УДК: 004.056

Черкесова Л.В., Профессор,

Доктор физико-математических наук,

кафедра «Кибербезопасность информационных систем»

ФГБОУ ВО ДГТУ

Россия, г. Ростов-на-Дону

Суслов А.П. магистрант, 2 года обучения

кафедра «Кибербезопасность информационных систем»

факультет «Информатика и вычислительная техника»

ФГБОУ ВО ДГТУ

Россия, г. Ростов-на-Дону

Д.А. Бондарев студент 6 Курс

кафедра «Кибербезопасность информационных систем»

факультет «Информатика и вычислительная техника»

МЕХАНИЗМ ЗАЩИТЫ ОТ ФИШИНГОВЫХ АТАК ТИПА КВИШИНГ, РЕАЛИЗОВАННЫЙ НА ПЛАТФОРМЕ OS ANDROID

Аннотация: *Статья посвящена исследованию и разработке мобильного приложения для платформы Android, предназначенного для обнаружения и предотвращения фишинговых атак типа квишинг (QR-фишинг). Проводится анализ особенностей кибератак, основанных на использовании QR-кодов, рассматриваются современные методы защиты и предлагается собственное решение, обеспечивающее автоматический анализ безопасности ссылок. Разработанный прототип реализует многоуровневый анализ URL, включающий сетевые, лексические проверки и взаимодействие с внешними сервисами репутации. Тестирование подтвердило эффективность предложенного подхода в распознавании опасных ссылок.*

Ключевые слова: *квишинг, фишинг, QR-код, Android, информационная безопасность, анализ URL.*

A QUISHING-TYPE PHISHING PROTECTION MECHANISM IMPLEMENTED ON THE ANDROID OS PLATFORM

Annotation: *The article is devoted to the research and development of a mobile application for the Android platform, designed to detect and prevent quishing attacks (QR-phishing). The analysis of the features of cyberattacks based on the use of QR codes is carried out, modern protection methods are considered, and our own solution is proposed that provides automatic analysis of link security. The developed prototype implements a multi-level URL analysis, including network, lexical checks and interaction with external reputation services. Testing confirmed the effectiveness of the proposed approach in recognizing dangerous links.*

Key words: *quishing, phishing, QR code, Android, information security, URL analysis.*

Ведение

Феномен фишинга в XXI веке проявляется в разнообразных формах, адаптируясь под новые технические и поведенческие контексты. Одним из заметных трендов последних лет стало использование QR-кодов в качестве носителя вредоносных ссылок — явление, получившее название «квишинг». QR-коды, изначально разработанные для облегчения доступа к информации, оказались удобным инструментом социальной инженерии: визуально нейтральный символ позволяет злоумышленнику скрыть целевой URL от поведенческой проверки пользователя. По данным отраслевых отчётов, количество фишинговых инцидентов сохраняется на высоком уровне, а случаи использования QR-кодов наглядно демонстрируют тенденцию к диверсификации векторов атак [1,2].

Технологические основы QR-кодов и риски безопасности**.**

QR-коды (Quick Response Codes) — это разновидность двумерных штрихкодов, способных хранить данные в горизонтальном и вертикальном направлениях. Они были разработаны в 1994 году японской компанией Denso Wave для маркировки автокомпонентов. Главное преимущество QR-кодов по сравнению с классическими штрихкодами — возможность быстрой расшифровки большого объёма информации при помощи камеры мобильного устройства.

Структура QR-кода включает в себя:

- зоны синхронизации и маркеры позиционирования;
- область кодирования данных, где записана полезная информация;
- зоны коррекции ошибок, реализованные по алгоритму Рида-Соломона.

Рис. 1 – Устройство QR-кода

QR-коды способны хранить:

- текст (до 4 296 символов ASCII);
- URL-ссылки;
- контакты (vCard);
- геолокацию;
- данные для Wi-Fi-подключения и многое другое.

Изначально QR-коды применялись в производственной логистике, но сегодня они широко используются в:

– платёжных системах (например, СБП в России, Alipay в Китае);
– транспорте (электронные билеты);
– медицине (идентификация пациентов и лекарств);
– маркетинге (переход на сайт, купоны, бонусные программы).

Повсеместное распространение QR-кодов стало возможным благодаря удобству генерации и сканирования: пользователи могут создать код через веб-сервисы, а смартфоны могут распознать его мгновенно. Однако именно это удобство и визуальная непрозрачность данных создают предпосылки для злоупотреблений и кибератак.

Структура и принцип работы QR-кода. Процесс формирования QR-кода включает в себя несколько последовательных и строго регламентированных этапов, определённых стандартом ISO/IEC 18004. Каждый этап направлен на обеспечение корректного кодирования информации, её защиты и удобства считывания. Ниже приведены основные стадии генерации QR-кода:

1. Кодирование исходных данных

   На этом этапе информация (текст, URL, байты и т.д.) преобразуется в бинарную последовательность в соответствии с выбранным режимом кодирования (цифровой, алфавитно-цифровой, байтовый и др.). В начало добавляется заголовок — идентификатор режима и длина сообщения.

2. Добавление служебной информации и данных заполнения

   При необходимости в конец бинарной последовательности добавляются специальные служебные биты (например, «конец данных») и заполняющие биты (pad bits), чтобы довести длину до необходимого размера, соответствующего выбранной версии QR-кода.

3. Разделение на блоки данных

   Полученная последовательность разбивается на один или несколько блоков, что позволяет параллельно добавлять к каждому блоку собственный набор байтов коррекции. Такое разбиение также повышает устойчивость к локальным повреждениям.

4. Генерация байтов коррекции ошибок

С использованием алгоритма Рида–Соломона для каждого блока вычисляются контрольные байты. Их количество зависит от выбранного уровня коррекции ошибок (L, M, Q, H) и определяет, насколько QR-код сможет восстановиться при частичном повреждении.

5. Объединение блоков в единую структуру

   Информационные и коррекционные блоки объединяются в единую чередующуюся последовательность, в соответствии с заданным стандартом. Это обеспечивает равномерное распределение полезной информации по всей площади QR-кода.

6. Размещение битов данных в матрице QR-кода

   Финальная последовательность размещается в графической структуре QR-кода по строго определённой схеме, с учётом позиционных маркеров, выравнивающих узоров, синхронизирующих линий и других служебных элементов. Также на этом этапе применяется одна из восьми масок для минимизации визуальных помех (слишком длинные ряды чёрных или белых модулей).

QR-коды поддерживают несколько режимов кодирования, каждый из которых предназначен для определённого типа данных:

- Цифровой режим — используется для кодирования только цифр (0–9), наиболее эффективный по плотности.
- Алфавитно-цифровой режим — кодирует цифры, латинские буквы (A–Z), пробел и 9 специальных символов.
- Байтовый режим — предназначен для кодирования произвольных байтов (например, UTF-8 или ISO 8859-1).
- Kanji — режим для кодирования японских иероглифов (использует систему Shift JIS).
- ECI (Extended Channel Interpretation) — позволяет задать нестандартную кодировку, например, для кириллицы.

QR-коды имеют 40 версий, каждая из которых определяет размер матрицы и объём информации, который можно закодировать. Размер QR-кода

увеличивается по формуле: Размер (в модулях) = 21 + 4 × (номер версии – 1). Так, версия 1 состоит из 21×21 модулей, версия 2 — 25×25, и так далее вплоть до версии 40 — 177×177 модулей. С увеличением версии возрастает и максимальное количество символов, которое может быть закодировано. На ёмкость также влияет выбранный режим кодирования (цифровой, текстовый, байтовый и др.) и уровень избыточности данных — то есть объём резервной информации, предназначенной для восстановления повреждённых участков кода. Для обеспечения надёжности QR-коды используют алгоритм Рида–Соломона, позволяющий восстановить данные при частичном повреждении изображения. Пользователь может выбрать один из четырёх уровней коррекции:

- L (Low) — восстановление до 7 % данных;
- M (Medium) — до 15 %;
- Q (Quartile) — до 25 %;
- H (High) — до 30 %.

Чем выше уровень коррекции, тем надёжнее QR-код при физическом повреждении или искажении изображения, но тем меньше становится объём полезных данных, поскольку часть пространства занимает избыточная информация. Пример зависимости:

Версия 1, уровень L — до 41 цифры.

Версия 1, уровень H — уже только 17 цифр.

Версия 10, уровень L — до 274 цифр.

Версия 10, уровень H — около 122 цифр.

Для наглядного понимания структуры QR-кода рассмотрим пример, в котором зашифрована ссылка на официальный сайт Донского государственного технического университета — https://donstu.ru.

На рисунке ниже представлен соответствующий QR-код:

Рис. 2 – QR-код со ссылкой на сайт Донского государственного технического университета

После сканирования кода смартфон или специализированное ПО декодирует его и отображает следующую строку: https://donstu.ru. Это означает, что QR-код использует байтовый режим кодирования и содержит URL в стандартной форме. Ниже приведён разбор компонентов кода:

- Режим кодирования: байтовый (UTF-8);
- Версия QR-кода: автоматически выбрана версия 2 (25×25), так как информации немного;
- Уровень коррекции ошибок: M (15%), как оптимальный для печати и экранов;
- Форматная информация: содержит сведения о маске и уровне коррекции;
- Позиционные и синхронизирующие маркеры: позволяют сканеру быстро определить ориентацию кода;
- Поле данных: содержит строку https://donstu.ru в бинарной форме;
- Маска: применяется автоматически, чтобы избежать повторяющихся паттернов и упростить считывание.

Для практики безопасности важны следующие технические аспекты QR-кодов. Во-первых, QR-код не различает «тип» данных — последовательность байтов может представлять собой URI, произвольный текст или даже data-URI с включённым HTML. Во-вторых, длинные URL, содержащие параметры и токены, легко помещаются в QR-кодах благодаря высокой плотности

кодирования. В-третьих, QR-коды часто используются в офлайн-среде, где пользователь не располагает дополнительной информацией о происхождении кода. С точки зрения атакующих техник, наиболее распространённые приёмы включают: использование коротких перенаправляющих сервисов (bit.ly, t.co), внедрение промежуточных редиректов через легитимные домены, применение Punycode (xn--) для визуальной подмены символов, и обфускация через base64/URL-encoding в параметрах. Комбинация этих приёмов делает автоматическое обнаружение сложной задачей и требует многоуровневого подхода.

Постановка задачи и цели исследования. Целью исследования является разработка и внедрение мобильного приложения для платформы Android, обеспечивающего анализ и предотвращение фишинговых атак, реализуемых посредством QR-кодов (квишинг). Основная идея заключается в создании программного средства, которое автоматически выполняет проверку URL, полученных из QR-кодов, на наличие признаков фишинговой активности и информирует пользователя о потенциальной угрозе до перехода по ссылке.

Для достижения поставленной цели необходимо решить следующие задачи: Провести анализ существующих подходов к обнаружению и предотвращению фишинговых атак, в том числе атак, основанных на QR-кодах, а также выявить их сильные и слабые стороны; Разработать архитектуру мобильного приложения, обеспечивающего многослойный анализ URL, включающий сетевые, лексические и контекстные проверки; Реализовать модули проверки безопасности ссылок с применением локальных эвристических фильтров и паттернов опасных признаков, анализа HTTPS-поддержки и корректности SSL-сертификатов, взаимодействия с внешними сервисами репутации (Google Safe Browsing, VirusTotal); Обеспечить асинхронное взаимодействие компонентов приложения с целью минимизации задержек при анализе и сохранения отзывчивости пользовательского интерфейса, Разработать пользовательский интерфейс, позволяющий сканировать QR-коды через камеру устройства, импортировать изображения

из галереи или вводить ссылки вручную; Провести тестирование разработанного решения, оценив точность и скорость анализа URL, а также определить долю корректно идентифицированных фишинговых ссылок; Сформулировать рекомендации по дальнейшему развитию системы, включая интеграцию с облачными системами мониторинга и расширение базы эвристических признаков.

Решение указанных задач позволит создать мобильное программное средство, способное в реальном времени выполнять комплексную оценку безопасности QR-ссылок и снижать вероятность успешных атак типа квишинг.

Методы исследования. В этом разделе последовательно рассматриваются применяемые в работе методы, начиная с анализа существующих подходов к обнаружению и предотвращению фишинговых атак (включая атаки, основанные на QR-кодах), и завершая обоснованием выбора набора инструментов для прототипа. Сначала приведён подробный анализ аналогов — их функциональности, сильных и слабых сторон — что позволяет сформулировать требования к разрабатываемому приложению.

Анализ существующих подходов и решений. За последние несколько лет наблюдается устойчивый рост URL-ориентированных фишинговых кампаний, при этом доля атак с использованием QR-кодов (so-called «quishing») заметно возросла: аналитические отчёты фиксируют миллионы атак и сотни тысяч кампаний, где злоумышленники внедряют QR-коды в письма, PDF-документы и наружную рекламу, чтобы перенаправлять пользователей на фишинговые страницы. Это делает QR-коды привлекательным вектором социальной инженерии, потому что пользователь не видит URL до момента перехода [3,4]. Анализ литературы и практических отчётов показывает, что существующие подходы к обнаружению фишинга в целом применяют комбинацию следующих стратегий:

- Сигнатурно-репутационные проверки — сопоставление URL/доменов с черными списками и сервисами репутации (Google Safe Browsing, VirusTotal, PhishTank и пр.). Эти механизмы быстры и хорошо работают для известных

угроз, но ограничены при нулеводневных (zero-day) атаках и при использовании прокси/редиректов [5].

- Эвристический и лексический анализ — проверка структуры URL (наличие IP-адреса вместо имени домена, подозрительные расширения, длинные строковые параметры, Punycode, явные ключевые слова типа «login», «verify» и т.д.). Эвристики гибки, но дают много ложных срабатываний на легитимных случаях (support-страницы, временные токены) [6,7].
- Сетевая проба (HTTPS / сертификаты / редиректы) — HEAD/GET-пробы для определения поддержки HTTPS, корректности TLS и конечного URL после цепочки редиректов. Это даёт сильный сигнал о качественности ресурса, но подвержено задержкам и ограничениям сетевой среды (ограниченный трафик, блоки, таймауты) [8].
- Динамический анализ контента — загрузка страницы, анализ DOM/скриптов, эмуляция выполнения JavaScript и проверка признаков фишинговых форм. Даёт хорошие показатели для сложных страниц, но требует ресурсов и часто невозможен на клиенте (мобильном) без серверной инфраструктуры.
- ML/статистические методы — обучение моделей на признаках URL и содержимого (n-grams, структурные признаки, признаки хоста). Может выявлять новые паттерны, но требует репрезентативных данных и мониторинга дрейфа распределений; также рискует переобучением и объяснимостью решений (black-box).

Изученные продукты и исследования показывают, что комбинированный (многоуровневый) подход обеспечивает наилучший компромисс между точностью и практичностью: локальные эвристики дают немедленный отклик, репутационные API — высокую точность для известных угроз, а при необходимости — динамический/серверный анализ для спорных случаев. Однако у большинства существующих мобильных решений (сканеры от вендоров, простые open-source проекты) наблюдаются типичные ограничения: неполный анализ редиректов, слабая интеграция с

API репутации, ограниченная поддержка нетипичных URI (intent:, data:, tel:) и низкая прозрачность выводимых предупреждений.

Ключевые сильные и слабые стороны существующих подходов

Сильные стороны:

- Быстрая проверка известных угроз при помощи репутационных баз (низкая латентность на клиенте при использовании локального кэша).
- Эвристики легко реализуются и дают сигналы в оффлайн-режиме (ограниченная сеть).
- Комбинация нескольких источников повышает устойчивость к обфускации (shorteners, Punycode, base64).

Слабые стороны:

- Зависимость от внешних сервисов — платёжные ограничения, квоты, приватность и задержки; это критично для мобильных приложений, ориентированных на быстрый UX.
- Ложные срабатывания (FP) — эвристики часто отмечают легитимные страницы как подозрительные (например, страницы сброса пароля, платежные шлюзы с параметрами).
- Ограничения клиентской проверки — мобильное устройство ограничено по ресурсам для глубокой динамической эмуляции и безопасного исполнения JS.
- Проблемы с прозрачностью — многие приложения не объясняют пользователю, почему URL оценён как опасный, что ухудшает доверие и приводит к игнорированию предупреждений.

Анализ существующих решений. Современные программные средства, предназначенные для проверки безопасности QR-кодов, как правило, обеспечивают лишь базовую функциональность, ограничиваясь декодированием содержимого и проверкой URL по репутационным базам данных. Проведённый анализ показал, что несмотря на устойчивый рост числа фишинговых атак, включая квишинг (QR-фишинг), специализированных решений, реализующих комплексную проверку QR-ссылок, по-прежнему крайне мало [1–3].

Большинство существующих инструментов выполняют только первичную проверку URL-адресов на предмет наличия в них признаков вредоносной активности и обращаются к облачным репутационным базам, таким как Google Safe Browsing или VirusTotal [4–5, 8]. Такой подход эффективен при выявлении известных угроз, однако он мало пригоден для обнаружения новых, целевых и замаскированных атак. Использование исключительно сигнатурных и репутационных методов ограничивает возможности систем при столкновении с ранее не зарегистрированными доменами и динамическими редиректами [6].

Согласно отчётам APWG и Kaspersky, в 2025 году доля атак, использующих QR-коды, выросла более чем на 60 % по сравнению с предыдущим годом, а количество кампаний с внедрением QR-кодов в электронную почту и печатные материалы продолжает увеличиваться [1–2]. Исследования ReliaQuest и IBM X-Force подтверждают, что злоумышленники активно используют короткие URL-сервисы, подмену символов (Punycode) и цепочки промежуточных редиректов для обхода фильтров. Эти приёмы значительно затрудняют автоматический анализ и требуют многоуровневого подхода, включающего лексический, сетевой и контекстный анализ.

Отдельные мобильные приложения обеспечивают отображение URL до перехода и выполняют поверхностную проверку через внутренние базы данных, однако они не проводят углублённую диагностику, не анализируют структуру перенаправлений и редко используют асинхронные механизмы взаимодействия с внешними API [4, 5, 8]. При этом многие решения не поддерживают анализ нетипичных URI-схем (таких как intent://, tel:, data:), что позволяет злоумышленникам скрывать вредоносное содержимое от стандартных фильтров.

Серьёзной проблемой также является низкая прозрачность результатов анализа: пользователю, как правило, не объясняется, почему ссылка классифицирована как безопасная или подозрительная. Отсутствие контекстных пояснений снижает доверие и приводит к тому, что пользователи

игнорируют предупреждения, воспринимая их как ложные срабатывания [6, 7]. Кроме того, ряд популярных решений демонстрируют высокую скорость работы за счёт упрощения логики анализа, отказа от многопоточных или асинхронных запросов и исключения дополнительных эвристических фильтров. Это снижает вычислительную нагрузку, но существенно ограничивает способность систем к обнаружению сложных схем фишинга, особенно при использовании цепочек редиректов и динамически подгружаемого контента [9].

Обобщая результаты проведённого анализа, можно выделить несколько ключевых проблем современных решений:

- отсутствие глубокого анализа URL-структуры и цепочек редиректов;
- ограниченное использование внешних репутационных сервисов;
- низкая интерпретируемость выводимых результатов;
- зависимость от сетевого подключения при отсутствии офлайн-эвристик;
- отсутствие регулярного обновления сигнатур и эвристических баз [3–5, 8].

Таким образом, существующие средства в основном ориентированы на поверхностные проверки и не обеспечивают комплексного анализа безопасности QR-ссылок. Это обосновывает необходимость разработки мобильного приложения, реализующего многоуровневую методологию анализа — сочетание локальных эвристик, сетевых проверок и асинхронного взаимодействия с репутационными системами. Такой подход обеспечивает баланс между скоростью обработки и полнотой детектирования угроз, что особенно важно в условиях мобильных ограничений.

Проведённый анализ показал, что существующие программные средства ориентированы преимущественно на поверхностные проверки QR-ссылок и не обеспечивают комплексного анализа угроз. Для повышения эффективности защиты целесообразно реализовать гибридный подход, сочетающий локальные эвристики и обращения к внешним репутационным сервисам.

Основное внимание при проектировании системы следует уделить приоритизации и интерпретации результатов анализа, чтобы пользователь мог

понимать причину классификации ссылки и принимать осознанное решение. В условиях ограниченных ресурсов мобильных устройств важна асинхронная архитектура, обеспечивающая параллельную обработку сетевых запросов без снижения отзывчивости интерфейса.

Таким образом, результаты анализа обосновали выбор методологии разработки — многоуровневую структуру приложения, включающую эвристический, сетевой и репутационный уровни анализа. Такой подход позволяет достичь баланса между скоростью проверки и полнотой выявления фишинговых угроз.

Архитектурная структура приложения. Создаваемое приложение разработано на языке Kotlin в среде Android Studio с использованием принципов модульного проектирования и разделения ответственности. Архитектура включает три ключевых уровня:

- Пользовательский интерфейс (UI Layer) — обеспечивает взаимодействие пользователя с системой, предоставляет функции сканирования QR-кодов, импорта изображений и ручного ввода ссылок. Интерфейс построен с использованием Material Design-компонентов, что обеспечивает интуитивное восприятие и адаптацию под разные устройства.
- Логический модуль анализа (Logic Layer) — отвечает за выполнение проверок URL. Центральным элементом является класс QrSecurityChecker, реализующий последовательный и параллельный вызов функций анализа: эвристических, сетевых и репутационных. Данный модуль агрегирует результаты в структурированном отчёте, вычисляя итоговый уровень угрозы.
- Сетевой модуль (Network Layer) — реализует асинхронное взаимодействие с внешними API, включая Google Safe Browsing, VirusTotal и внутренние HTTPS-проверки. Для параллельной обработки запросов используются Kotlin Coroutines, что обеспечивает отзывчивость приложения даже при высокой сетевой задержке.

Взаимодействие между компонентами осуществляется по принципу “observer–observable”: при поступлении новых данных логический модуль уведомляет UI о результатах анализа, обновляя визуализацию уровня угрозы.

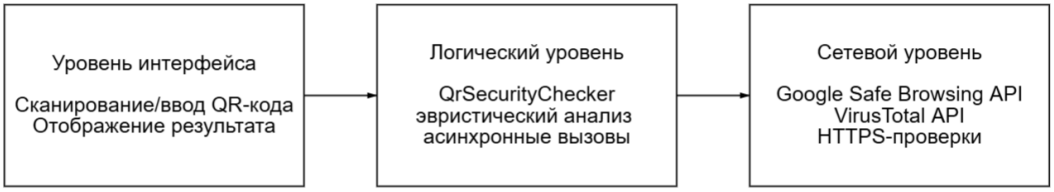


Рис. 3 — Архитектурная схема мобильного приложения.

Методы анализа URL и выявления фишинговых признаков. Система реализует трёхуровневую модель анализа, включающую локальные, сетевые и репутационные проверки.

1. Локальный эвристический анализ

   На первом этапе производится синтаксическая и лексическая оценка URL. Реализованы следующие проверки: наличие IP-адреса вместо доменного имени; избыточная длина строки запроса и количество параметров; присутствие подозрительных подстрок (login, update, verify, confirm, secure, password); наличие кодированных символов (%xx, base64) и Punycode-доменов (xn--).

Эвристическая модель основана на правилах, описанных в литературе по фишинг-детекции, и позволяет проводить офлайн-анализ без обращения к сети.

2. Сетевой анализ (Network probe)

   На втором уровне выполняется активная проверка URL посредством HTTPS-пробы. Приложение отправляет HEAD- или GET-запрос к целевому адресу, регистрируя: поддержку HTTPS и корректность SSL-сертификата;

наличие или отсутствие редиректов (с фиксированием финального домена); код ответа сервера и задержку отклика.

Обнаруженные ошибки типа SSLHandshakeException или несоответствие сертификата классифицируются как признаки потенциального риска. Дополнительно анализируются метаданные ответа (Server, Location, Content-Type) для определения подозрительных шаблонов поведения.

3. Интеграция с внешними системами репутации

На третьем уровне выполняется взаимодействие с VirusTotal API и Google Safe Browsing API. Для VirusTotal реализован запрос GET /urls/{id} после регистрации URL, анализируется поле last_analysis_results, где учитываются обе категории — category и result. В Google Safe Browsing передаётся список проверяемых URL в формате JSON-запроса; ответ классифицируется по типу угрозы: phishing, malware, unwanted software. Полученные результаты агрегируются в сводный отчёт, где каждой угрозе присваивается уровень критичности.

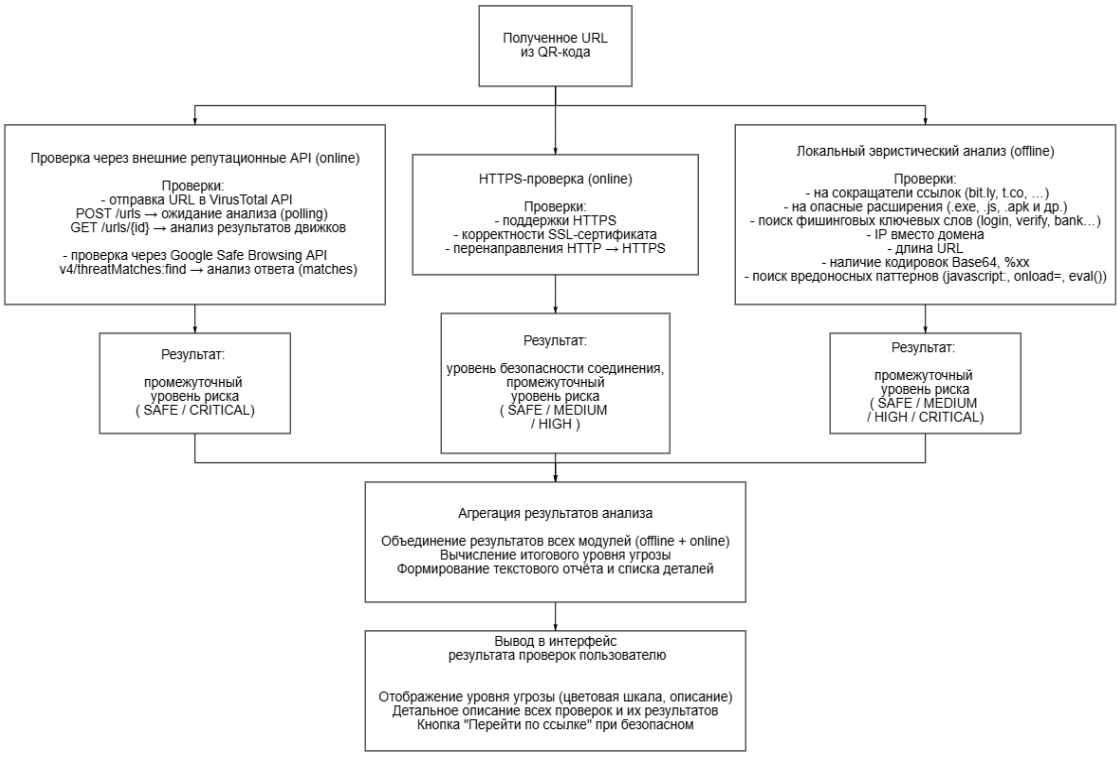


Рис. 4 – Блок-схема алгоритма анализа URL в мобильном приложении.

Алгоритм агрегирования и оценки риска. После выполнения всех проверок система объединяет результаты, формируя интегральную оценку риска.

Финальный уровень угрозы отображается пользователю в виде цветовой шкалы и текстового обозначения (зелёный – безопасно, жёлтый – сомнительно, красный – опасно/критично), сопровождаемой кратким пояснением причины оценки.

Технологическая реализация и особенности. Использование асинхронных корутин Kotlin позволило минимизировать задержки при взаимодействии с API и повысить отзывчивость интерфейса. В архитектуре отсутствуют блокирующие вызовы, что особенно важно для мобильных устройств с ограниченными ресурсами. Для обеспечения устойчивости и безопасности при сетевых операциях реализованы следующие меры:

- централизованная обработка исключений (try-catch) с журналированием ошибок в локальный лог;
- ограничение числа параллельных сетевых потоков;
- тайм-ауты для внешних запросов (до 60 с);
- контроль подлинности сертификата осуществляется встроенными средствами Android при установлении HTTPS-соединения; результаты соединения используются в качестве индикатора доверия к домену.

При запуске мобильного приложения пользователю отображается главное меню, представленное на рис. 5. В нём реализованы три основных действия:

- сканирование QR-кода с помощью камеры устройства;
- загрузка QR-кода из галереи;
- ручной ввод ссылки для анализа.

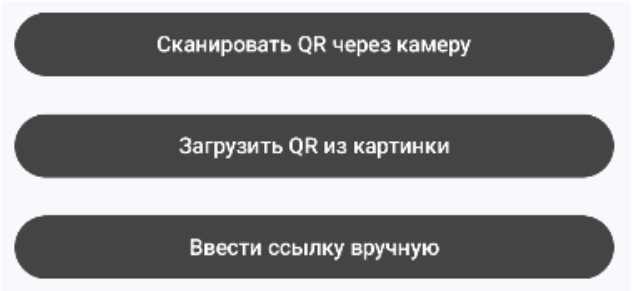


Рис. 5 – Главное меню приложения

После выбора режима ручного ввода или загрузки QR-кода открывается экран с отображением полученной ссылки (рис. 6). На данном этапе пользователь может убедиться, что распознанный адрес корректен и действительно соответствует ожидаемому ресурсу.

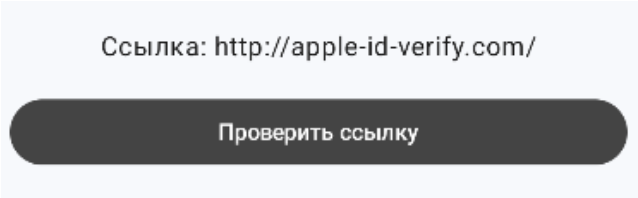


Рис. 6 – Окно с введённой ссылкой

После завершения анализа пользователю показывается итоговое окно с результатом проверки (рис. 7). На экране отображается общий уровень угрозы, классифицированный по цветовой шкале (зелёный — безопасно, жёлтый — сомнительно, красный — опасно), а также краткое описание выявленных признаков.

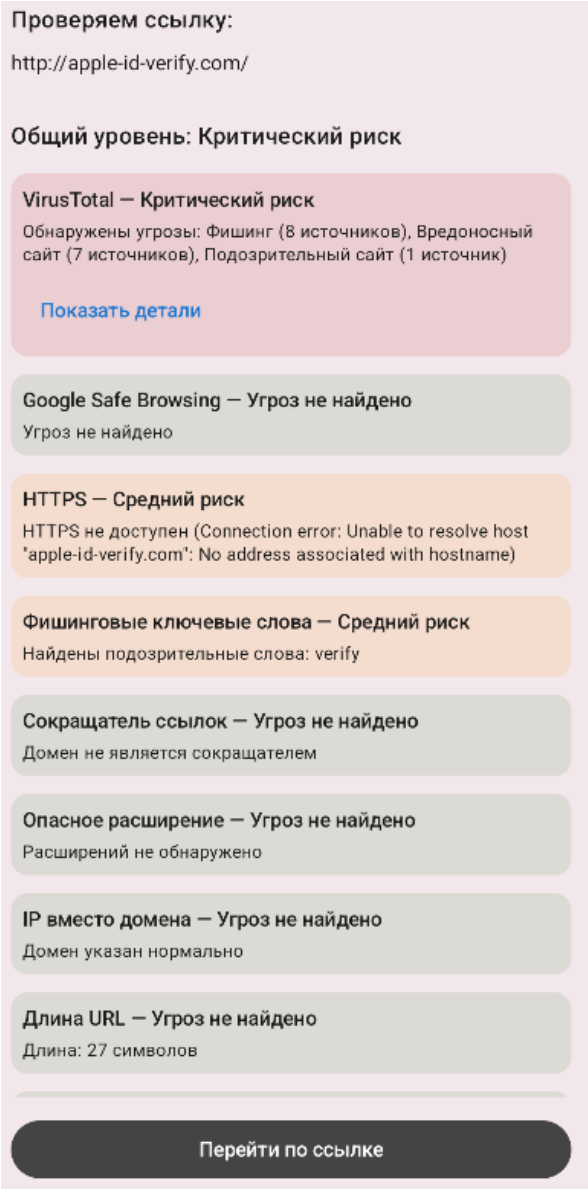
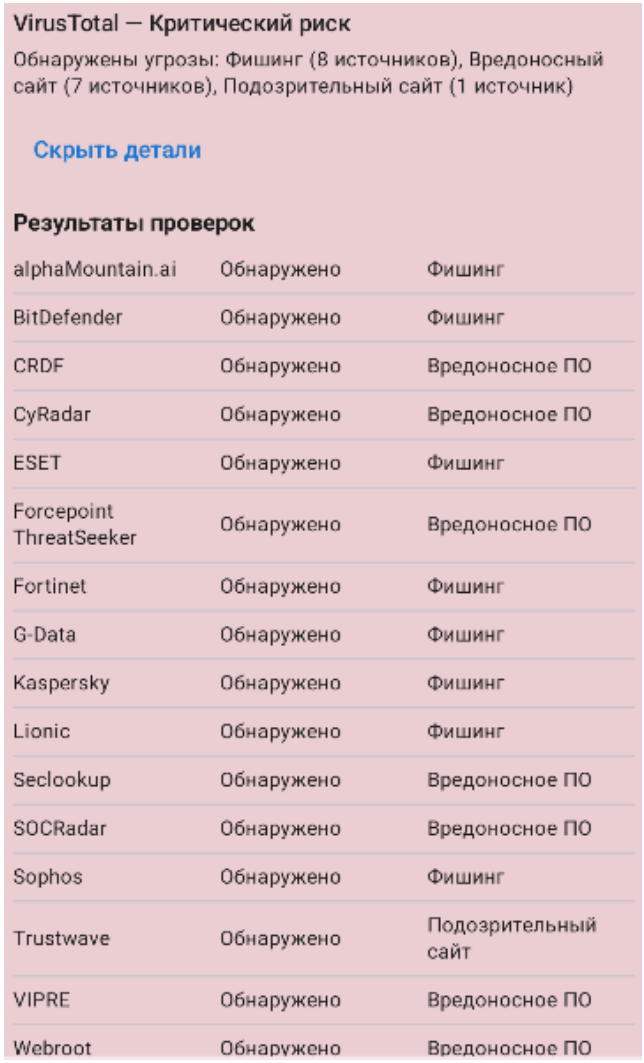


Рис. 7 – Результат анализа ссылки

При необходимости пользователь может развернуть подробный отчёт (рис. 8), где показаны детальные результаты проверки через внешние сервисы. На примере VirusTotal видно, какие аналитические движки и сервисы репутации выявили угрозу и какой вердикт они выдали.



VirusTotal — Критический риск

Обнаружены угрозы: Фишинг (8 источников), Вредоносный сайт (7 источников), Подозрительный сайт (1 источник)

Скрыть детали

Результаты проверок

alphaMountain.ai	Обнаружено	Фишинг
BitDefender	Обнаружено	Фишинг
CRDF	Обнаружено	Вредоносное ПО
CyRadar	Обнаружено	Вредоносное ПО
ESET	Обнаружено	Фишинг
Forcepoint ThreatSeeker	Обнаружено	Вредоносное ПО
Fortinet	Обнаружено	Фишинг
G-Data	Обнаружено	Фишинг
Kaspersky	Обнаружено	Фишинг
Lionic	Обнаружено	Фишинг
Seclookup	Обнаружено	Вредоносное ПО
SOCRadar	Обнаружено	Вредоносное ПО
Sophos	Обнаружено	Фишинг
Trustwave	Обнаружено	Подозрительный сайт
VIPRE	Обнаружено	Вредоносное ПО
Webroot	Обнаружено	Вредоносное ПО

Рис. 8 – Расширенный отчёт проверки через VirusTotal

Если при анализе не было обнаружено угроз, то приложение так же показывает произведённый анализ по всем критериям и выводит подробный результат, пример обработки безопасной ссылки представлен ниже (рис. 9)

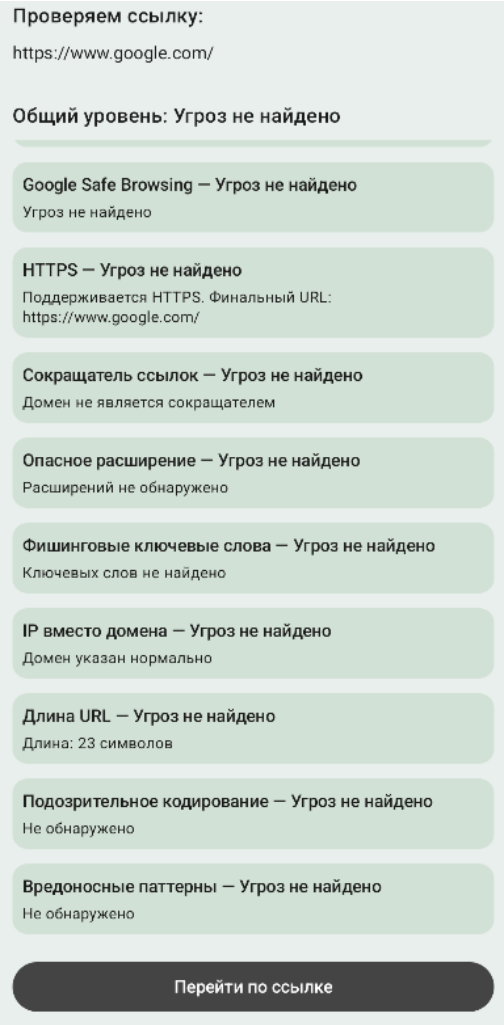


Рис. 9 – Результат анализа безопасной ссылки

В перспективе предусмотрено расширение функциональности за счёт добавления WHOIS/RDAP-модуля и поддержки облачной аналитики для коллективного обучения эвристик. Дальнейшее развитие может включать использование моделей машинного обучения для классификации URL на основе признаков, извлечённых из корпуса данных.

Вывод. Можно сделать вывод, что разработанное мобильное приложение для анализа QR-кодов соответствует современным требованиям к обеспечению информационной безопасности и демонстрирует устойчивость к основным видам фишинговых угроз. Реализованный механизм защиты успешно справляется с обнаружением и блокировкой попыток перехода по вредоносным ссылкам, закодированным в QR-кодах, а также минимизирует риск утечки пользовательских данных.

Существует ряд рекомендаций, которым пользователю следует придерживаться, чтобы снизить вероятность успешной фишинговой атаки:

- Внимательно проверять источник размещения QR-кода — не сканировать коды с неизвестных или подозрительных носителей (листовок, писем, рекламных баннеров).
- Использовать программные средства, выполняющие предварительный анализ ссылок перед открытием в браузере.
- Избегать ввода конфиденциальных данных (логинов, паролей, платёжных реквизитов) на страницах, открытых через случайные QR-коды.
- Обновлять операционную систему и приложения, чтобы минимизировать уязвимости, которые могут быть использованы злоумышленниками.
- При работе в корпоративной среде использовать централизованные инструменты безопасности, включая системы контроля сетевого трафика и фильтрации контента.

В результате исследования все поставленные цели были достигнуты: проведён анализ кибератак типа квишинг, рассмотрены существующие методы защиты и выявлены их ограничения, спроектировано и реализовано мобильное приложение для проверки безопасности QR-ссылок, а также выполнено тестирование, подтвердившее правильность работы разработанного механизма.

Обзор аналогичных программных решений и сравнение их с разработанной программной реализацией.

Kaspersky QR Scanner. Платформа: Android, iOS. Функциональность: выполняет базовую проверку URL, отображает ссылку до перехода. Преимущества: надёжный бренд, простота использования. Недостатки: платное приложение, не анализирует редиректы, не использует внешние базы данных или API.

Trend Micro QR Scanner. Платформа: Android. Функциональность: проверка ссылок с использованием облачной инфраструктуры, анализ редиректов, предупреждение о вредоносном содержимом. Преимущества:

высокая скорость работы, активная поддержка, актуальные базы угроз. Недостатки: доступен только для Android, отсутствует расширенная диагностика URL.

Norton Snap QR Code Reader. Платформа: ранее — iOS, приложение снято с поддержки в 2021 году. Функциональность: выполнял проверку ссылок по базе Symantec, предоставлял предупреждения о подозрительных ресурсах. Преимущества: глубокая проверка ссылок (на момент поддержки). Недостатки: неактуальность, отсутствие поддержки, несовместимость с новыми ОС.

SecScanQR (open-source). Платформа: Android. Функциональность: позволяет подключать сторонние API, в том числе VirusTotal и другие, поддерживает базовую фильтрацию подозрительных ссылок. Преимущества: гибкость настройки, открытый исходный код. Недостатки: нестабильная работа, необходимость ручной конфигурации, ограниченный пользовательский интерфейс.

ScanURL (веб-сервис). Платформа: веб-интерфейс. Функциональность: принимает на вход URL, проводит анализ на фишинг, зловредный код, репутацию домена (через базы Google, PhishTank и McAfee). Преимущества: доступность, простота использования. Недостатки: не является мобильным приложением, не предназначен для автоматизированной обработки QR.

Наименование решения	**Декодирование содержимого**	**Проверка URL**	**Анализ редиректов**	**Использование API**	**Акту баз**
Kaspersky QR Scanner	+	+	–	–	Сред
Trend Micro QR Scanner	+	+	+	–	Высо

Norton Snap QR Reader	+	+	+	–	Низк
SecScanQR (open-source)	+	Частично	Частично	+	Зави
ScanURL (веб-сервис)	–	+	Частично	+	Высо

Таблица 11 — Сравнительная характеристика существующих решений для проверки безопасности QR-кодов

Разработанное программное средство имеет ряд преимуществ над существующими аналогами благодаря комплексному подходу к анализу QR-ссылок, который сочетает локальные проверки безопасности, анализ цепочек переадресаций и обращение к внешним сервисам репутации доменов (таким как Google Safe Browsing и VirusTotal). В отличие от большинства доступных решений, приложение проводит глубокий многоуровневый анализ: оценивает структуру URL, проверяет наличие подозрительных параметров, а также валидность SSL-сертификата целевого ресурса. Благодаря асинхронной обработке запросов обеспечивается высокая скорость анализа и стабильность работы даже при нестабильном интернет-соединении.

Библиографический список:

1. [Электронный ресурс] Anti-Phishing Working Group. Phishing Activity Trends Report Q1 2025. — APWG, 2025. — URL: https://apwg.org/reports (дата обращения: 10.10.2025).

2. [Электронный ресурс]. Kaspersky. Kaspersky Security Bulletin 2025. — Kaspersky Lab, 2025. — URL: https://securelist.com/reports (дата обращения: 10.10.2025).
3. [Электронный ресурс] APWG. Phishing Activity Trends Report (PDF). — APWG, 2025. — URL: https://docs.apwg.org/reports/apwg_trends_report_q1_2025.pdf (дата обращения: 10.10.2025).
4. [Электронный ресурс]. Barracuda Networks. Threat Spotlight: Evolving QR Code Phishing Attacks. — Barracuda Blog, 2024. — URL: https://blog.barracuda.com/2024/10/22/threat-spotlight-evolving-qr-codes-phishing-attacks (дата обращения: 10.10.2025).
5. [Электронный ресурс]. Google. Google Safe Browsing API v4 Documentation. — Google Developers, 2025. — URL: https://developers.google.com/safe-browsing/v4 (дата обращения: 10.10.2025).
6. ACM Digital Library. Phishing Detection Research Papers. — DOI: 10.1145/3549015.3554172, ACM Press, 2022.
7. [Электронный ресурс]. arXiv. Deep Learning Models for Phishing URL Detection. — arXiv preprint arXiv:2505.0345, 2025.
8. [Электронный ресурс]. VirusTotal. API Overview. — VirusTotal Documentation, Google Cloud, 2025. — URL: https://docs.virustotal.com/docs/api-overview (дата обращения: 10.10.2025).