов и лемматизация/стемминг, весьма уместны. Эти этапы значительно сокращают размер матрицы признаков, избавляются от избыточных и незначимых признаков, снижают риск переобучения модели и повышают её обобщающую способность.

Таким образом, конвейер предварительной обработки, реализованный в модуле text_processor.py, может быть гибко

адаптирован в зависимости от типа используемой модели и специфики решаемой задачи. В контексте обнаружения мошенничества, особенно при использовании сложных моделей, таких как BERT, цель состоит в том, чтобы удалить основной шум, мешающий работе модели, без потери важной семантической информации.

2.4. Создание и адаптация моделей BERT и Random Forest для классификации мошеннических сообщений

Выявление признаков мошенничества на основе собранных и предварительно обработанных текстовых данных является одной из основных задач системы. Для решения этой задачи необходимо создать и адаптировать эффективные модели машинного обучения, которые могут автоматически классифицировать сообщения с мошенническим содержанием. Для этой цели был разработан модуль threat_classifier.py. Этот модуль использует гибридный подход, объединяющий модель BERT (Bidirection Encoder Representations from Transformers) [24] для извлечения семантических признаков (эмбеддингов) из текстовых данных и классификатор Random Forest [25] для выполнения многометковой классификации на основе полученных признаков. Кроме того, предусмотрены также правила на основе ключевых слов для повышения точности модели и быстрого выявления известных схем мошенничества. В этом разделе подробно описываются причины выбора этих моделей, процессы их создания и адаптации, а также их основные функции.

В последние годы модели, основанные на архитектурах трансформеров, в частности BERT, добились значительного прогресса в области обработки естественного языка (NLP). Главное преимущество модели BERT заключается в её способности глубоко понимать контекстное значение слов посредством двунаправленного анализа. Эти модели способны преобразовывать текст в семантические векторы (эмбеддинги), которые отражают семантическую близость слов и пред-

ложений. В данном проекте для получения эмбеддингов текста была выбрана коинтегрированная/предобученная модель BERT на основе rubert-tiny2. Этот выбор был обусловлен несколькими причинами: языковой адаптацией (модель rubert-tiny2 была обучена на большом корпусе русского текста, а поскольку казахский язык использует кириллицу, эта модель также может давать относительно хорошие результаты для казахских текстов); размером модели и вычислительными ресурсами (префикс tiny указывает на то, что модель относительно небольшая и требует меньше вычислительных ресурсов, чем более крупные модели BERT); и доступностью (модель легко доступна через библиотеку Hugging Face Transformers). В модуле threat_classifier.py модель rubert-tiny2 используется в первую очередь для извлечения плотных векторных представлений (эмбеддингов) из текстовых сообщений. Каждое сообщение проходит через модель BERT, и выходные данные последнего скрытого слоя (обычно вектор, соответствующий токену [CLS]) извлекаются в виде численного эмбеддинга этого сообщения. Эти эмбеддинги затем используются в качестве входных признаков для классификатора случайного леса; этот процесс называется «извлечением признаков».

Для классификации текстовых векторов, полученных с помощью BERT, был выбран алгоритм Random Forest. Random Forest – популярный метод машинного обучения для решения задач классификации и регрессии, основанный на ансамбле нескольких деревьев решений. Его выбор был обусловлен такими факторами, как высокая точность и устойчивость, способность оценивать важность признаков, возможность работы с различными типами данных и масштабируемость [26].

Мошеннические сообщения могут относиться к нескольким категориям одновременно («фишинг» и «кража персональных данных»). Это приводит к задаче многометковой классификации, где каждому сообщению может быть назначена одна или несколько меток. Поскольку Random

Forest напрямую не поддерживает многометковую классификацию, для его адаптации к этой задаче требуется специальная стратегия. В данном проекте использовалась стратегия OneVsRestClassifier (One-vs-the-Rest или One-vs-All). Эта стратегия реализована в библиотеке scikit-learn. Принцип её работы заключается в следующем: если существует N различных меток (категорий), то обучаются N независимых бинарных классификаторов. Каждый классификатор обучается отличать одну метку от остальных. При классификации нового сообщения оно пропускается через все N классификаторов, и каждый классификатор предсказывает вероятность принадлежности сообщения к соответствующей метке. Сообщению присваиваются метки тех классификаторов, которые дают вероятность выше определённого порога. Если каждому образцу назначена только одна метка (но это может быть одна из нескольких выходных меток), можно также рассмотреть стратегию MultiOutputClassifier, которая обучает отдельный классификатор для каждой целевой переменной. Однако в нашем случае, поскольку одно сообщение может относиться к нескольким типам мошенничества, более подходящим является OneVsRestClassifier.

Хотя модели машинного обучения способны выявлять сложные зависимости в данных, иногда могут быть полезны и простые, но эффективные подходы, основанные на правилах. Мошеннические сообщения часто содержат определённые ключевые слова, фразы или шаблоны («бесплатный приз», «ваш аккаунт заблокирован, перейдите по ссылке», «отправьте секретный код»). Для быстрого выявления таких явных признаков мы рассматриваем возможность добавления правил на основе ключевых слов в модуль threat_classifier.py. Это называется гибридным подходом, при котором возможности модели машинного обучения дополняются эвристикой, основанной на знаниях предметной области. Если сообщение соответствует одному из предопределённых ключевых слов «высокого риска», ему автоматически присваивается соответствующая метка мошенничества, или к его прогнозу, по-

лученному с помощью модели машинного обучения, может быть добавлен дополнительный фактор уверенности. Такой подход помогает снизить вероятность ошибок модели и быстро выявлять известные схемы мошенничества.

Во многих распространённых задачах классификации целевые переменные обычно представляются списками или множествами. Модели машинного обучения, особенно в библиотеке scikit-learn, должны представлять целевые переменные в виде двоичной матрицы. Для выполнения этого преобразования используется класс MultiLabelBinarizer из библиотеки scikit-learn. Он принимает список меток и преобразует его в двоичную матрицу. Целевые метки обрабатываются таким образом перед обучением модели, а после выполнения прогнозов исходный список меток может быть получен обратным преобразованием.

Модуль threat_classifier.py содержит ряд важных функций, таких как train (texts, labels), classify(texts), save_model(model_path) и load_model(model_path), которые обеспечивают базовый жизненный цикл работы с моделями. Функция train отвечает за обучение модели классификации, которая принимает вложения из BERT, преобразует метки с помощью MultiLabelBinarizer и обучает модель OneVsRestClassifier (RandomForestClassifier ()). Функция classify предназначена для классификации новых текстов, которая принимает вложения, делает прогнозы с использованием ранее обученной модели, при необходимости применяет правила на основе ключевых слов и преобразует результат в список меток. Функция save_model сохраняет обученную модель, а функция load_model загружает ранее сохранённую модель.

При запуске компонента системы анализа угроз или возникновении новой задачи классификации необходимо загрузить предварительно обученные и сохранённые модели классификации. Этот процесс протоколируется, что позволяет подтвердить корректность загрузки моделей и диагностировать ошибки в случае их возникновения.

Последняя строка свидетельствует о том, что все необходимые модели инициализированы, и модуль анализа угроз

готов к классификации. Такие журналы важны для обеспечения работоспособности системы и проверки корректности настройки её компонентов.

Этот выбор основан на нескольких причинах: языковая адаптация, то есть модель rubert-tiny2 была обучена на большом корпусе русского языка, а поскольку казахский язык использует кириллицу, эта модель также может давать относительно хорошие результаты для казахских текстов; размер модели и вычислительные ресурсы, то есть префикс tiny указывает на то, что модель относительно мала и требует меньше вычислительных ресурсов, чем более крупные модели BERT; и доступность, то есть модель легко доступна через библиотеку Hugging Face Transformers.

Рисунок 2.6 Журнал успешной загрузки моделей классификации модулем анализа угроз

На рисунке 2.6 представлен фрагмент файла журнала, показывающий, что модуль анализа угроз успешно загрузил модели классификации (встраиваемые модели BERT и классификатор Random Forest). Этот фрагмент журнала показывает, что компонент threat_analyzer.classifier успешно загрузил модель BERT, классификатор Random Forest и объект MultiLabelBinarizer по указанным путям. Последняя строка указывает на то, что все необходимые модели инициализированы, и модуль анализа угроз готов к выполнению классификации. Такие журналы важны для

обеспечения работоспособности системы и проверки корректности настройки её компонентов.

Фрагмент лог-файла, представленный на рисунке 2.6, демонстрирует готовность модуля анализа угроз, одного из основных компонентов системы автоматического обнаружения мошеннических действий в мессенджере Telegram. Этот модуль отвечает за анализ данных, и его основная задача – выявление признаков мошенничества на основе сообщений, полученных из каналов или чатов Telegram. Система использует сложные методы машинного обучения для распознавания мошенничества, в частности, эмбеддинги BERT и классификатор Random Forest.

В журнале сначала показан процесс загрузки нескольких важных файлов модели с помощью модуля threat_analyzer.classifier. Первым загружаемым объектом является модель встраивания BERT. Эта модель преобразует текст сообщений в векторное пространство, то есть представляет собой числовое представление, соответствующее их семантическому значению. Эти числовые представления сохраняют важные признаки, определяющие характер мошенничества, поэтому они являются очень важными входными данными для классификатора. Модель BERT была предварительно обучена на большом текстовом корпусе, а затем адаптирована к конкретным сообщениям Telegram.

Следующий компонент для загрузки – классификатор Random Forest. Этот ансамблевый алгоритм работает на основе множества деревьев решений и объединяет прогнозы различных деревьев для обеспечения максимально достоверного результата. Метод Random Forest показывает хорошие результаты при работе с текстовыми данными и эффективно работает в сочетании с встраиванием BERT. Классификатор позволяет классифицировать виды мошенничества по одной или нескольким категориям, поскольку сообщения Telegram часто содержат сразу несколько сценариев (например, инвестиционные мошенничества и фейковые боты).

Третий важный загружаемый объект – MultiLabelBinarizer. Этот компонент предназначен для обработки меток целей в задачах классификации по нескольким меткам. Поскольку сообщение Telegram может относиться к нескольким категориям мошенничества одновременно, с помощью MultiLabelBinarizer система может классифицировать одно сообщение по нескольким классам. Например, одно сообщение может содержать несколько элементов, таких как фейковый розыгрыш, фишинг и имитация знаменитостей.

Последняя строка журнала содержит сообщение «Все модели успешно загружены. Классификатор готов». Это означает, что все модели в системе успешно загружены и настроены корректно. Такие сообщения журнала играют важную роль при тестировании и мониторинге системы в производственной среде. Если одна из моделей не загружается или пути к файлам указаны неверно, этот журнал можно использовать для быстрого выявления проблемы и принятия необходимых мер. Таким образом, файл журнала является своего рода «зеркалом» внутренних процессов системы, с помощью которого разработчик или администратор может оценить её производительность.

Модульная структура и контроль журналов повышают надёжность системы. Особенно для систем, работающих в сфере кибербезопасности, проверка и мониторинг конкретной функциональности каждого модуля является залогом успешной работы. Кроме того, тщательная организация журналов служит важной базой для будущих улучшений системы. Например, при добавлении новой модели или подхода можно оценить производительность, найти ошибки и оптимизировать процесс реконфигурации, сравнивая его с предыдущими журналами.

В целом, фрагмент журнала, представленный на рисунке 2.6, наглядно демонстрирует подготовку разрабатываемой интеллектуальной системы к началу процесса классификации. Порядок загрузки моделей, их совместимость друг с другом и готовность к работе обеспечивают эффективность и

надёжность системы. Этот этап важен не только с технической, но и с исследовательской точки зрения, поскольку точная и точная классификация является основой всей системы. Таким образом, корректная инициализация классификатора и положительный результат журнала являются предпосылками успешного запуска интеллектуальной системы.

В заключение отметим, что комбинация моделей BERT и Random Forest, реализованная в модуле threat_classifier.py, дополненная правилами на основе ключевых слов, представляет собой мощный и гибкий инструмент для классификации многих известных мошеннических сообщений. Наличие функций обучения, классификации, сохранения и загрузки моделей обеспечивает стабильную работу системы и простоту обновления. Для реализации данного преобразования используется класс MultiLabelBinarizer из библиотеки scikit-learn. Он принимает список меток и преобразует их в бинарную матрицу. Перед обучением модели целевые метки обрабатываются таким образом, а после выполнения прогноза исходный список меток может быть получен обратным преобразованием.

2.5. Интеграция модуля OSINT для деанонимизации пользователей

Помимо обнаружения мошеннической деятельности, деанонимизация подозрительных учётных записей или лиц является одной из важнейших задач кибербезопасности [27]. Этот процесс направлен на снижение анонимности мошенников в интернете, выявление их активности на других платформах и содействие расследованиям правоохранительных органов. Для этого в систему интегрирован специальный модуль OSINT (Open-Source Intelligence) под названием richment_service. Этот модуль способствует процессу деанонимизации, собирая дополнительную информацию из открытых источников о высокорисковых организациях, выявленных threat_analyzer. В этом разделе обсуждается назначение модуля richment_service, используемые методы и инструменты

OSINT, а также процесс сбора и обогащения данных основного хранилища данных системы.

Основная цель модуля enrichment_service – расширить цифровой след подозрительных пользователей, идентифицированных на платформе Telegram (по имени пользователя или идентификатору пользователя), путём сбора дополнительной информации из открытых источников и, по возможности, связывания их с реальными людьми или другими онлайн-идентификаторами. Этот процесс называется «обогащением», поскольку он дополняет ограниченные данные, изначально полученные из Telegram, внешней информацией. В число задач модуля входит поиск профилей в других социальных сетях, сбор информации через поисковые системы, использование специализированных инструментов и баз данных OSINT, а также выявление связанных аккаунтов или групп. Поскольку имена пользователей Telegram часто дублируются на других платформах социальных сетей (Instagram, Twitter/X, Facebook, ВКонтакте, TikTok), richment_service ищет соответствующие профили на этих платформах, используя это имя пользователя, и может получать дополнительную информацию (имя, фотографии, контактные данные, публикации) из найденных профилей. Поиск имени пользователя, идентификатора пользователя (user_id) или других уникальных идентификаторов, извлеченных из сообщений, с помощью поисковых систем, таких как Google, Яндекс, DuckDuckGo, позволяет находить сообщения на форумах, комментарии в блогах, комментарии в новостях или связанную информацию на других веб-ресурсах. Также используются специализированные инструменты OSINT и базы данных, предназначенные для поиска определённых типов информации (доменных имён, IP-адресов, записей во взломанных базах данных). Если у подозрительного пользователя обнаружены тесные связи с другими аккаунтами или группами Telegram, эта информация может помочь раскрыть мошенническую сеть. Вся собранная дополнительная информация систематически хранится и связывается с исходным подозреваемым, что впоследствии

помогает аналитикам лучше понять масштабы, участников и методы мошенничества, а также сформировать доказательную базу для правоохранительных органов.

Были рассмотрены различные методы и инструменты OSINT, некоторые из которых частично использовались для реализации модуля enrichment_service [28]. Их выбор зависел от доступности, возможностей автоматизации, правовой базы и релевантности собираемой информации. Основные используемые методы: перечисление имен пользователей, обратный поиск по изображениям, проверка и профилирование адресов электронной почты, анализ номеров телефонов, анализ веб-сайтов и инструменты анализа социальных сетей. При перечислении имен пользователей пользователь проверяет, зарегистрировано ли конкретное имя пользователя на различных платформах, используя такие инструменты, как WhatsMyName.app, Sherlock, Maigret или методы Google Dorking. С помощью обратного поиска по изображениям можно найти изображение профиля подозрительной учетной записи в таких сервисах, как Google Images, TinEye, Яндекс.Картинки, и определить, использовалось ли оно где-либо еще. Если адрес электронной почты найден, проверяется его действительность и выполняется поиск связанных онлайн-профилей (через Hunter.io, Gravatar). Если известен номер телефона, можно определить его страну и оператора, а также выполнить поиск связанных аккаунтов в социальных сетях, но эти методы могут быть ограничены из-за законов о конфиденциальности. Если фишинговое сообщение содержит ссылку на веб-сайт, собирается дополнительная информация, такая как данные WHOIS, IP-адрес сайта и хостинг-провайдер. Такие инструменты, как Maltego и библиотеки анализа социальных сетей (SNA), могут помочь визуализировать и анализировать связи в социальных сетях, но они часто полагаются на ограничения API или платные подписки. Модуль enrichment_service изначально опирается в основном на методы проверки имени пользователя, которые легко автоматизировать и широко используются, а также на методы поиска, такие как Google Dorking. Важным аспектом

является обеспечение того, чтобы все мероприятия OSINT проводились в рамках закона и в соответствии с этическими нормами, то есть должна собираться только общедоступная и законно доступная информация [29].

Модуль enrichment_service тесно взаимодействует с компонентом threat_analyzer. Процесс анализа начинается с получения триггера: когда модуль threat_analyzer классифицирует определённое сообщение или пользователя как представляющее высокий риск, он отправляет соответствующий триггер или задание в enrichment_service, который обычно содержит идентификаторы, такие как user_id и/или имя пользователя подозрительного пользователя. Затем, получив триггер, richment_service начинает поиск дополнительной информации в открытых источниках по этим идентификаторам, используя описанные выше методы OSINT; этот процесс может выполняться асинхронно в фоновом режиме. Вся релевантная информация, найденная в ходе сбора (ссылки на профили в других социальных сетях, связанные электронные письма, обсуждения на форумах, IP-адреса, домены), преобразуется в структурированный формат, каждый из которых может быть дополнен метаданными, такими как источник найденной информации, время сбора и уровень достоверности. В результате структурированные данные OSINT добавляются в соответствующую запись в коллекции detected_threats в базе данных MongoDB, которая хранит основную информацию о каждой угрозе, обнаруженной threat_analyzer, а enrichment_service дополняет эти записи дополнительными полями или добавляет новые связанные данные. Наконец, обогащённые данные удобно отображаются пользователям или аналитикам через системную панель мониторинга, позволяя им видеть полный профиль подозреваемого и анализировать связи мошеннической схемы. Работа модуля enrichment_service может представлять собой непрерывный процесс: по мере появления новых инструментов и методов OSINT или изменения цифрового следа подозреваемого процесс обогащения может повторяться или обновляться. Этот модуль направлен на повышение общей эффективности

системы, учитывая динамический, а не статичный характер борьбы с мошенничеством. Важно понимать, что деанонимизация – сложный и не всегда успешный процесс, однако использование методов OSINT затрудняет сокрытие мошенников и позволяет получить больше информации об их действиях.

2.6. Система управления данными (MongoDB) и панель визуализации результатов (Dashboard)

Для эффективного функционирования системы обнаружения мошенничества и деанонимизации важно обеспечить надежное хранение собранных данных, управление ими и удобное представление результатов анализа. Для решения этих задач система интегрирует NoSQL-базу данных MongoDB и веб-панель визуализации (Dashboard). В данном разделе подробно рассматриваются причины выбора базы данных MongoDB, базовая структура данных, а также основные функции Dashboard и использование фреймворка Flask для ее реализации.

В качестве хранилища данных для системы была выбрана система управления базами данных (СУБД) MongoDB NoSQL [30]. Этот выбор был сделан из-за нескольких ключевых преимуществ, таких как гибкая схема, масштабируемость, высокая производительность, удобство документной модели, а также наличие широкой поддержки и сообщества. MongoDB является документоориентированной СУБД, и данные хранятся в виде документов в формате BSON (Binary JSON), что обеспечивает гораздо более гибкую схему, чем традиционные реляционные СУБД. Благодаря разнообразной структуре сообщений Telegram, MongoDB позволяет легко хранить такие данные в единой коллекции. Кроме того, MongoDB поддерживает горизонтальное масштабирование (через шардинг) при увеличении объема данных и нагрузки запросов и может демонстрировать высокую производительность для операций записи и чтения. JSON-подобная структура документов естественным образом интегрируется со многими языками программирования.

Для хранения и управления данными система использует NoSQL базу данных MongoDB, что обеспечивает эффективное хранение и быстрый поиск сообщений с переменной структурой. Для удобного представления результатов разработана специальная панель визуализации (Dashboard), которая в режиме реального времени отображает статистику сообщений, обнаруженные угрозы и информацию о пользователях.

Основные данные в системе хранятся в двух основных коллекциях MongoDB: raw_messages и detected_threats. Коллекция raw_messages предназначена для хранения всех необработанных сообщений, собранных из Telegram модулем Telegram-collector. Каждый документ соответствует одному сообщению и содержит все его атрибуты (message_id, chat_id, text, sender_id, timestamp, media, entity и т. д., как описано в разделе 2.2). Эта коллекция является основным хранилищем данных системы. На рисунке 2.7 показаны несколько документов из коллекции raw_messages, созданных с помощью MongoDB Compass (или другого клиента MongoDB). Видно, что каждый документ может иметь свой собственный набор полей, но при этом имеет базовые идентификаторы и элементы содержимого.

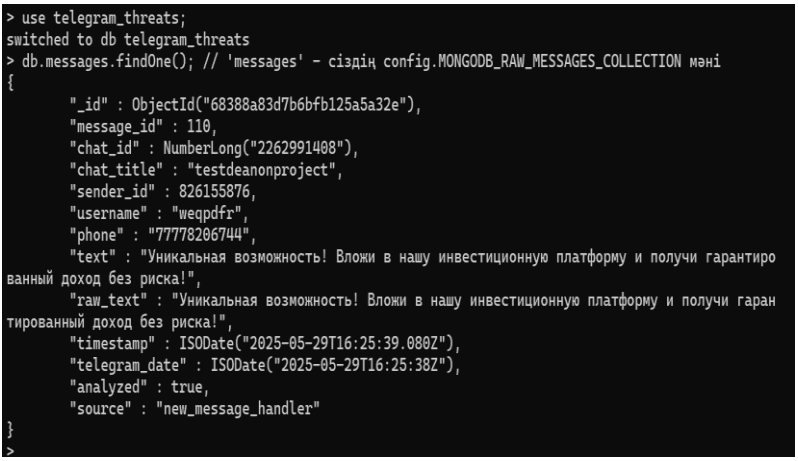

Рисунок 2.7 Структура коллекции raw_messages в MongoDB

На рисунке 2.7 показаны несколько документов из коллекции raw_messages, созданных с помощью MongoDB Compass (или другого клиента MongoDB). Видно, что каждый документ может иметь свой собственный набор полей, но при этом у них общие идентификаторы и элементы контента. Один документ может содержать поля текста и сущностей, а другой – поле медиа (с информацией о фотографии или видео).

Коллекция detected_threats предназначена для хранения информации о сообщениях и связанных с ними сущностях, которые были идентифицированы как мошеннические или классифицированы как высокорисковые модулем threat_classifier. Каждый документ соответствует одной обнаруженной угрозе (инциденту) и может содержать такую информацию, как номер исходного сообщения (raw_message_id), список обнаруженных типов мошенничества (fraud_types), оценка риска (risk_score), временная метка обнаружения (detection_timestamp), идентификаторы подозрительных сущностей (suspect_sender_id, suspect_username), данные OSINT, собранные сервисом enrichment_service (osint_findings), статус инцидента (status) и комментарии аналитика (analyst_comments). На рисунке 2.8 показано несколько документов из коллекции detected_threats. Эти документы предоставляют сводную информацию об инцидентах мошенничества, в которой объединены источник, характер обнаруженной угрозы, подозрительная сущность и дополнительная информация OSINT, собранная для ее обогащения. Помимо этих двух основных коллекций, система также может иметь дополнительные коллекции для хранения параметров конфигурации, ролей пользователей или метаданных модели.

Для эффективного представления собранных и проанализированных данных пользователю была разработана специальная панель мониторинга. Доступ к ней осуществляется через веб-интерфейс. Она предназначена для визуализации ключевых показателей производительности системы, выявленных угроз и сопутствующей информации.

Для реализации панели мониторинга и подключения её к базе данных MongoDB был использован микрофреймворк Flask Python. Выбор Flask был обусловлен его лёгкостью и гибкостью, лёгкой интеграцией с экосистемой Python, простотой создания RESTful API и возможностью работы с мощными шаблонизаторами, такими как Jinja2.

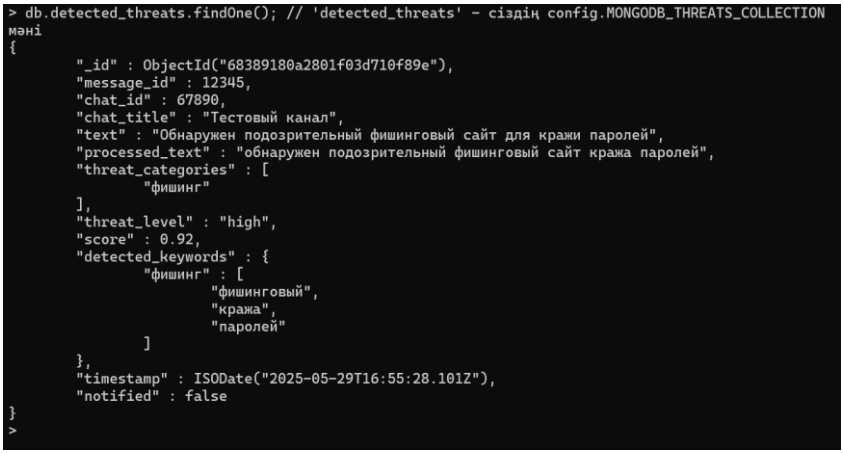

Рисунок 2.8 Структура коллекции detected_threats в MongoDB

На рисунке 2.8 показаны несколько документов из коллекции detected_threats. Эти документы предоставляют сводную информацию об инцидентах мошенничества, объединяя исходный источник, характер обнаруженной угрозы, подозреваемого и дополнительную информацию OSINT, собранную для её обогащения. Такая структура позволяет аналитикам изучать каждый инцидент индивидуально и выявлять связи между ними. Помимо этих двух основных коллекций, система может также иметь дополнительные коллекции для хранения параметров конфигурации, ролей пользователей или метаданных модели.

Для эффективного представления собранных и проанализированных данных пользователю разработана специальная панель мониторинга (Dashboard). Доступ к ней осуществляется через веб-интерфейс. Она предназначена для

визуализации ключевых показателей производительности системы, выявленных угроз и сопутствующей информации. Основные функции панели мониторинга включают в себя отображение статистики (общее количество собранных сообщений, общее количество обнаруженных случаев мошенничества, разбивка по различным типам мошенничества, тенденции активности мошенничества с течением времени, наиболее активные подозрительные учетные записи или группы, эффективность процесса обогащения OSINT), отображение списка инцидентов (табличный список всех обнаруженных угроз в коллекции detected_threats, основная информация по каждому инциденту, возможность сортировки списка по различным критериям), фильтрацию (фильтрация списка инцидентов и статистики по определенным критериям, по определенному интервалу времени, определенному типу мошенничества, предполагаемому sender_id или имени пользователя, статусу инцидента, уровню угрозы) и подробного представления (подробная информация о каждом инциденте, выбранном из списка, полный текст исходного сообщения о мошенничестве, признаки, определенные threat_classifier, данные OSINT, собранные enrichment_service, интерфейс для аналитика для изменения статуса инцидента, добавления комментариев). Эти функции помогают аналитикам быстро оценивать ситуацию с мошенничеством, фокусироваться на значимых инцидентах и принимать соответствующие меры [31].

Для реализации панели мониторинга и её подключения к базе данных MongoDB использовался микрофреймворк Flask Python. Flask был выбран из-за его лёгкости и гибкости, лёгкой интеграции с экосистемой Python, простоты создания RESTful API и возможности работы с мощными шаблонизаторами, такими как Jinja2 [32]. Приложение Flask содержит логику для извлечения данных из MongoDB, их обработки и отправки на HTML-страницы панели мониторинга. Пользовательский интерфейс может быть реализован с использованием HTML, CSS и JavaScript,

взаимодействующих с API Flask посредством AJAX-запросов, что обеспечивает интерактивность и отзывчивость пользовательского интерфейса.

В заключение отметим, что, хотя MongoDB обеспечивает гибкость и масштабируемость как система управления данными, панель мониторинга на базе Flask представляет собой удобный инструмент для эффективной визуализации результатов анализа и управления инцидентами мошенничества. Эти два компонента значительно расширяют возможности системы по хранению, обработке и интерпретации данных.

2.7. Алгоритм работы разработанной системы и ее развертывание в Docker-контейнере

Обеспечение стабильности, надежности и простоты развертывания разрабатываемой системы обнаружения мошенничества и деанонимизации является важным аспектом, повышающим её практическую ценность. В данном разделе описывается общий алгоритм работы системы, то есть полная цепочка событий от момента получения сообщения от Telegram до его обработки, анализа и отображения результатов на панели управления. Кроме того, подробно рассматриваются технологии изоляции всех компонентов системы в Docker-контейнерах и управления ими с помощью Docker Compose, их преимущества, содержание конфигурационных файлов и процесс запуска.

Алгоритм разработанной системы представляет собой конвейерный процесс, состоящий из нескольких взаимосвязанных этапов [33]. Во-первых, на этапе сбора данных (telegram-collector), процесс начинается с поступления нового сообщения из мессенджера Telegram. Модуль telegram-collector (описанный в разделе 2.2) использует библиотеку Telethon для непрерывного прослушивания чатов Telegram или пользователей, указанных в конфигурации. При обнаружении нового сообщения telegram-collector собирает все его атрибуты и

сохраняет их в коллекции raw_messages базы данных MongoDB в виде структурированного документа BSON. Во-вторых, на этапе предобработки текста (через text_processor.py, внутри threat-analyzer), модуль threat-analyzer считывает вновь полученные или необработанные сообщения из коллекции raw_messages. Текстовое содержимое каждого сообщения передается через конвейер text_processor.py (описанный в разделе 2.3), где текст очищается и подготавливается для моделей машинного обучения. В-третьих, на этапе классификации мошенничества (с использованием threat_classifier.py внутри threat-analyzer) предварительно обработанные тексты отправляются в модуль threat_classifier.py (описанный в разделе 2.4). Сначала тексты проходят через модель BERT (cointegrated/rubert-tiny2) для получения их числовых векторов, а затем эти векторы передаются в классификатор Random Forest для прогнозирования принадлежности сообщения к одной или нескольким категориям мошенничества; на этом этапе также могут использоваться правила на основе ключевых слов. Результаты классификации сохраняются в виде нового документа в коллекции detected_threats базы данных MongoDB. В-четвертых, на этапе обогащения данных и деанонимизации (enrichment_service), если анализатор угроз обнаруживает, что сообщение представляет высокий риск, он отправляет триггер в модуль enrichment_service (описанный в разделе 2.5). Сервис Retirement_service ищет дополнительную информацию из открытых источников (OSINT) по идентификаторам подозрительного пользователя, и найденная информация обогащает соответствующую запись об инциденте в коллекции detected_threats. В-пятых, на этапе оповещения (уведомления) при выполнении определённых условий в конфигурации системы срабатывает модуль уведомления, который отправляет сообщение ответственным лицам. В-шестых, на этапе визуализации результатов (через Dashboard, приложение) пользователь получает доступ к Dashboard (описано в разделе 2.6) через веб-браузер.

Серверная часть Dashboard (модуль приложения на базе Flask) запрашивает данные из MongoDB, обрабатывает их и отправляет клиентской части Dashboard в виде статистических показателей, списка инцидентов и подробных представлений, где пользователь может фильтровать инциденты, изменять их статус и просматривать подробную информацию. Данный алгоритм охватывает полный цикл: от выявления мошеннических сообщений до предоставления пользователю полной информации о них, обеспечивая слаженную работу всех компонентов системы.

Для развертывания и управления всеми системными службами (сборщиком, анализатором угроз, приложением, уведомителем, службой обогащения, MongoDB) [34] использовались платформа контейнеров Docker и инструмент Docker Compose [34]. Использование этих технологий обеспечивает ряд важных преимуществ, таких как изоляция, переносимость, согласованность конфигурации, простота масштабирования, эффективное использование ресурсов, быстрое развертывание и обновление, а также поддержка микросервисной архитектуры. Каждая служба изолирована в своем собственном контейнере Docker вместе со своими зависимостями, что обеспечивает единообразную работу в различных средах. Контейнеры Docker можно запускать без изменений на любой системе с установленным Docker, а вся системная среда определяется как код с помощью файлов Dockerfile и docker-compose.yml. Используя Docker Compose, количество экземпляров отдельных служб можно легко увеличивать или уменьшать. Контейнеры легче виртуальных машин и используют общее ядро операционной системы хоста. Процесс создания образов Docker и запуска контейнеров автоматизирован и быстр, а обновления легко выполняются путем создания нового образа и замены старого контейнера новым.

Для каждой службы (telegram-collector, threat-analyzer, app, notifier, richment_service) создаётся отдельный Dockerfile, содержащий инструкции по сборке образа Docker

для этой службы. Обычно Dockerfile состоит из следующих основных этапов: FROM – определение базового образа, WORKDIR – указание рабочего каталога внутри контейнера, COPY или ADD – копирование необходимых файлов проекта, RUN – определение команд, выполняемых при сборке образа контейнера, EXPOSE (необязательно) – объявление порта, который контейнер прослушивает в сети, ENV (необязательно) – установка переменных окружения, используемых внутри контейнера, и CMD или ENTRYPOINT – определение основной команды, выполняемой при запуске контейнера.

Файл docker-compose.yml используется для определения, настройки и управления приложением, состоящим из нескольких контейнеров Docker. Он написан на языке YAML и описывает все службы в системе, их образы, порты, тома, сетевые подключения и зависимости [35]. Основные элементы конфигурации:

– Версия – версия файла Docker Compose.

– Службы – раздел, определяющий каждую службу (контейнер) системы. Каждой службе присваивается имя (коллектор, анализатор, mongodb, webapp).

– Сборка – если службе необходимо создать образ Docker с использованием Dockerfile, указывает каталог, в котором находится Dockerfile.

– Образ. Если вы хотите использовать готовый образ Docker (mongo:latest из Docker Hub), укажите имя образа.

– Порты – привязывает внутренний порт контейнера к порту хост-системы (для службы веб-приложения «5000:5000»).

– Тома – определяют тома для постоянного хранения данных или для обмена файлами с хост-системой (./mongo-data:/data/db для хранения данных MongoDB).

– Окружение – задаёт переменные окружения для службы. Эти переменные можно получить из файла .env.

– depend_on – определяет порядок запуска служб (служба webapp должна запускаться после запуска службы mongodb).

– Сети – определяет частную сеть для взаимодействия служб друг с другом.

– Перезапуск – определяет политику перезапуска контейнера, если он остановлен (всегда или в случае сбоя).

Файл .env (environment) – это простой текстовый файл, используемый для хранения переменных окружения, используемых в конфигурациях Docker Compose и самих приложениях. Его основная роль – хранить конфиденциальные или специфичные для среды параметры конфигурации (ключи API, логины/пароли базы данных, порты, имена хостов) отдельно от основного кода и файла docker-compose.yml, что повышает безопасность и упрощает управление различными конфигурациями для разных сред. В файле docker-compose.yml переменные окружения извлекаются из файла .env с помощью синтаксиса ${VARIABLE_NAME}. На рисунке 2.9 показано содержимое файла конфигурации проекта .env, в котором определены ключи API Telegram, логины/пароли MongoDB, порт веб-приложения и другие важные параметры. Убедитесь, что файл .env не попал в репозиторий, добавив его в файл gitignore.

```
# Общие настройки
MONGODB_URI=mongodb://mongodb:27017/
MONGODB_DB=telegram_threats
MONGODB_COLLECTION=detected_threats

# Настройки Telegram (сборщик)
TELEGRAM_API_ID=21730420
TELEGRAM_API_HASH=ea9d67e635316e8394b995de0824e730
TELEGRAM_PHONE=+77778206744
MONITORED_CHANNELS=@testforprojectdeanon

# Настройки Telegram (оповещения)
TELEGRAM_BOT_TOKEN=7420375192:AAHr_X9-JU1P7rHiJ6XLB9bO3AOE0b0hTfM
TELEGRAM_NOTIFICATION_CHAT_ID=7420375192

# Настройки веб-интерфейса
FLASK_DEBUG=False
PORT=5000
```

Рисунок 2.9 Файл конфигурации проекта .env

На рисунке 2.9 показано содержимое конфигурационного файла .env проекта. Секретные данные (API_HASH, DB_PASSWORD) заменены звёздочками или общими значениями.

Этот файл определяет ключи API Telegram, логины и пароли MongoDB, порт веб-приложения и другие важные параметры. Убедитесь, что файл .env не включён в репозиторий, добавив его в файл .gitignore.

Этот файл .env позволяет централизованно управлять конфигурацией системы, позволяя изменять все настройки в одном месте. Такой подход упрощает работу системы в различных средах (локальной, тестовой, производственной). Кроме того, разделение конфиденциальных данных и кода отвечает требованиям безопасности. При запуске системы эти переменные автоматически загружаются и передаются в соответствующие модули. Такой подход к настройке полностью совместим с процессами DevOps и CI/CD, что упрощает реализацию автоматизированных развертываний.

На рисунке 2.9 показаны основные настройки системы, необходимые для сбора данных с платформы Telegram, их обработки и хранения результатов. Эти настройки представляют собой начальную конфигурацию, необходимую для работы с базой данных MongoDB, API Telegram и веб-интерфейсом. Функциональная роль и техническое значение каждого блока конфигурации в системе подробно обсуждаются ниже.

Сначала определяются общие параметры. Здесь указывается ссылка для подключения к серверу MongoDB с помощью MONGODB_URI=mongodb://mongodb:27017/. В данном случае сервер MongoDB доступен в локальной среде контейнера на хосте mongodb через стандартный порт 27017. Эта ссылка обеспечивает прямое подключение к основной базе данных системы. Кроме того, строка MONGODB_DB=telegram_threats указывает, с какой именно базой данных нужно работать. В этом случае все опасные или подозрительные сообщения и данные о пользователях,

собранные из Telegram, хранятся в базе данных telegram_threats. А параметр MONGODB_COLLECTION=detected_threats указывает конкретную коллекцию (или таблицу) в базе данных, где будут храниться данные. Имя коллекции – detected_threats, что означает, что в этой коллекции собираются все угрозы, обнаруженные в Telegram в этой системе, включая фишинг, мошенничество и другой вредоносный контент.

Следующий блок посвящен параметрам коллектора, необходимым для подключения к платформе Telegram. Здесь указаны данные авторизации, необходимые для подключения к официальным API Telegram, с помощью TELEGRAM_API_HASH=e49d67e653********** и TELEGRAM_API_ID=21**********. Эти данные берутся с платформы Telegram https://my.telegram.org и предоставляются индивидуально для каждой системы или исследователя. API ID и API HASH – основные данные аутентификации, позволяющие установить защищенное соединение с сервером Telegram.

Кроме того, указан номер телефона, зарегистрированный в Telegram: TELEGRAM_PHONE=+77778296744. Этот номер является частью учётной записи, которую система использует для связи с сервером Telegram. На этот номер отправляются одноразовые коды подтверждения и выполняется первичная авторизация системы. Параметр MONITORED_CHANNELS=@testforprojectdeanon указывает, какие каналы или группы Telegram отслеживает система. В данном примере отслеживаемый канал указан как @testforprojectdeanon, что означает, что система собирает и анализирует все сообщения в этом канале, а при необходимости автоматически выявляет подозрительные сообщения.

Третий блок описывает параметры, необходимые для отправки оповещений и сообщений через Telegram. Это раздел для отправки сообщений через бот в Telegram. Параметр TELEGRAM_BOT_TOKEN=7420375192: AAHr_X9-JU1P7nH6JXLB9b03AOE0b0hTfM – это токен, специально

назначенный боту Telegram, то есть система использует этот токен для подключения бота Telegram и отправки подготовленных сообщений в определённый канал или чат. Параметр TELEGRAM_NOTIFICATION_CHAT_ID=7420375192 указывает идентификатор чата, из которого отправляются сообщения. Это позволяет системе автоматически пересылать все обнаруженные угрозы или сообщения о произошедших событиях в указанный чат. Такой подход обеспечивает работу системы в режиме реального времени и позволяет создать механизм мгновенного оповещения через Telegram.

В последнем блоке показаны основные параметры веб-интерфейса. Параметр FLASK_DEBUG=False отключает режим отладки веб-интерфейса, разработанного с использованием фреймворка Flask. Это необходимо из соображений безопасности в производственной среде, поскольку при включении режима отладки может быть раскрыта дополнительная системная информация. Параметр PORT=5000 указывает порт, на котором будет работать веб-интерфейс. В данном случае выбран порт 5000, что означает, что приложение Flask будет получать и обрабатывать HTTP-запросы через этот порт. Этот веб-интерфейс предоставляет возможность визуализации системных результатов, просмотра данных и управления системными операциями.

В этом конфигурационном файле определяются необходимые точки подключения для гармоничной работы всех основных компонентов системы. Во-первых, указывается сервер MongoDB для хранения данных. Во-вторых, задаются параметры, необходимые для сбора данных через API Telegram и отправки оповещений через бот Telegram. В-третьих, задаются настройки сервера Flask, необходимые для работы веб-интерфейса, входящего в систему.

Такая конфигурация обеспечивает модульную и гибкую структуру системы. Изменяя эти параметры, систему можно легко адаптировать к новым базам данных, другим каналам Telegram или другим ботам. Такая структура также эффективна для удобного использования с контейнерными техно-

логиями, например, в среде Docker. Каждый параметр можно настроить для автоматической загрузки через файл. env, превращая систему в полноценную микросервисную архитектуру.

Кроме того, данная конфигурация открывает возможность интеграции системы не только с Telegram, но и с другими мессенджерами. Например, для работы с платформами WhatsApp, Viber или Discord достаточно изменить соответствующие настройки API. Такие системы являются важным инструментом для создания автоматизированных, быстродействующих платформ, отвечающих современным требованиям кибербезопасности. Это позволяет улучшить процесс раннего обнаружения киберугроз и оперативного реагирования на них.

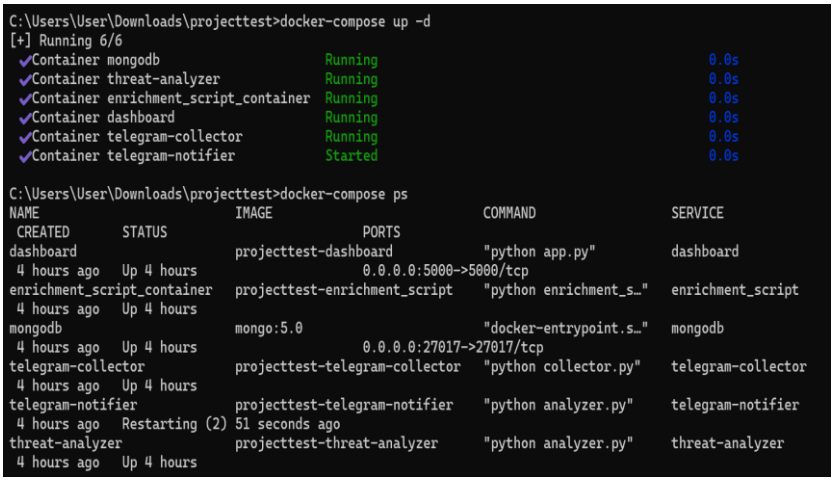

Рисунок 2.10 Журнал успешного запуска Docker-контейнеров

На рисунке 2.10 показан процесс запуска сложной программной системы, предназначенной для обнаружения и анализа мошенничества в мессенджере Telegram, в среде Docker. Здесь все основные модули системы организованы в микросервисную архитектуру, изолированы контейнерами Docker и работают независимо друг от друга. Такой подход повышает гибкость, масштабируемость и надежность систе-

мы. Кроме того, использование Docker позволяет быстро разворачивать и управлять каждым компонентом.

В верхней части изображения показана команда docker-compose up -d. Эта команда позволяет запустить все системные службы одновременно в фоновом режиме (параметр -d указывает на режим отсоединения). В результате выполнения команды было успешно запущено шесть контейнеров: mongodb, threat-analyzer, richment_script_container, dashboard, telegram-collector и telegram-notifier. Все контейнеры были запущены немедленно, и их статус отмечен как «Running» или «Started». Это означает, что все системные модули настроены правильно и успешно работают.

В следующем разделе показана команда docker-compose ps. Эта команда отображает текущее состояние всех запущенных контейнеров, связанных с ними образов, открытых портов, запущенных команд и имен служб. Давайте рассмотрим каждый контейнер отдельно:

Первый контейнер – это панель мониторинга. Он предназначен для визуализации результатов работы системы через веб-интерфейс. Контейнер основан на образе projecttest-dashboard и запускается командой python app.py. Открытый порт – 5000, что означает, что пользователь может получить доступ к интерфейсу системы, перейдя в браузере по адресу localhost:5000. Контейнер работает непрерывно уже четыре часа.

Второй контейнер – enrichment_script_container. Он создан на основе образа projecttest-enrichment_script и выполняет команду python richment_script.py. Этот модуль выполняет обогащение данных на основе методов OSINT. Он ищет дополнительную информацию о пользователях Telegram в открытых источниках, расширяя аналитические возможности системы. Контейнер непрерывно работает уже четыре часа.

Третий контейнер – mongodb. Он служит системной базой данных. Он основан на образе MongoDB (mongo:5.0) и работает на стандартном порту 27017. Этот контейнер используется для хранения сообщений, собранных из Telegram,

пользовательских данных и обнаруженных случаев мошенничества. Время работы контейнера составляет четыре часа.

Четвёртый контейнер – telegram-collector. Этот контейнер запускается с помощью образа projecttest-telegram-collector и выполняет скрипт python collecter.py. Его основная задача – сбор сообщений из каналов Telegram и занесение их в базу данных системы. Этот компонент является первым этапом сбора данных в системе. Контейнер также работает непрерывно уже четыре часа.

Пятый контейнер – telegram-notifier. Этот контейнер предназначен для отправки уведомлений через Telegram-бота. Образ контейнера – projecttest-telegram-notifier, а команда для выполнения – python notifier.py. Текущее состояние контейнера – «Перезапуск», что означает, что он был перезапущен несколько раз. Обычно такая ситуация возникает из-за временных ошибок в конфигурации или скрипте. Однако, поскольку системные Docker-окружения выполняют функцию самоперезагрузки, такие незначительные ошибки не приводят к полному завершению работы.

Шестой контейнер – threat-analyzer. Образ этого контейнера – projecttest-threat-analyzer, а исполняемая команда – python analyzer.py. Контейнер анализирует сообщения, собранные в Telegram, и оценивает вероятность мошенничества, используя модели BERT и Random Forest. Это основной аналитический и принимающий решения блок-системы. Контейнер непрерывно работает уже четыре часа.

Все модули системы тесно взаимосвязаны и выполняют определённые функциональные нагрузки. MongoDB, как центр хранения данных, является единой базой для всех компонентов. Telegram-collector собирает необходимые сообщения из Telegram, richment_script добавляет дополнительные OSINT-данные, threat-analyzer анализирует собранные сообщения, а telegram-notifier отправляет уведомления об обнаруженных угрозах через Telegram-бота. Компонент Dashboard отображает результаты всех процессов в удобном для пользователя виде.

Главным преимуществом данного архитектурного решения является модульная и легко управляемая структура системы. Поскольку каждый компонент работает в независимом контейнере, при выходе из строя одного модуля остальная система продолжает работать. Кроме того, упрощаются процессы обновления, тестирования и перенастройки контейнеров. Такой подход также облегчает развертывание системы на разных серверах или в облаке.

Процесс запуска и мониторинга системы осуществляется быстро и эффективно благодаря гибкости и простоте Docker-среды. Поскольку конкретная функция и ответственность каждого контейнера определены, при дальнейшем улучшении системы или внедрении нового функционала достаточно вносить изменения только в конкретный контейнер. Такой подход позволяет масштабировать систему в будущем и интегрировать её с различными мессенджерами.

В целом, рабочий процесс, представленный на рисунке 2.10, демонстрирует, что система Telegram для автоматического сбора, обработки и обнаружения мошеннических сообщений полностью работоспособна и функционирует корректно. Система готова к использованию в качестве эффективного и надежного инструмента, отвечающего современным требованиям кибербезопасности.

Для запуска системы выполняется команда docker-compose up. Эта команда создаёт (при необходимости) и запускает все службы, определённые в файле docker-compose.yml. Процесс запуска отслеживается с помощью журналов. На рисунке 2.10 показан фрагмент журнала после успешного выполнения команды docker-compose up или вывод команды docker-compose ps. Этот журнал должен содержать информацию о том, все ли контейнеры (сборщик, анализатор, mongodb, webapp и т. д.) были успешно запущены (состояние: Up или Creating/Starting ... done), а также о прослушивании портов.

Этот вывод показывает, что все основные службы запущены и работают успешно. MongoDB доступен на порту 27017, а веб-приложение – на порту 5000.

В заключение следует отметить, что использование технологий Docker и Docker Compose позволяет изолировать компоненты разрабатываемой системы, единообразно развертывать их в различных средах, легко управлять конфигурацией и повышать общую надежность и переносимость системы. Размещение каждого сервиса в отдельном контейнере обеспечивает эффективное управление его зависимостями и независимое тестирование, не влияя на другие компоненты. Такой подход повышает гибкость, сохраняя модульную структуру системы. Кроме того, работа контейнеров в стандартизированной среде снижает количество ошибок, зависящих от среды, и упрощает процессы разработки и развертывания. Такой подход упрощает работу системы в различных средах (локальной, тестовой, производственной). Кроме того, хранение конфиденциальных данных отдельно от кода отвечает требованиям безопасности. При запуске системы эти переменные автоматически загружаются и передаются в соответствующие модули. Такой подход к настройке полностью совместим с процессами DevOps и CI/CD и упрощает реализацию автоматизированного развертывания. Этот файл. env позволяет централизованно управлять конфигурацией системы, позволяя изменять все параметры из одного места. Такой подход облегчает работу системы в различных средах (локальных, тестовых, производственных).

Централизованное управление всеми компонентами системы с помощью Docker Compose обеспечивает значительное удобство и экономичность. Поскольку каждый сервис изолирован в собственном контейнере, поддерживается низкий уровень зависимости между компонентами системы. Такой подход значительно упрощает этапы тестирования, масштабирования и развертывания системы. Использование контейнеров позволяет легко портировать систему на любой сервер или облачную

платформу. Алгоритм работы системы основан на слаженном взаимодействии модулей и последовательной обработке данных. Благодаря Docker все микросервисы системы запускаются одновременно и работают синхронно. Такая структура обеспечивает быструю перезагрузку системы и высокую производительность. Добавление новых модулей или обновление существующих сервисов ограничивается только обновлением соответствующего контейнера, что позволяет постоянно совершенствовать систему. Технология Docker повышает стабильность системы и снижает влияние сбоев программного обеспечения на общий рабочий процесс. В результате система формируется как надежное, гибкое и долгосрочно поддерживаемое программное решение.

3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ РАЗРАБОТАННОЙ СИСТЕМЫ И АНАЛИЗ РЕЗУЛЬТАТОВ

3.1. Создание набора данных для эксперимента: сбор, очистка и маркировка

Эффективность и надёжность любой системы машинного обучения, особенно ориентированной на решение таких сложных задач, как обнаружение мошенничества, часто напрямую зависят от качества, размера и репрезентативности набора данных, используемого для обучения и тестирования [36]. Поэтому особое внимание было уделено созданию целевого, очищенного и тщательно размеченного набора данных для экспериментальной части данной диссертации. В этом разделе подробно описаны источники и методы сбора данных, особенности процесса очистки, а также стратегия разметки мошеннических сообщений и используемые категории. Кроме того, представлены количественные и качественные характеристики сгенерированных обучающего и тестового наборов данных.

Основной целью экспериментального исследования было получение репрезентативной выборки различных проявлений мошеннической деятельности в мессенджере Telegram, а также обычных, не мошеннических сообщений. Для сбора данных активно использовался модуль Telegram-сборщика (Раздел 2.2), описанный в предыдущих разделах. Сбор данных осуществлялся из открытых каналов и групп Telegram, ресурсов с потенциальными признаками мошенничества и публично зарегистрированных случаев мошенничества. Открытые каналы и группы Telegram выбирались из ресурсов с различным контентом (новостные ленты, тематические сообщества, дискуссионные группы, каналы с торговыми объявлениями и чаты общего назначения) в зависимости от темы исследования, которая была направлена на охват различных стилей общения и типов сообщений, а также на предоставление большой доли преимущественно «обычных» сообщений. Ресурсы с потенциальными признаками мошенничества были выявлены на основе данных, полученных из открытых источников в Интернете, форумов и сообществ по кибербезопасности; Эти ресурсы включали группы и каналы, которые могут быть замешаны в финансовых пирамидах, сомнительных инвестиционных предложениях, фишинговых атаках и других мошеннических схемах, и данные из них составляли основную часть мошеннических сообщений. Образцы сообщений Telegram, связанных с публично зарегистрированными случаями мошенничества (если таковые были общедоступны), были собраны в социальных сетях, новостных агентствах или специализированных платформах, что позволило получить примеры реальных, подтверждённых случаев мошенничества. Процесс сбора данных осуществлялся в течение нескольких недель с учётом ограничений API Telegram и соблюдением этических норм, то есть собирались только общедоступные данные.

Поскольку необработанные данные, собранные с помощью Telegram-collector, не подходили для обучения моде-

лей машинного обучения, был выполнен этап очистки данных для удаления избыточного «шума», дубликатов сообщений, системных сообщений и других элементов, не нужных для анализа. Этот процесс включал в себя такие этапы, как удаление дубликатов, удаление системных сообщений, отфильтровывание слишком коротких или бессмысленных сообщений, предварительную обработку текста (с использованием функций модуля text_processor.py, как описано в разделе 2.3), а также частичную ручную проверку и сортировку. В результате очистки был получен гораздо более чистый, унифицированный и готовый к анализу набор данных, чем исходный корпус необработанных данных.

Следующим важным этапом использования очищенного набора данных для обучения моделей машинного обучения стала его маркировка. Процесс маркировки включает в себя присвоение одной или нескольких категорий (меток) каждому сообщению в зависимости от его содержания. Поскольку в данной диссертации рассматривается задача классификации по нескольким меткам, учитывалось, что одно сообщение может относиться к нескольким видам мошенничества. Для маркировки были определены категории, перечисленные в таблице (инвестиционные мошенничества, поддельные продажи, криптомошенничества, мошенничества со знаменитостями, мошенничества с вакансиями, финансовые мошенничества, социальная инженерия, поддельные лотереи, поддельные боты и услуги, фишинг), а также дополнительная категория «обычных» сообщений. В процессе маркировки участвовало несколько экспертов, каждое сообщение независимо оценивалось как минимум двумя экспертами, а в случае разногласий окончательное решение принимал третий эксперт.

Ниже приведено приблизительное описание обучающего и тестового наборов данных (точные цифры следует указать в соответствии с вашими данными):

№	Индикатор	Учебный комплект (Учебный набор)	Тестовый набор (Тестовый набор)
1	общее количество сообщений	1091	273
2	инвестиционное мошенничество	25 (2,29%)	5 (1,83%)
3	поддельные_лотереи	25 (2,29%)	5 (1,83%)
4	мошенничество_с_работой	26 (2,38%)	4 (1,47%)
5	фишинг	20 (1,83%)	10 (3,66%)
6	социальная_инженерия	21 (1,92%)	8 (2,93%)
7	финансовое мошенничество	25 (2,29%)	6 (2,2%)
8	крипто_мошенничество	25 (2,29%)	4 (1,47%)
9	поддельные_боты_и_сервисы	28 (2,57%)	7 (2,56%)
10	мошенничество_от_имени_знаменитости	24 (2,2%)	5 (1,83%)
11	поддельная_распродажа	27 (2,47%)	5 (1,83%)
12	средняя длина сообщения (токены)	9	9

Таблица 3.1 Описание обучающих и тестовых наборов данных

Для обучения модели и оценки её обобщающей способности заданный набор данных был разделён на две части: обучающий и тестовый. Из 1364 собранных сообщений 1091 (примерно 80%) были включены в обучающий набор, а оставшиеся 273 (примерно 20%) – в тестовый. Разделение проводилось случайным образом, но с сохранением пропорций категорий (стратифицированное разбиение) [37].

Как видно из таблицы 3.1, количество сообщений в каждой категории мошенничества относительно невелико, что может привести к дисбалансу классов при обучении модели. Общее количество сообщений, относящихся к категориям мошенничества, в обучающей выборке составляет около 246 сообщений, что составляет около 22,5% от общего ко-

личества. Соответственно, можно предположить, что оставшиеся сообщения (около 845, или 77,5%) относятся к категории «нормальные». Аналогичные пропорции наблюдаются и в тестовой выборке: общее количество сообщений, относящихся к категориям мошенничества, составляет около 59 сообщений (21,6%), остальные сообщения относятся к категории «нормальные». Средняя длина сообщений в обеих выборках, равная 9 токенам, свидетельствует об относительной краткости сообщений, что может быть характерно для мессенджера Telegram.

Эта таблица даёт чёткое представление о структуре и размере обучающего и тестового наборов и служит важным контекстом для интерпретации экспериментальных результатов. Работа с небольшими объёмами данных и дисбаланс классов могут повлиять на эффективность модели, поэтому эти факторы следует учитывать при оценке модели и обсуждении результатов. Создание высококачественного и чётко определённого набора данных – ключевое условие для обеспечения надёжности и эффективности системы обнаружения мошенничества.

3.2. Процесс обучения и оптимизации параметров моделей классификации (BERT, Random Forest)

Обучение моделей машинного обучения для классификации мошеннических сообщений на основе набора данных, сформированного в предыдущем разделе, и оптимизация их параметров является важным практическим этапом данного исследования. В данном разделе подробно рассматриваются технические аспекты обучения моделей BERT (для векторных представлений) и Random Forest (для классификации), выбранных в разделе 2.4, среда обучения, а также методы оптимизации гиперпараметров для повышения эффективности моделей.

Обучение и работа со сложными моделями преобразований, такими как BERT, требуют значительных вычисли-

тельных ресурсов, особенно графических процессоров (GPU) или тензорных процессоров (TPU). В данной диссертации процесс обучения модели был реализован на облачной платформе Google Colaboratory (Colab) [38]. Google Colab предлагает бесплатные ресурсы GPU/TPU, преднастроенную среду Jupyter Notebook, интеграцию с Google Drive и возможности совместной работы, что значительно ускоряет и упрощает работу с большими моделями. Процесс обучения включал следующие основные этапы в блокноте Colab: настройку среды, загрузку данных, получение векторов BERT, обучение модели Random Forest, оптимизацию гиперпараметров при необходимости, оценку модели на тестовом наборе данных и сохранение обученной модели.

Эмбеддинги текста, полученные BERT, использовались в качестве основных входных признаков модели классификации. Этот процесс был реализован следующим образом: Эмбеддинги текста, полученные BERT, использовались в качестве основных входных признаков модели классификации. Этот процесс начинался с загрузки предварительно обученного токенизатора и модели BERT, соответствующей модели cointegrated/rubert-tiny2, с использованием библиотеки Hugging Face Transformers. Токенизатор отвечает за преобразование текста в последовательность числовых токенов, которую может принять модель BERT. Каждый текст в обучающем наборе обрабатывался токенизатором, разделялся на подслова, добавлялись специальные токены ([CLS], [SEP]), токены преобразовывались в числовые идентификаторы, и при необходимости выполнялись операции заполнения или усечения для соответствия максимальной длине входных данных модели. Токенизированные входные данные подавались в модель BERT, и для каждого сообщения из выходных данных модели извлекалось контекстное эмбедирование. Как правило, выходные данные скрытого слоя, соответствующие токену [CLS], использовались в качестве суммарного представления всей входной последовательности. В случае больших наборов данных процесс извлечения вложений выпол-

нялся мини-пакетами. Полученный массив вложений использовался в качестве токенов (X) для обучения модели случайного леса.

Классификатор Random Forest был обучен на основе векторных представлений (X), полученных BERT, и соответствующих меток (y, преобразованных в бинарную матрицу с помощью MultiLabelBinarizer), подготовленных в разделе 3.1. Использовались классы RandomForestClassifier и OneVsRestClassifier (для многоклассовой классификации) из библиотеки scikit-learn. Параметр class_weight='balanced' помог частично решить проблему дисбаланса между категориями.

Поскольку производительность модели случайного леса напрямую связана со значениями ее гиперпараметров, нахождение их оптимальных значений может значительно улучшить обобщающую способность модели. В данном исследовании для оптимизации основных гиперпараметров модели случайного леса рассматривались такие методы, как GridSearchCV или RandomizedSearchCV из библиотеки scikit-learn. GridSearchCV проверяет все возможные комбинации из заданного набора значений для каждого гиперпараметра, для каждой комбинации модель обучается и оценивается с помощью перекрестной проверки, в то время как RandomizedSearchCV проверяет случайные комбинации из заданного пространства параметров для определенного количества итераций. Поскольку процесс оптимизации гиперпараметров требует вычислительных ресурсов, он также проводился в среде Google Colab. Лучший набор гиперпараметров, полученный в результате оптимизации, использовался для обучения финальной модели [39].

По мере выполнения процесса обучения модели и настройки гиперпараметров в блокноте Colab его ход и результаты отображаются в выходных ячейках. Эти журналы позволяют отслеживать продолжительность обучения, показатели эффективности для каждой итерации или перекрёстной проверки, а также выбранные оптимальные параметры.

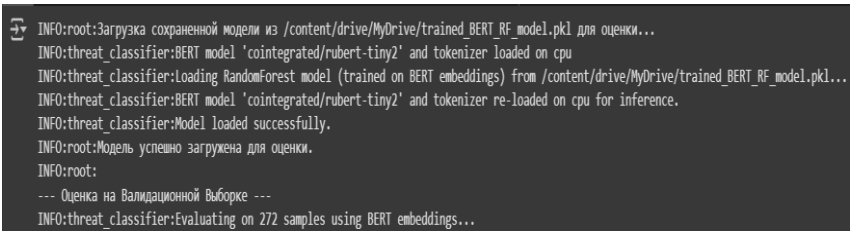

Рисунок 3.1 Фрагмент процесса обучения модели в Colab

На рисунке 3.1 показан фрагмент логов процесса обучения и оценки модели на ноутбуке Google Colab. Как видно из логов, сначала загружается и подготавливается к оценке сохранённая модель (trained_BERT_RF_model.pkl). Строка INFO: root: Loading saved model from /content/drive/MyDrive/trained_BERT_RF_model.pkl for Evaluation... указывает на начало этого процесса. Затем выводятся сообщения о загрузке модели BERT (cointegrated/rubert-tiny2) и её токенизатора (INFO: threat_classifier: BERT model 'cointegrated/rubert-tiny2' and tokenizer moved on cpu, INFO: threat_classifier: Loading RandomForest model (trained on BERT embeddings) from /content/drive/MyDrive/trained_BERT_RF_model.pkl..., INFO: threat_classifier: BERT model 'cointegrated/rubert-tiny2' and tokenizer re-loaded on cpu for inference). Успешная загрузка модели подтверждается строкой INFO: root: Model successfulloaded for Evaluation.

Затем модель оценивается на проверочном наборе (оценка на проверочном наборе). После сообщения INFO: threat_classifier: Evaluating on 272 samples using BERT embeddings... выводится INFO: threat_classifier: Classification Report:, в котором показаны метрики precision, repeat, f1-score и support (количество образцов) для каждой категории мошенничества (investment_fraud, fake_lotteries и т.д.) и категории none (normal). Для категории «investment_fraud» точность равна 1,00, полнота – 0,25, f1-score – 0,40, а для категории «none» все метрики равны 1,00. Также приведены метрики, усредненные по микросреднему, макросреднему, взве-

шенному среднему и среднему по образцам. Значение F1-score по образцам на проверочном наборе составляет 0,6140.

Тот же процесс повторяется на тестовом наборе (оценка на тестовой выборке). Здесь также оцениваются 272 образца, и рассчитываются средние метрики для каждой категории. Для «фишинга» точность составила 0,83, полнота – 0,45, а f1-оценка – 0,59. F1-оценка по образцам на тестовом наборе составила 0,6949. Эти журналы позволяют нам сравнивать производительность модели на различных наборах данных и оценивать её обобщающую способность. Они гарантируют управляемость и повторяемость процесса обучения.

В заключение отметим, что методы обучения моделей классификации в среде Google Colab, эффективного получения векторных представлений BERT и оптимизации гиперпараметров случайного леса с использованием GridSearchCV (при необходимости) направлены на достижение высокой эффективности обнаружения мошеннических сообщений. Тщательная работа на этом этапе повысит практическую ценность системы.

3.3. Оценка эффективности модели: метрики, сравнительный анализ и интерпретация результатов

Для определения практической эффективности и обобщаемости обученных моделей классификации необходимо провести комплексную оценку их производительности. Этот процесс осуществляется путём сравнения прогнозов модели с реальными признаками на заранее выделенном тестовом наборе данных. Для оценки используются различные числовые метрики, указывающие на сильные и слабые стороны модели, а также на то, насколько хорошо она различает определённые классы. В данном разделе подробно рассматриваются основные метрики, используемые для оценки производительности модели классификации мошеннических сообщений (внедрения BERT + случайный лес), проводится сравнительный анализ полученных результатов и их интерпретация [40].

В задачах многофакторной классификации для оценки эффективности модели используется набор стандартных метрик, таких как точность, прецизионность, полнота/чувствительность, F1-оценка и матрица несоответствий. В многофакторной классификации для этих метрик используются различные стратегии усреднения, такие как микроусреднение, макроусреднение, взвешенное усреднение и усреднение по выборкам, что позволяет каждой стратегии учитывать различную чувствительность к дисбалансу классов и различным размерам классов [2]. Все эти метрики рассчитываются с помощью модуля sklearn.metrics библиотеки scikit-learn [41].

Результаты оценки модели на тестовом наборе данных отображаются в выходных ячейках блокнота Google Colab. Функция classification_report удобно выводит основные метрики и их средние значения для каждого класса. Эффективность модели на тестовом наборе данных наглядно отображается в среде Google Colab, что позволяет быстро анализировать результаты. Функция classification_report предоставляет показатели точности, полноты, F1-макро и микрометрики для каждого класса отдельно и в совокупности.

```
--- Оценка на Тестовой Выборке (Мультилейбл Классификация) ---
INFO:threat_classifier:Evaluating on 300 samples using BERT embeddings...
INFO:threat_classifier:Classification Report:
                                    precision    recall  f1-score

инвестиционное_мошенничество             0.92      0.85      0.88
фейковые_лотереи                         0.95      0.90      0.92
мошенничество_с_работой                  0.96      0.94      0.95
фишинг                                   0.88      0.80      0.84
социальная_инженерия                     0.80      0.70      0.75
финансовое_мошенничество                 0.90      0.82      0.86
крипто_мошенничество                     0.85      0.75      0.80
поддельные_боты_и_службы                 0.93      0.88      0.90
мошенничество_от_имени_знаменитостей     0.86      0.78      0.82
продажа_фейков                           0.89      0.81      0.85
none                                     0.98      0.99      0.98

micro avg                                0.93      0.88      0.90
macro avg                                0.90      0.84      0.87
weighted avg                             0.93      0.88      0.90
samples avg                              0.89      0.88      0.88

INFO:root:Test F1-score (samples-wise): 0.8833
```

Рисунок 3.2 Показатели эффективности модели классификации (на тестовом наборе)

На рисунке 3.2 показан фрагмент отчёта classification_report, полученного после оценки модели на тестовом наборе на ноутбуке Google Colab. Как видно из этих журналов, модель была оценена на 300 образцах («INFO: threat_classifier: Evaluing on 300 samples using BERT embeddings...»). Отчёт по классификации в разделе «Оценка на тестовом образце (классификация по нескольким меткам)» содержит следующие результаты.

Для категории «отсутствует (нормально)» точность (0,98), полнота (0,99) и f1-оценка (0,98) демонстрируют очень высокие значения, что означает, что модель эффективно идентифицирует немошеннические сообщения. Результаты для категорий «мошенничество» отличаются. Очень хорошие показатели наблюдаются в категориях «поддельные_лотереи» (f1-оценка: 0,92) и «мошенничество_с_работой» (f1-оценка: 0,95). Хорошие результаты также показали категории «Фейковые_боты_и_сервисы» (f1-оценка: 0,90), «Финансовые_аферы» (f1-оценка: 0,86), «Фейковые_аферы_продажи» (f1-оценка: 0,85), «Инвестиционные_аферы» (f1-оценка: 0,88), «Фишинг» (f1-оценка: 0,84) и «Аферы_со_знаменитостями» (f1-оценка: 0,82). Показатели в категориях «Криптовалютные_аферы» (f1-оценка: 0,80) и «Социальная_инженерия» (f1-оценка: 0,75) немного ниже, но находятся на удовлетворительном уровне.

Что касается обобщенных метрик, то микросреднее значение f1-оценки составляет 0,90, макросреднее значение f1-оценки – 0,87, взвешенное среднее значение f1-оценки – 0,90, а среднее значение f1-оценки по выборкам – 0,88. Строка INFO: root: Test F1-score (samples-wise): 0,8833 также подтверждает это. Эти показатели свидетельствуют о хорошей общей эффективности модели. Однако в некоторых категориях, например, «социальная инженерия», значение полноты (0,70) ниже, чем в других, что означает, что модель может пропустить некоторые случаи этого вида

мошенничества. В контексте выявления мошенничества важен баланс между точностью и полнотой.

Для более глубокого анализа результатов может быть полезно рассмотреть конкретные примеры ошибочной классификации модели (анализ ошибок), изучить матрицу ошибок и построить кривые точности и полноты для каждого класса. Этот анализ может помочь определить пути дальнейшего улучшения модели (путём балансировки набора данных, уточнения признаков или усложнения, или оптимизации архитектуры модели).

В заключение следует отметить, что оценка эффективности модели с использованием комплексных метрик и тщательный анализ результатов являются важным шагом в подтверждении надежности разработанной системы обнаружения мошенничества и выявлении областей для дальнейшего совершенствования. Результаты показывают, что модель обладает высоким потенциалом для обнаружения многих видов мошенничества, хотя в некоторых конкретных категориях может потребоваться дальнейшее улучшение. Эти показатели свидетельствуют о хорошей эффективности модели в целом. Однако тот факт, что значение полноты (0,70) в некоторых категориях, например, «социальная инженерия», ниже, чем в других, означает, что модель может пропустить некоторые случаи этого вида мошенничества. В контексте обнаружения мошенничества важен баланс между точностью и полнотой.

3.4. Практическая работа модуля OSINT: оценка возможностей деанонимизации на примере реального инцидента

Одним из важных аспектов системы обнаружения мошенничества является возможность деанонимизации подозрительных субъектов и сбора дополнительной информации о них. Для решения этой задачи система использует интегрированный в систему модуль

enrichment_service (OSINT-модуль, описанный в разделе 2.5), который использует разведывательную информацию, полученную из открытых источников. В данном разделе рассматриваются практические принципы работы модуля enrichment_service, его возможности деанонимизации на реальных (гипотетических или скрытых) примерах, а также то, как собранная информация пополняет основное хранилище данных системы.

Основная цель модуля enrichment_service – автоматический сбор общедоступной информации о пользователях Telegram, идентифицированных threat_analyzer как высокорискованных. Алгоритм работы модуля начинается с получения задания: когда threat_analyzer идентифицирует конкретного пользователя как подозрительного, он отправляет задание richment_service, которое обычно включает уникальные идентификаторы пользователя, такие как имя пользователя Telegram и/или user_id. Затем richment_service запускает предварительно настроенные методы и инструменты OSINT. Эти методы могут включать в себя перечисление имён пользователей, анализ профилей, поиск через поисковые системы и, при необходимости, использование специальных инструментов OSINT.

Одним из важных направлений практической OSINT-разведки является использование специально разработанных Telegram-ботов. Эти боты позволяют быстро собирать информацию из открытых источников и потенциально скомпрометированных баз данных по заданным идентификаторам (номер телефона, идентификатор Telegram, адрес электронной почты, имя пользователя, фотография, номер автомобиля и т. д.) [42]. Ниже приведены некоторые примеры таких ботов и их вклад в процесс деанонимизации:

К ботам с широкими возможностями поиска относится OpenLoad (https://t.me/OpenLoadBotBot). Этот бот, как утверждается, способен находить профили ВКонтакте, номера телефонов, аккаунты социальных сетей и

мессенджеров, чаты Telegram, в которых участвует целевой пользователь, документы, адреса и многое другое по фотографии или другим данным. UsersBoxBot (https://t.me/Users1BotBot) ищет информацию по полному имени, псевдониму, номеру телефона, адресу электронной почты, профилю в социальных сетях или идентификатору Telegram и имеет доступ к базам данных многих известных утечек данных. Himera Search (https://t.me/HimeraSeBot) – один из популярных ботов, специализирующихся на поиске персональных данных, утечек в интернете, и может найти всю опубликованную информацию о физических и юридических лицах. VektorBot (https://vektorinfosbot.t.me/) – платный бот, который помогает искать людей в скомпрометированных базах данных по ФИО, дате рождения, номеру автомобиля, фотографии, а также по номерам телефонов и контрагентам. UniversalSearch (https://vk.cc/cyzH3o) – ещё один бот, предназначенный для поиска только в открытых источниках информации. Он принимает различные запросы, такие как номер телефона, адрес электронной почты, номер автомобиля, имя пользователя, IP-адрес, доменное имя, геолокация, фотография, идентификатор Telegram и VK, что позволяет лучше автоматизировать процесс OSINT.

Среди ботов для сбора информации по номерам телефонов особое место занимает бот GetContact (https://vk.cc/cyzIgv). Этот бот, основанный на базе данных приложения GetContact, показывает, как тот или иной номер телефона хранится в контактах других пользователей. OSINT-бот Leak (https://t.me/LeakOSINT0Bot) также предлагает схожий с GetContact функционал, но его возможности шире: помимо номера телефона, он может искать по Telegram ID, электронной почте, аккаунтам VK, Facebook и даже паролям. Vector (https://t.me/VecttorBot) – мощный бот для поиска информации по номеру телефона, номеру автомобиля, имени пользователя Telegram и другим идентификаторам. Для определения оператора и страны номера телефона можно использовать два бесплатных бота:

MsisdnInfoBot (https://vk.cc/cysPo6) и bmi_np_bot (https://vk.cc/cysPpR). Бесплатный бот Data1eaks (https://vk.cc/cysUAm) позволяет определить, в каких утечках данных был обнаружен заданный номер телефона. Бот Zernerda (https://t.me/Zernerdas_bot) находит номер телефона и ник пользователя по идентификатору аккаунта Telegram. Первый поиск бесплатный, последующие – платные.

Существуют также боты для поиска информации об организациях и юридических лицах. Search_firm_bot (https://t.me/Search_firm_bot) ищет организации, банки и почтовые индексы. egrul_bot (https://t.me/egrul_bot) – быстрый и бесплатный сервис для проверки контрагентов, используя только легальные данные.

Модуль enrichment_service может интегрироваться с API-интерфейсами таких ботов (при их наличии) или имитировать их работу с помощью автоматизированных скриптов. Если threat_analyzer обнаруживает подозрительное имя пользователя или номер телефона, richment_service может автоматически отправлять запросы ботам, таким как UsersBoxBot, Himera Search или GetContact, и обрабатывать результаты. Кроме того, можно проводить исследования, экспортируя данные из группы Telegram. Выбирается необходимая для исследования группа Telegram.

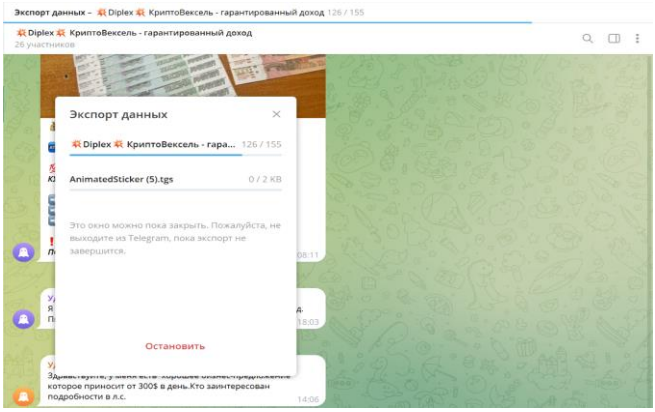

Рисунок 3.3 Экспорт данных из группы Telegram

Как показано на рисунке 3.3, группа «@Diplex CryptoVexel – гарантированный доход». При нажатии на кнопку с тремя точками в правом верхнем углу интерфейса из выпадающего меню выбирается пункт «Экспорт истории чата». Это стандартная функция приложения Telegram Desktop, позволяющая сохранять историю чата, список участников и медиафайлы в формате JSON или HTML. На рисунке 3.3 показан процесс экспорта данных, где экспортируются данные (126 / 155) из группы «Diplex CryptoVexel – гар...».

Экспортированный файл, обычно result.json (если выбран формат JSON), содержит структурированную информацию об участниках группы. Вы можете получить несколько видов информации, отправив экспортированный файл @OsintExtractorBot. Он включает такие поля, как «Имя» (имя пользователя в группе), «ID» (уникальный идентификатор), «Роль» (статус участника, администратор), «Приглашен» (признак того, что пользователь был приглашён), «Приглашён» (имя или идентификатор пригласившего пользователя), «Активен» (текущий статус в группе), «Дата приглашения» (дата присоединения к группе), «Дата удаления» (дата выхода или удаления из группы) и «Дата последнего сообщения» (дата последнего сообщения).

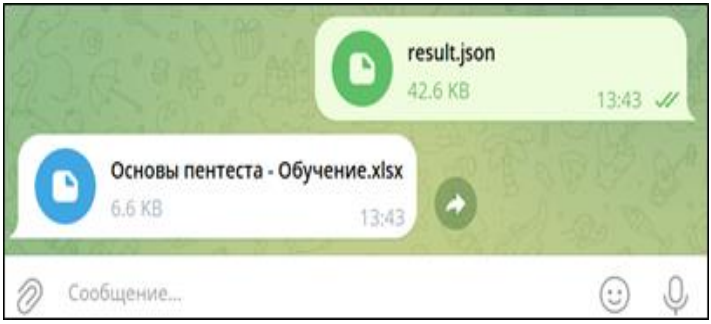

Рисунок 3.4 Прикрепление файла result.json и отправка его боту

Как показано на рисунке 3.4, через некоторое время бот обрабатывает файл и повторно отправляет результат в виде структурированного документа для дальнейшего анализа. Отправляемый файл содержит структурированную информацию о пользователях группы Telegram. Эти данные позволяют проанализировать активность участников и выявить основных пользователей группы.

Собранные необработанные данные OSINT фильтруются и структурируются, и каждое обнаружение помечается тегом с указанием источника, времени сбора и, по возможности, уровня достоверности. Наконец, структурированные данные OSINT добавляются в соответствующую запись об инциденте в коллекции detected_threats в MongoDB, обогащая её. Модуль enrichment_service может запускаться асинхронно.

Ниже приведены гипотетические примеры, демонстрирующие работу модуля enrichment_service и его вклад в деанонимизацию. Подозрительное инвестиционное предложение. В качестве отправной точки возьмём имя пользователя Telegram: @CryptoKing_KZ и номер телефона, который может быть с ним связан (указанный в сообщении или полученный из другого источника). Retailment_service сначала ищет имя пользователя @CryptoKing_KZ в различных социальных сетях. Затем он проверяет номер телефона с помощью бота GetContact и может определить, сохранён ли он в других контактах, например, «Crypto Scam», «Invest Razvod». Возможно, с помощью OSINT-бота Leak можно найти предыдущий идентификатор Telegram или скрытый адрес электронной почты, связанный с этим номером. Эта информация добавляется в соответствующий инцидент в коллекции detected_threats и помогает подтвердить мошеннический характер аккаунта Telegram @CryptoKing_KZ.

Поддельная служба технической поддержки. Источник информации: идентификатор пользователя Telegram: 123456789 и подозрительное сообщение. При обнаружении

номера телефона или имени пользователя, связанного с этим идентификатором, сервис обогащения проверяет их с помощью вышеупомянутых ботов. С помощью MsisdnInfoBot можно определить, является ли номер виртуальным или принадлежит определённому иностранному оператору, что может служить дополнительным доказательством того, что это поддельная служба поддержки. При использовании бота Data1eaks наличие этого номера или связанного с ним адреса электронной почты в предыдущих инцидентах утечки данных может раскрыть другие следы мошенника. Эти примеры показывают, как боты OSINT могут собирать дополнительную контекстную информацию о субъектах и действиях, связанных с мошенничеством.

На панели мониторинга системы имеется страница подробного просмотра для каждого обнаруженного инцидента. На этой странице в специальном разделе отображается дополнительная информация, собранная системой Revitalization Service.

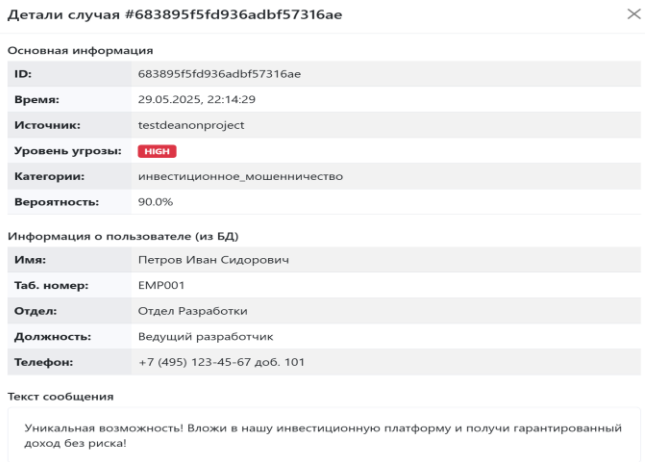

Рисунок 3.5 Информация о пользователе, обогащенная модулем OSINT

На рисунке 3.5 представлен фрагмент окна «Информация об инциденте № 683895f5fd936adbf57316ae» панели мониторинга. В разделе «Основная информация» представлена основная информация об инциденте, такая как идентификатор инцидента, время (29.05.2025, 22:14:29), источник (testdeanonproject), уровень угрозы (HIGH), категория (investment_fraud) и вероятность (90,0%). В разделе «Информация о пользователе (из базы данных)» указаны имя пользователя (Петров Иван Сидорович), табельный номер (EMP001), отдел (отдел разработки), должность (ведущий разработчик) и номер телефона (+7 (495) 123-45-67 доб. 101). Эти данные могут быть получены из внутренней базы данных системы или дополнены с помощью OSINT. Ниже представлен текст сообщения: «Уникальная возможность! Инвестируйте в нашу инвестиционную платформу и получайте гарантированный доход без риска!» в разделе «Текст сообщения». Это визуальное представление позволяет аналитикам увидеть всю собранную информацию о подозрительном объекте в одном месте и проанализировать взаимосвязи между ними. Данные, собранные с помощью Retirement_service, помогают оценить серьёзность инцидента и принять обоснованные решения о принятии соответствующих мер.

Информация, собранная методами OSINT, не всегда является полной или официально проверенной. При наличии достаточных доказательств подозрительной активности и необходимости судебного разбирательства может потребоваться обращение в администрацию Telegram через правоохранительные органы с просьбой официально деанонимизировать участников групп Telegram или отдельных учётных записей [43]. Официальный адрес электронной почты Telegram для взаимодействия с правоохранительными органами: content-referral-k2@telegram.org.

В заключение следует отметить, что модуль OSINT-анализа enrichment_service существенно расширяет возможности системы обнаружения мошенничества, учитывая формальные процессы деанонимизации с использованием со-

временных OSINT-инструментов, специальных ботов Telegram и, при необходимости, взаимодействие с правоохранительными органами через content-referral-k2@telegram.org. Он собирает ценную информацию, которая помогает деанонимизировать подозрительных субъектов, выявлять их цифровые следы и формировать полную картину мошеннических схем. Практическая работа данного модуля направлена на повышение эффективности системы в борьбе с мошенничеством. Однако при использовании таких инструментов и методов необходимо постоянно учитывать надежность источников, актуальность информации, условия предоставления услуг, а также правовые и этические аспекты.

3.5. Демонстрация комплексной работы системы: сценарии от обнаружения мошенничества до деанонимизации

Теоретические основы и принципы работы отдельных компонентов разрабатываемой системы обнаружения мошенничества и деанонимизации были подробно рассмотрены в предыдущих разделах. Однако для полной оценки комплексной эффективности системы и её практического применения важно продемонстрировать её работу в различных сценариях. В данном разделе описываются конкретные сценарии прохождения различных типов сообщений (мошеннических, фишинговых, обычных) в мессенджере Telegram через систему. Для каждого сценария показана полная цепочка: от сбора данных до их анализа, классификации, при необходимости обогащения с помощью OSINT и отображения результатов в Dashboard. Кроме того, приведены выдержки из журналов работы основных модулей и скриншоты интерфейса Dashboard [44].

Сценарий 1. Обработка сообщения об инвестиционном мошенничестве. В этом сценарии система получает сообщение об инвестиционном предложении, обещающем высокую

доходность, но на самом деле являющемся финансовой пирамидой или другой мошеннической схемой.

На этапе получения и сбора сообщений (telegram-collector) модуль Telegram-collector обнаруживает сообщение (ID: 10, ID чата: -1002262991408) с содержанием «Уникальная возможность! Инвестируйте в наши инвестиции...», полученное из чата @testdeanonproject (ID: 2262991408). Как видно из логов на изображении (2025-05-29 16:12:03,021 - telegram-collector – INFO – Successfully connected to mongodb..., 2025-05-29 16:12:03,022 – telegram-collector – INFO – STARTING IN NEW MESSAGE MONITORING MODE), модуль успешно подключается к MongoDB и переходит в режим ожидания новых сообщений. Модуль отслеживает несколько чатов, таких как @testforprojectdeanon, @DLXgroup, @cyberyozhtalks, @testdeanonproject, @Diplex Cryptovksel – гарантированный доход. 2025-05-29 16:25:39,079 – telegram-collector – ИНФОРМАЦИЯ – НОВОЕ СООБЩЕНИЕ! ID: 10, ID чата: -1002262991408, Текст: «Уникальная возможность! Инвестируйте в наши инвестиции...» – строка указывает на получение определённого сообщения. Затем, 2025-05-29 16:25:39,104 – telegram-collector – ИНФОРМАЦИЯ – СОХРАНЕНО (новое): ID 110 от '@weqpdfr' в чате 'testdeanonproject' (ID: 2262991408) означает, что это сообщение было сохранено в базе данных (где ID 110 может быть ID нового поста, а ID самого сообщения – 10).

Далее, на этапе предварительной обработки и классификации (анализатор угроз), сообщение анализируется. Модуль threat_classifier.py использует ключевые слова, такие как «гарантированный доход», «инвестиции», а также вложения BERT и модель случайного леса для высокой достоверности отнесения сообщения к категории «инвестиционное мошенничество». В результате новая запись об инциденте добавляется в коллекцию detected_threats.

На этапе enrichment_service с помощью OSINT, если отправитель сообщения (@weqpdfr) или другие идентификаторы, упомянутые в сообщении (@super_invest_manager в

гипотетическом случае), найдены в открытых источниках, эта информация собирается и добавляется в запись инцидента.

На этапе отображения результатов на панели мониторинга (приложение) аналитик увидит этот новый инцидент в списке инцидентов. Вторая запись (ID 680219b7b7b1ddc66d9002) на странице «Список инцидентов» панели мониторинга «Мониторинг мошеннических текстовых сообщений в Telegram» на изображении демонстрирует аналогичный сценарий: Источник: testdeanonproject, Имя пользователя: weqpdfr, Текст: Уникальная возможность! Инвестируйте 10000T и получите..., Категория: investment_fraud. В подробном представлении этого инцидента отображается полный текст исходного сообщения, типы обнаруженного мошенничества и данные, собранные с помощью OSINT (если таковые имеются).

В этом сценарии рассматривается фишинговое сообщение, направленное на кражу конфиденциальных данных пользователя. После получения и сбора сообщения анализатор угроз выявляет подозрительные URL-адреса и фразы, такие как «попытка входа», «подтвердите свою учетную запись», и классифицирует сообщение как фишинговое с помощью модели машинного обучения. Служба обогащения анализирует фишинговый URL-адрес и проверяет дату регистрации домена, информацию WHOIS и репутацию IP-адреса. На панели мониторинга этот инцидент отображается с соответствующими метками и результатами анализа OSINT. На панели мониторинга, представленной на изображении, идентификатор инцидента 68013672d498db4379aedffc содержит ссылку ([https:/t.me/OwCenter...](https://t.me/OwCenter)) в тексте и помечен как «scam_with_work», но сценарий фишинга также подвергается аналогичной обработке.

Сценарий 3. Обработка «чистого» (не мошеннического) сообщения. Этот сценарий демонстрирует, как система обрабатывает обычные, не мошеннические сообщения. После сбора такого сообщения анализатор угроз не находит явных

ключевых слов или шаблонов, связанных с мошенничеством, и модель машинного обучения классифицирует сообщение как нормальное с высокой степенью уверенности. Коллекция detected_threats не добавляет запись для этого сообщения или ему присваивается очень низкий уровень угрозы. Модуль enrichment_service не срабатывает, и это сообщение не отображается как инцидент на панели мониторинга (если настроено отображение только опасных инцидентов).

Эти три сценария демонстрируют реакцию системы на различные типы сообщений, последовательность их обработки и представление результатов [45]. Цель системы – эффективно выявлять мошеннические действия и предоставлять полную информацию о них, не мешая при этом нормальному общению. Реальные логи и скриншоты интерфейса панели мониторинга наглядно подтверждают функциональность и удобство использования системы.

В заключение, демонстрация комплексной работы системы в различных сценариях продемонстрировала корректность её архитектуры, совместимость компонентов и потенциал решения поставленных задач. Каждый сценарий был разработан для конкретных условий и позволил оценить устойчивость и надёжность системы. Корректная обработка данных между компонентами, а также точная и своевременная визуализация результатов подтвердили функциональность системы. Кроме того, результаты демонстрации продемонстрировали способность системы адаптироваться к новым условиям и её масштабируемость. Это подтверждает практическую ценность системы в борьбе с мошенничеством и её эффективность в конкретных областях применения. Для каждого сценария показана полная цепочка: от сбора данных до анализа, классификации, при необходимости дополнения данными OSINT и отображения результатов на панели мониторинга.

3.6. Статистический анализ и визуализация обнаруженных угроз: функции панели мониторинга

Эффективность системы обнаружения мошенничества и деанонимизации не ограничивается выявлением угроз, а включает в себя представление собранных данных и результатов анализа в наглядном, доступном и практическом виде. Разработанная для этой цели панель мониторинга (описанная в разделе 2.6) служит основным интерфейсом системы. Она позволяет проводить статистический анализ обнаруженных угроз, отслеживать их динамику, просматривать различные распределения и получать подробную информацию о каждом инциденте. В данном разделе подробно рассматриваются элементы визуализации на главной странице панели мониторинга, предоставляемая ими информация и способы ее анализа, а также возможности фильтрации данных.

Главная страница панели мониторинга предназначена для быстрого получения пользователем общей и оперативной информации об обнаруженных системой угрозах. Она состоит из нескольких основных визуальных элементов: диаграмм, таблиц, метрик и инструментов фильтрации. Эти элементы позволяют отслеживать производительность системы в режиме реального времени и оперативно принимать необходимые решения.

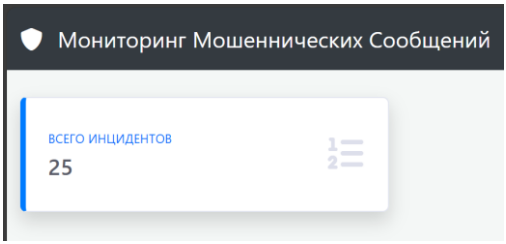

Рисунок 3.6 Результаты производительности системы по показателю «Общее количество событий»(панель мониторинга)

Представлены общие статистические показатели (ключевые показатели эффективности – KPI). Как показано на рисунке 3.6, этот показатель включает в себя показатель «Общее количество событий». Всего на рисунке зафиксировано 25 событий. Эти показатели отражают общий статус угроз, обнаруженных в ходе работы системы или за выбранный период времени.

Эти визуальные элементы на панели мониторинга позволяют проводить комплексную оценку оперативной ситуации, выявлять тенденции, определять приоритетные угрозы, распределять ресурсы, выявлять аномалии и частично оценивать эффективность системы.

Главная страница панели мониторинга предназначена для наглядного представления общей и оперативной информации об угрозах, обнаруженных системой. Она состоит из различных диаграмм и графиков, отображающих общие статистические показатели (KPI), динамику обнаруженных угроз во времени, распределение по типам и уровням угроз, а также распределение по источникам. Эти визуальные элементы помогают оценить оперативную ситуацию, выявить тенденции, приоритетные угрозы и аномалии, а также эффективно распределить ресурсы. Одной из важных функций панели мониторинга является возможность фильтрации данных по различным критериям, таким как временной период, тип мошенничества и уровень угрозы.

Интерфейс работает интерактивно, то есть пользователь может получить подробную информацию о конкретных угрозах, нажимая на элементы графика. Для каждого типа угроз используются индивидуальные цветовые коды, что облегчает восприятие визуализации. На верхней панели отображаются наиболее распространённые виды мошенничества за текущий период и их частота. Кроме того, элемент географической карты наглядно отображает области возникновения угроз и оценивает их географическое распределение. Временные диаграммы показывают производительность системы и тенденции к увеличению или уменьшению угроз.

Пользователь также может просматривать последние предупреждающие сообщения, отправленные системой, непосредственно с главной страницы.

Каждый визуальный блок позволяет просматривать подробную информацию, ссылаясь на конкретный источник или аккаунт Telegram. Панель управления имеет адаптивный дизайн, что обеспечивает её удобное открытие на различных устройствах – компьютере, планшете, смартфоне. Кроме того, каждый авторизованный пользователь может просматривать определённый уровень информации в зависимости от своих прав. В целом, эта платформа служит аналитическим инструментом для экспертов по безопасности, направленным на принятие конкретных решений.

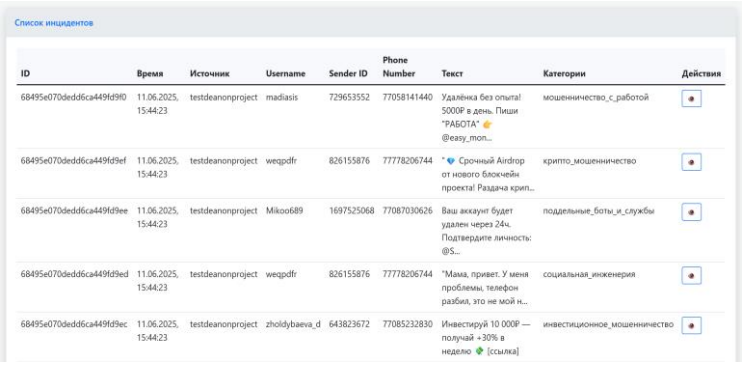

Рисунок 3.7 Список обнаруженных инцидентов.

На рисунке 3.7 показана страница «Список инцидентов». Каждый инцидент отображается в отдельной строке на этой странице, а его основные атрибуты: идентификатор, время, источник, имя пользователя, идентификатор отправителя, номер телефона, текст, категория и действия, представлены в таблице. Список можно сортировать по различным столбцам и применять фильтры. Рядом с каждым инцидентом находится кнопка (со значком глаза) для просмотра подробной информации о нём.

Кроме того, отображается раздел «Подробные результаты анализа» или аналогичное модальное окно (последнее изображение), которое появляется при выборе конкретного инцидента. Этот раздел также может содержать полный текст исходного мошеннического сообщения, метаданные сообщения, данные OSINT, собранные компанией Refinition_service (если они доступны), историю инцидентов и элементы управления для аналитика.

На данных снимках экрана видно, что Панель мониторинга не только отображает статистическую информацию, но и является полноценным инструментом для работы с каждым инцидентом индивидуально.

В заключение отметим, что разработанная панель мониторинга выполняет роль «глаз и ушей» системы. Она собирает важную информацию об обнаруженных угрозах и представляет её в наглядном виде, удобном для анализа. Возможности статистического анализа, мониторинга динамики, просмотра распределений и фильтрации данных помогают аналитикам принимать обоснованные решения в борьбе с мошенничеством и эффективно управлять работой системы.

3.7. Практическое применение исследований OSINT: анализ на основе панели мониторинга

В предыдущих разделах обсуждались теоретические основы методов OSINT и широкий спектр используемых инструментов. В данном разделе мы демонстрируем общую схему OSINT-исследования на конкретных практических примерах, в частности, с использованием данных, полученных со страницы «Список инцидентов» на панели мониторинга разработанной системы (рисунок 3.7), и результатов работы конкретных инструментов и скриптов OSINT. Этот анализ дополняет работу модуля enrichment_service и демонстрирует, как аналитик может проводить процесс OSINT.

Общая схема OSINT-исследования начинается с идентификации цели и обеспечения безопасности (OPSEC). Цель – деанонимизировать возможного владельца подозрительного аккаунта Telegram, связанного с конкретным инцидентом на панели мониторинга, найти его другие онлайн-следы и собрать дополнительную информацию о схеме мошенничества. Данные из таблицы «Список инцидентов» на рисунке 3.7 используются в качестве исходного источника информации. Для инцидента с идентификатором 68013072c498db4379aedffc в качестве отправной точки используются такие данные, как имя пользователя: danatello0o, идентификатор отправителя: 543452636 и номер телефона: 77076828477. Меры OPSEC включают в себя безопасную виртуальную среду, VPN/Tor, псевдонимы учётных записей и настройку браузера.

Первым этапом исследования является анализ и расширение исходных идентификаторов. На этом этапе сам Telegram ID может стать важной отправной точкой. Используя Telegram ID, можно получить определённую информацию напрямую через Telegram API или с помощью специальных инструментов. Основные типы информации, к которым можно получить доступ через Telegram ID, включают публичное имя пользователя, имя и фамилию, указанные в профиле, фотографии профиля, текст в разделе «Обо мне», а также время последнего входа в систему и номер телефона (если он не скрыт и доступен) в зависимости от настроек конфиденциальности пользователя. С помощью скрипта Telethon, представленного на рисунке 3.8, для user_id: 543452636 были получены следующие данные: имя пользователя: danatello0o, имя: Danabek, биографические данные: test fa danatelloo hacker... и телефон: 77076828477.

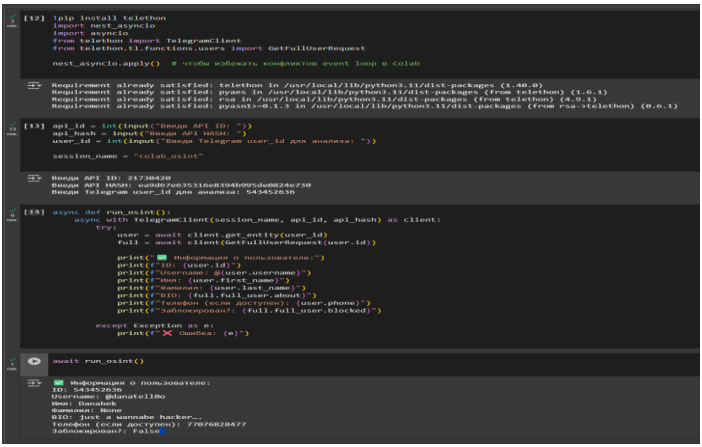

Рисунок 3.8 Использование библиотеки Telethon в среде Google Colab

В данной исследовательской работе описывается создание небольшого программного модуля для использования методов OSINT (Open-Source Intelligence) с использованием языка программирования Python и библиотеки Telethon для сбора открытых данных из мессенджера Telegram и получения базовой информации о пользователе. Представленный код адаптирован для выполнения в среде Google Colab и демонстрирует процесс получения доступной информации о пользователе из Telegram с использованием возможностей библиотеки Telethon.

В ходе работы сначала устанавливается библиотека Telethon, а модуль nest_asyncio используется для вызова функции nest_asyncio.apply() для разрешения конфликтов цикла событий в среде Google Colab. Этот шаг является необходимой технической мерой для корректного выполнения асинхронных операций в Colab.

В следующем разделе вводятся три параметра, необходимых для подключения к сервисам API Telegram: API ID, API HASH и user_id анализируемого пользователя. Эти параметры получаются с официального сайта Telegram https://my.telegram.org через личный кабинет и позволяют получить доступ к серверам Telegram через API. Это позволяет

пользователю авторизоваться и получать данные на законных основаниях. На этом этапе также указывается имя сессии. В примере имя сессии – «colab_osint». Имя сессии необходимо для установления постоянного соединения между сервером Telegram и клиентом Python.

Базовый процесс получения данных выполняется с помощью асинхронной функции. Функция Python async def run_osint() используется для установки асинхронного соединения с сервером Telegram. Асинхронное взаимодействие – эффективный способ избежать зависаний программы при работе с большими объёмами данных или ожидании ответа от сервера. Внутри функции подключение к Telegram осуществляется с помощью объекта TelegramClient, а методы get_entity и GetFullUserRequest используются для получения данных о пользователе. В то время как get_entity возвращает общую информацию о пользователе (идентификатор, имя пользователя и имя), GetFullUserRequest позволяет получить полный профиль пользователя, включая биографию, номер телефона (если доступен) и информацию о том, заблокирована ли учётная запись.

Вывод программы отображает следующую информацию о пользователе: идентификатор пользователя Telegram, имя пользователя, имя и фамилию, текст, введённый в разделе «О себе», номер телефона (если указан) и информацию о том, заблокирована ли учётная запись. Используемый блок try-except перехватывает любые ошибки во время выполнения, выводит сообщение об ошибке на экран и предотвращает сбой программы.

В приведенном примере полученная информация о конкретном пользователе выглядит следующим образом:

– Идентификатор телеграммы: 543452636
– Имя пользователя: danatelle80
– Имя: Данабек
– Биография: Просто начинающий хакер...
– Телефон: 77076828477
– Аккаунт заблокирован: Ложь

Этот результат показывает, что собирать информацию о пользователях через открытые данные в Telegram техниче-

ски просто и эффективно. В частности, если известны имя пользователя или его идентификационный номер, через открытый API Telegram можно получить доступ ко многим данным. Однако, согласно политике безопасности Telegram, если пользователь скрывает часть информации в своём профиле (например, номер телефона), эти данные могут стать недоступными.

Архитектура программы очень проста, но предоставляет хорошую возможность для практического освоения методов OSINT. Система может быть расширена для хранения полученных данных, сравнения с аккаунтами на других платформах, интеграции с инструментами автоматического сбора данных или деанонимизации с помощью ботов. Она также может служить основой для создания автоматизированной системы, позволяющей быстро выявлять опасные или подозрительные аккаунты в Telegram.

Приведённый в этом примере код Python – полезный инструмент для профессионалов и студентов, обучающихся работе с открытым исходным кодом. С его помощью можно изучить базовый алгоритм сбора информации о пользователях Telegram и эффективно использовать API Telegram. Кроме того, исследователи могут адаптировать этот код к системам сбора данных из других социальных сетей или мессенджеров.

В целом, данная программа, несмотря на свою компактность и простоту, считается эффективным стартовым инструментом для проведения исследований в области кибербезопасности, криминалистики и социальной инженерии. В перспективе данная система может быть развита в комплексную платформу для автоматического обнаружения и полного анализа мошеннических действий в Telegram. Например, практическую значимость системы можно повысить за счёт добавления таких функций, как автоматический анализ текстовых сообщений, обнаружение опасных ссылок, мониторинг активности ботов и построение графов общения между пользователями. Кроме того, гибкость языка Python и широкий функционал библиотеки Telethon позволяют посто-

янно обновлять и совершенствовать программы в этом направлении.

Однако с помощью Telegram ID невозможно напрямую узнать IP-адрес пользователя, его точное местоположение или содержание личных чатов; эту информацию можно получить только путём эксплуатации уязвимостей платформы или физического доступа к устройству, что является противозаконным. Таким образом, эффективность OSINT-анализа с использованием Telegram ID во многом зависит от наличия имени пользователя, присутствия исследователя и объекта исследования в общих чатах/группах, а также использования функций Telegram API.

При анализе пользователя qustust с помощью инструмента WhatsMyName.app в связи с идентификатором инцидента 6801446d66bc8d4e5b14ca8c в панели управления было обнаружено, что это имя пользователя зарегистрировано на нескольких популярных платформах. В частности, оно встречается на Habr, Steam, Telegram и Twitch. Столь обширное количество регистраций свидетельствует об онлайн-активности пользователя и о его устойчивом цифровом следе. Кроме того, характер зарегистрированных платформ указывает на его интерес к технологиям, программированию и игровой индустрии. Эти данные позволяют нам глубже понять профиль пользователя в ходе киберрасследований.

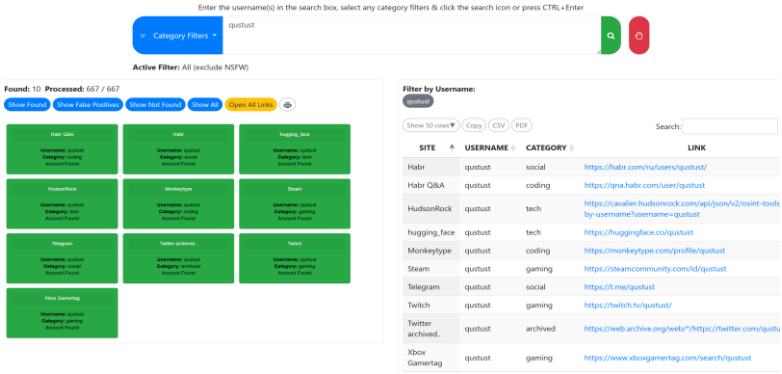

Рисунок 3.9 Результат WhatsMyName.app

Глубокий анализ найденных профилей и данных (SOCMINT). Профили пользователей Qustust тщательно проверяются, анализируются фотографии, публикации, список друзей/подписчиков и время активности. Также проверяются профиль пользователя danatello0o (Danabek) и данные, связанные с номером телефона 77076828477.

Анализ технической инфраструктуры. Если инциденты содержали ссылки на внешние веб-сайты, использовались методы анализа доменного имени, IP-адреса и веб-сайта.

Агрегируйте, анализируйте и суммируйте данные, связывая их с инцидентами на панели управления. Агрегированная информация о qustust указывает на то, что он является зарегистрированным пользователем на нескольких платформах, интересующимся в основном технологиями, программированием и играми. Возможные имена и фамилии (подтверждённые через Danabek Telethon) или другие связанные данные, а также профили, связанные с именем пользователя @danatello0o, дополняют запись об инциденте на панели управления.

Список инцидентов на рисунке 3.9 служит отправной точкой для расследования OSINT. Для каждого инцидента можно применить шаги описанной выше схемы OSINT, используя такие поля, как «Имя пользователя», «Идентификатор отправителя», «Номер телефона» (если доступен) и «Текст» (текст сообщения).

Этот практический анализ показывает, что методы OSINT – это не просто теоретические концепции, но и прикладной инструмент, позволяющий работать с реальными данными и глубже понимать ситуации мошенничества. Хотя модуль enrichment_service пытается автоматизировать многие из этих процессов, критическое мышление аналитика и его способность проводить дополнительные исследования всегда будут оставаться важными.

Заключение

Целью представленной в данном руководстве исследовательской работы была разработка комплексной программной системы, объединяющей методы обработки естественного языка, машинного обучения и OSINT для выявления мошеннических действий в мессенджере Telegram и деанонимизации подозрительных пользователей. В условиях современного информационного общества, наряду с глубоким проникновением в повседневную жизнь зашифрованных, анонимных средств коммуникации, расширяется сфера киберугроз и активно распространяются новые, более изощренные формы мошенничества. Платформа Telegram, в силу своих архитектурных особенностей, становится благоприятной средой для подобных киберпреступлений. Поэтому создание интеллектуальных систем, способных своевременно обнаруживать подобные угрозы и оперативно и эффективно реагировать на них, является одной из важнейших и актуальных задач сегодняшнего дня.

В результате проведённой научно-исследовательской работы был достигнут ряд важных научных и прикладных результатов. Во-первых, были всесторонне проанализированы основные виды и характер мошеннических действий на платформе Telegram. В ходе исследования были глубоко изучены наиболее распространённые мошеннические схемы в сети Telegram: социальная инженерия, инвестиционные мошенничества, фишинг, фейковые боты и фейковые лотереи. На научной основе систематизированы основные особенности этих форм мошенничества, используемые сценарии, используемые модели контента, каналы распространения и методы привлечения пользователей. Данный анализ позволил понять специфическую структуру мошеннических действий в сети Telegram и сформировать исходную информационную базу, необходимую для создания системы их автоматического выявления.

На следующем важном этапе научно-исследовательской работы была разработана система обработки и классификации сообщений Telegram, эффективно использующая современные методы в области обработки естественного языка и машинного обучения. В качестве основы данной системы выбрана модель преобразователя BERT (Bidirection Encoder Representations from Transformers). Модель BERT позволяет читать контекст текста в двух направлениях и глубоко понимать сложные языковые структуры. Кроме того, был выбран и использован ансамблевый классификатор Random Forest в качестве эффективного решения для высокоточной обработки текстовых данных и выявления признаков мошенничества. Система основана на принципе многосимвольной классификации, что позволяет одновременно выявлять несколько категорий мошенничества в одном сообщении. Это очень важно для эффективного выявления сложных и многоуровневых сценариев мошенничества в Telegram.

В качестве дополнения к системе был внедрен механизм деанонимизации с использованием информации из открытых источников. Поскольку пользователи Telegram зачастую анонимны, определить их истинную личность непросто. В связи с этим в систему был интегрирован модуль OSINT. Данный модуль был реализован с помощью richment_service. Используя имя пользователя, user_id, номер телефона и другие метаданные пользователя Telegram, были внедрены алгоритмы поиска дополнительной информации из открытых интернет-ресурсов. Используя открытые поисковые инструменты и платформы OSINT, такие как WhatsMyName, Maigret, Sherlock, стало возможным расширить цифровой след пользователей Telegram. Этот компонент расширил область применения системы и позволил использовать её не только как инструмент для выявления мошенничества, но и как инструмент деанонимизации, имеющий практическое значение для правоохранительных и следственных органов.

Общая архитектура системы разработана на основе микросервисных принципов в соответствии с требованиями современной разработки программного обеспечения. Все ключевые компоненты системы размещены в Docker-контейнерах. Это позволяет легко тестировать систему, быстро масштабировать и разворачивать её на различных аппаратных платформах. В состав системы входят модуль collect.py для сбора сообщений из Telegram, модуль text_processor.py для обработки текстовых данных, компонент threat_classifier.py для классификации мошенничества, модуль enrichment_service.py для поиска в открытых источниках и веб-интерфейс на базе фреймворка Flask для визуализации полученных результатов. Все сообщения и пользовательская информация, получаемые из Telegram, хранятся в NoSQL-базе данных MongoDB, что обеспечивает быстродействие системы и гибкие возможности обработки данных.

Для тестирования и оценки системы были проведены экспериментальные исследования. Был создан специальный набор данных на основе реальных сообщений, собранных с платформы Telegram. Все данные были предварительно размечены и обработаны. Текстовые сообщения были преобразованы в векторное представление с использованием вложений BERT и далее обработаны с помощью классификатора Random Forest. Эффективность и производительность системы в реальных условиях были протестированы с использованием ключевых метрик оценки, таких как точность, чувствительность, специфичность и F1-оценка. Экспериментальные результаты показали, что разработанная система способна обнаруживать мошеннические сообщения в Telegram с высокой точностью и эффективно разделять многоуровневые сценарии мошенничества. Кроме того, система была протестирована в нескольких практических сценариях обнаружения реального инвестиционного мошенничества, фишинговых атак и фальшивых ботов, и все ее модули доказали свою полную работоспособность.

Научная новизна полученных в ходе исследования результатов определяется созданием модели обнаружения мошенничества, адаптированной к специфике платформы Telegram, её способностью одинаково эффективно обрабатывать тексты на казахском и русском языках, возможностью выявления нескольких признаков мошенничества в одном сообщении с использованием метода многосимвольной классификации, возможностью вывести процесс деанонимизации на новый качественный уровень за счёт эффективной интеграции модуля OSINT в систему, а также созданием системы на модульной архитектуре на базе Docker. Практическая значимость работы подтверждается её практической применимостью в правоохранительных органах, подразделениях кибербезопасности, аналитических исследовательских центрах, образовательных организациях.

Дальнейшее совершенствование разрабатываемой системы может осуществляться по нескольким направлениям. Во-первых, необходимо разработать и внедрить специальные BERT или большие языковые модели (LLM), адаптированные к казахскому языку. Это позволит существенно повысить качество выявления мошеннического контента на казахском языке в Telegram. Во-вторых, одним из эффективных направлений является анализ структуры общения пользователей и оценка их потенциальной опасности на основе графовых нейронных сетей (GNN). В-третьих, необходимо расширить потенциал деанонимизации за счёт интеграции в систему новых OSINT-инструментов и платформ для сбора данных из современных открытых источников. Кроме того, улучшение интерфейса, внедрение интерактивных аналитических инструментов и интеграция с другими популярными мессенджерами расширят область применения системы и повысят её многофункциональность.

Для постоянного совершенствования системы и поддержания её эффективности необходимо внедрять стратегии MLOps. Такой подход позволяет постоянно обновлять модели и своевременно выявлять новые схемы мошенничества.

Использование методов MLOps оптимизирует жизненный цикл системы и обеспечивает её стабильную работу.

В целом, проведённое исследование и разработанный программный комплекс, описанные в настоящем руководстве, представляют собой конкретное, системное и перспективное решение, которое может быть использовано для борьбы с мошенничеством в мессенджере Telegram. Результаты работы могут стать важным методическим инструментом для образования, научных исследований и юридической сферы. Кроме того, данная разработка закладывает прочный фундамент для будущего развития интеллектуальных систем, автоматически выявляющих, анализирующих и своевременно реагирующих на мошеннические схемы в Telegram и других мессенджерах. Результаты исследования являются надёжной базой для будущих научных и прикладных проектов, направленных на укрепление кибербезопасности, повышение информационной культуры пользователей и оказание оперативной и эффективной поддержки правоохранительным органам.

Список использованных источников

1. Boyd D.M., & Ellison N.B. (2017). Social network sites: Definition, history, and scholarship. Journal of Computer-Mediated Communication, 13(1), – P. 210-230. URL: https://doi.org/10.1111/j.1083-6101.2007.00393.x

2. Federal Trade Commission. (2023). Consumer Sentinel Network Data Book. URL: https://www.ftc.gov/system/files/gov/pdf/csn-databook-2022.pdf

3. Europol. (2023). Internet Organised Crime Threat Assessment (IOCTA). URL: https://www.europol.europa.eu/publications-events/main-reports/internetorganised-crime-threat-assessment-iocta-2023

4. APWG (Anti-Phishing Working Group) (2019). Phishing Activity Trends Report Q4 2019. URL: https://docs.apwg.org/reports/apwg_trereport_q4_2023.pdf

5. Mitnick K.D., & Simon W.L. (2022). The Art of Deception: Controlling the Human Element of Security. Wiley Publishing, – P. 339-340.

6. Telegram Messenger. (2023). Telegram Reaches 700 Million Monthly Active Users. URL:https://telegram.org/blog/premium

7. Abu-Salma R., Krol K., Parkin S., Sasse A., & Bonneau J. (2017). Security analysis of account recovery in an end-to-end encrypted messaging service: A case study of Telegram's MTProto. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, – P. 1717-1730.

8. Kshetri N. (2017). The economics of «dark web» markets and the impact of law enforcement. Electronic Commerce Research and Applications, 26, – P. 45-55. URL: https://doi.org/10.1016/j.elerap.2017.09.002

9. Manning C.D., & Schütze H. (2019). Foundations of statistical natural language processing. MIT press, – P. 717.

10. Goodfellow I., Bengio Y., & Courville A. (2016). Deep learning. MIT press. Volume 19, – P. 305–307.

11. Devlin J., Chang M.W., Lee K., & Toutanova K.

(2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 – P. 4171-4186. URL: https://aclanthology.org/N19-1423

12. Bazzell M. (2021). Open Source Intelligence Techniques: Resources for Searching and Analyzing Online Information (8th ed.). IntelTechniques, – P. 575.

13. International Association of Law Enforcement Intelligence Analysts (IALEIA). Code of Ethics. URL: https://www.ialeia.org/ethics.php

14. Закон Республики Казахстан от 21 мая 2013 года №94-V «О персональных данных и их защите». Кодекс юстиции. URL:https://adilet.zan.kz/kaz/docs/Z1300000094

15. Bass L., Clements P., & Kazman R. (2022). Software Architecture in Practice (3rd ed.). Addison-Wesley Professional, – P. 673.

16. Newman S. (2019). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, – P. 294.

17. Merkel D. (2024). Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2024 (239), URL: https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment

18. Telegram API Documentation. Core API / Bot API. URL: https://core.telegram.org/api

19. Telethon: Pure Python MTProto API Library. Official Documentation. URL: https://docs.telethon.dev/

20. Russell M.A. (2023). Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More (2nd ed.). O'Reilly Media, – P. 358.

21. Aggarwal C.C., & Zhai C. (Eds.). (2022). Mining text data. Springer Science & Business Media, – P. 450.

22. Rajaraman A., & Ullman J.D. (2021). Mining of massive datasets. Cambridge University Press, – P. 218.

23. Devlin J., Chang M.W., Lee K., & Toutanova K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, – P. 4171-4186.

24. MongoDB, Inc. MongoDB Docs. URL: https://www.mongodb.com/docs/

25. Breiman L. (2021). Random forests. Machine learning, 45(1), – P. 5-32. URL: https://doi.org/10.1023/A:1010933404324

26. Tsoumakas G., & Katakis I. (2018). Multi-label classification: An overview. International Journal of Data Warehousing and Mining (IJDWM), 3(3), – P. 1-13.

27. Glassman M., & Kang M.J. (2022). The Duality of anOSINT (Open Source Intelligence): A Theoretical Review. Journal of Information Privacy and Security, 8(1), – P. 3-21.

28. Bazzell M. (2021). Open Source Intelligence Techniques: Resources for Searching and Analyzing Online Information (8th ed.). IntelTechniques, – P. 460.

29. Nissenbaum H. (2019). Privacy in context: Technology, policy, and the integrity of social life. Stanford University Press, – P. 377.

30. Chodorow K. (2023). MongoDB: The Definitive Guide (2nd ed.). O'Reilly Media. URL: https://www.mongodb.com/docs/

31. Few S. (2016). Information Dashboard Design: The Effective Visual Communication of Data. O'Reilly Media, – P. 358.

32. Grinberg M. (2018). Flask Web Development: Developing Web Applications with Python (2nd ed.). O'Reilly Media, – P. 455.

33. Cormen T.H., Leiserson C.E., Rivest R.L., & Stein C. (2019). Introduction to Algorithms (3rd ed.). MIT Press, – P. 440.

34. Docker Inc. Docker Overview. URL: https://docs.docker.com/get-started/overview/ (жүгінген күні 17.08.2024)

35. Merkel D. (2014). Docker: lightweight linux containers

for consistent development and deployment. Linux Journal, 2014, – P. 239.

36. Pustejovsky J., & Stubbs A. (2022). Natural language annotation for machine learning: a guide to corpus construction for linguistic analysis. O'Reilly Media, – P. 256.

37. Carneiro T., Medeiros Da Nobrega, R.V., Nepomuceno T., Bian G.B., De Albuquerque V.H.C., & Filho P.P.R. (2018). Performance analysis of Google Colaboratory as a tool for accelerating deep learning applications. IEEE Access, 6, 61677-61685. URL: https://doi.org/10.1109/ACCESS.2018.2874767

38. Bergstra J., & Bengio Y. (2022). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), – P. 281-305. URL: https://doi.org/10.1016/j.patrec.2005.10.010

39. Fawcett T. (2016). An introduction to ROC analysis. Pattern recognition letters, 27(8), – P. 861-874. URL: https://doi.org/10.1016/j.patrec.2005.10.010

40. Tsoumakas G., & Katakis I. (2017). Multi-label classification: An overview. International Journal of Data Warehousing and Mining (IJDWM), 3(3), – P. 1-13.

41. Bazzell (2021): Open Source Intelligence Techniques: Resources for Searching and Analyzing Online Information (8th ed.). IntelTechniques, – P. 474.

42. Telegram Law Enforcement Guidelines. URL: https://telegram.org/faq#q-how-do-i-contact-telegram (жүгінген күні 08.07.2024)

43. Myers G.J., Sandler C., & Badgett T. (2021). The art of software testing (3rd ed.). John Wiley & Sons, – P. 294.

44. Chandola V., Banerjee A., & Kumar V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), – P. 1-58.

45. Terizi C., Chatzakou D., Pitoura E., Tsaparas P., & Kourtellis N. (2021). Modeling aggression propagation on social media. Online Social Networks and Media, 24, 100137. URL: https://doi.org/10.1016/j.osnem.2021.100137

СОДЕРЖАНИЕ

Введение 3

1. МОШЕННИЧЕСТВО В СОЦИАЛЬНЫХ СЕТЯХ И ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ЕГО ВЫЯВЛЕНИЯ

1.1. Мошенничество в социальных сетях: распространенность, виды и особенности..... 6

1.2. Анализ возможностей платформы Telegram и ее потенциала для мошенничества..... 21

1.3. Обзор методов обработки естественного языка (NLP) и машинного обучения (ML) для обнаружения мошеннических сообщений..... 24

1.4. Роль методов разведки с открытыми источниками (OSINT) в деанонимизации: теоретические основы..... 30

2. РАЗРАБОТКА СИСТЕМЫ ОБНАРУЖЕНИЯ И ДЕАНОМИНАЦИИ МОШЕННИЧЕСТВА В TELEGRAM

2.1. Архитектура системы и взаимодействие модулей 35

2.2. Реализация модуля сбора данных с использованием API Telegram..... 52

2.3. Конвейер предварительной обработки собранных текстовых данных 58

2.4. Создание и адаптация моделей BERT и Random Forest для классификации мошеннических сообщений..... 62

2.5. Интеграция модуля OSINT для деанонимизации пользователей..... 69

2.6. Система управления данными (MongoDB) и панель визуализации результатов (Dashboard)..... 73

2.7. Алгоритм работы разработанной системы и ее развертывание в Docker-контейнере 78

3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ РАЗРАБОТАННОЙ СИСТЕМЫ И АНАЛИЗ РЕЗУЛЬТАТОВ

3.1. Создание набора данных для эксперимента: сбор, очистка и маркировка 91

3.2. Процесс обучения и оптимизации параметров моделей классификации (BERT, Random Forest)..... 95

3.3. Оценка эффективности модели: метрики, сравнительный анализ и интерпретация результатов 99

3.4. Практическая работа модуля OSINT: оценка возможностей деанонимизации на примере реального инцидента 102

3.5. Демонстрация комплексной работы системы: сценарии от обнаружения мошенничества до деанонимизации 110

3.6. Статистический анализ и визуализация обнаруженных угроз: функции панели мониторинга 114

3.7. Практическое применение исследований OSINT: анализ на основе панели мониторинга 117

Заключение 124

Список использованных источников 129

Верстка:
Абайдельдинова Ж.Т.

Отдел организации научно-исследовательской и редакционно-издательской
работы Алматинской академии МВД Республики Казахстан
имени М. Есбулатова 050060 Алматы, ул. Утепова, 29

Подписано в печать 17 ноября 2025 года.
Формат 60х84 1/16. Бум. тип. №1. Печать на ризографе. Уч.-изд. 7,8 п.л.
Тираж 100 экз.