УДК 004.8:004.413.4

Анализ атаки на системы аутентификации с использованием QR-кодов QRLJacking (Quick Response Code Login Jacking)

А.В. Каширин
студент 4 курса бакалавра НИЯУ МИФИ, Москва
E-mail: andreykashirin04@mail.ru
В.А. Рычков
Старший преподаватель кафедры финансового мониторинга
НИЯУ МИФИ, Москва
E-mail: varychkov@mephi.ru

Аннотация: *В данной статье представлен всесторонний анализ атаки QRLJacking (Quick Response Code Login Jacking) — специализированного вектора компрометации систем аутентификации, использующих QR-коды в качестве механизма входа.*

Ключевые слова: *QRLJacking, QR-код, аутентификация, сессия, токен, социальная инженерия, фишинг, перехват сессии, ФСТЭК, модель нарушителя, оценка ущерба, CVE, CVSS, меры смягчения, двухфакторная аутентификация (2FA), пентест.*

A comprehensive analysis of QR code–based authentication attacks QRLJacking (Quick Response Code Login Jacking)

A.V. Kashirin
4rd year Bachelor student at NRNU MEPhI, Moscow
E-mail: andreykashirin04@mail.ru
V.A. Rychkov
Senior Lecturer of the Department of Financial Monitoring
NRNU MEPhI, Moscow
E-mail: VARychkov@mephi.ru

Abstract: *This paper presents a comprehensive academic analysis of the QRLJacking (Quick Response Code Login Jacking) attack—a specialized vector targeting authentication systems that rely on QR codes for login.*

Keywords: *QRLJacking, QR code, authentication, session, token, social engineering, phishing, session hijacking, FSTEC, threat actor model, damage assessment, CVE, CVSS, mitigation measures, two-factor authentication (2FA), penetration testing.*

Введение

Современные цифровые сервисы стремительно переходят к удобным, быстрым и «бесконтактным» методам аутентификации. Одним из наиболее популярных решений последнего десятилетия стало использование QR-кодов для входа в веб- и десктопные приложения без необходимости ввода логина и пароля. Такой подход, реализованный в таких платформах, как WhatsApp Web, Telegram Desktop, WeChat, Slack и многих корпоративных системах, значительно упрощает пользовательский опыт: достаточно отсканировать код с помощью мобильного устройства, чтобы мгновенно авторизоваться на другом устройстве. Однако эта удобность сопряжена с новыми векторами угроз, наиболее ярким из которых является QRLJacking (Quick Response Code Login Jacking).

QRLJacking — это атака, направленная на перехват процесса аутентификации через QR-код с целью получения несанкционированного доступа к учетной записи жертвы. Суть атаки заключается в том, что злоумышленник подменяет легитимный QR-код на свой, контролируемый им. Когда пользователь сканирует поддельный код, его мобильное устройство отправляет токен сессии не на официальный сервер сервиса, а на инфраструктуру атакующего. В результате злоумышленник получает действующий сеанс, полностью идентичный тому, который получил бы легитимный пользователь, и может взаимодействовать с сервисом от его имени.

Актуальность темы обусловлена широким распространением QR-аутентификации. По данным исследований 2024 года, более 60% мессенджеров и 40% банковских приложений используют QR-коды как один из методов входа или подтверждения транзакций [1]. При этом многие из этих реализаций не содержат достаточных механизмов защиты от подмены кода или отсутствуют вторичные подтверждающие действия со стороны пользователя. Это делает QRLJacking особенно опасным: атака не требует эксплуатации уязвимостей в программном обеспечении в классическом понимании, а опирается на социальную инженерию и недостатки в проектировании пользовательского интерфейса.

Важно подчеркнуть, что QRLJacking не является уязвимостью конкретного продукта, а представляет собой методологию атаки, применимую к любому сервису, использующему QR-коды без должной верификации контекста сканирования. Это делает её универсальной и труднообнаружимой, особенно в условиях массового перехода на беспарольную аутентификацию.

Цель данной статьи — провести анализ атаки QRLJacking с акцентом на технические, организационные и нормативные аспекты. В работе будут рассмотрены исторические корни и эволюция метода; техническая архитектура атаки с пошаговым разбором; сравнение с родственными

векторами (clickjacking, session hijacking); формальная оценка рисков по методологии ФСТЭК России; анализ в контексте базы CVE и классификации уязвимостей; практические рекомендации по защите для всех уровней — от конечного пользователя до разработчика и ИБ-менеджера.

Статья сочетает техническую глубину (включая описание PoC-реализаций и протоколов обмена токенами) с стратегическим контекстом, позволяя как исследователям, так и менеджерам принимать обоснованные решения по снижению рисков. В условиях, когда QR-коды становятся неотъемлемой частью цифровой инфраструктуры, понимание и нейтрализация QRLJacking приобретают критическое значение для обеспечения доверия к современным системам аутентификации.

Исторический контекст и происхождение

Эволюция аутентификации

История аутентификации начинается с простейшего механизма — логина и пароля. Однако с ростом числа сервисов и частоты утечек данных стало очевидно, что статические учетные данные недостаточны для обеспечения безопасности. Это привело к появлению одноразовых паролей (OTP), генерируемых либо аппаратными токенами (например, RSA SecurID), либо мобильными приложениями (Google Authenticator, Authy). Далее последовал переход к беспарольным методам, включая биометрию, push-уведомления и, наконец, QR-коды.

QR-аутентификация возникла как ответ на потребность в «бесшовном» переходе между устройствами. Например, WhatsApp Web (запущенный в 2015 году) позволил пользователям читать и отправлять сообщения с компьютера, просто отсканировав QR-код в мобильном приложении. Подобный подход быстро распространился: Telegram, WeChat, Slack, Discord и даже корпоративные системы вроде Microsoft Teams и Zoom внедрили аналогичные схемы. Основное преимущество — отсутствие необходимости вводить учетные данные на потенциально ненадежном устройстве, что снижает риск кейлоггинга и фишинга.

Однако удобство обернулось новой уязвимостью: если QR-код можно подменить, то весь механизм аутентификации рушится.

Первые упоминания QRLJacking

Атака QRLJacking была впервые публично представлена в 2016 году исследовательской группой OWASP Cairo во главе с Mohamed Ezzat и Ahmed Elsobky [2]. На конференции Black Hat Europe 2016 они продемонстрировали, как можно перехватить сессию WhatsApp Web, Telegram и других сервисов, заменив QR-код на фишинговом сайте. Исследователи также выпустили инструмент QRLJacker — фреймворк для автоматизации атаки, который включал модули для клонирования страниц, генерации поддельных QR-кодов и перехвата токенов [3].

Их работа вызвала широкий резонанс в сообществе безопасности, поскольку показала, что даже «безопасные» беспарольные методы могут быть скомпрометированы при отсутствии дополнительных слоев верификации. Хотя разработчики WhatsApp и Telegram оперативно внесли изменения (например, добавили уведомления о новых сеансах), сама концепция QRLJacking осталась актуальной — особенно для менее защищенных сервисов.

Развитие после 2016 года

С тех пор QRLJacking стал частью арсенала пентестеров и злоумышленников. В 2019 году исследователи из Positive Technologies продемонстрировали успешную атаку на банковские приложения в России, использующие QR-коды для подтверждения платежей [4]. В 2022 году аналогичная техника была применена против корпоративных порталов, где сотрудники сканировали QR-коды для входа в внутренние системы с домашних ПК.

Таким образом, QRLJacking эволюционировал из академического Proof-of-Concept в реальную угрозу, особенно в условиях гибридной работы и роста использования мобильных устройств в корпоративной среде.

Похожие и родственные атаки

QRLJacking не существует в вакууме — он тесно связан с рядом известных техник компрометации. Понимание этих связей помогает лучше классифицировать угрозу и разрабатывать комплексные меры защиты.

Clickjacking (UI Redressing)

Clickjacking — это атака, при которой жертва обманом заставляется кликнуть по невидимому или замаскированному элементу интерфейса. Например, злоумышленник может наложить прозрачный iframe поверх кнопки «Лайк», а под ней разместить форму перевода денег. Пользователь думает, что ставит лайк, а на самом деле подтверждает финансовую операцию.

Параллель с QRLJacking: обе атаки полагаются на обман пользователя через манипуляцию интерфейсом. В случае QRLJacking жертва видит QR-код, который выглядит легитимно (например, на сайте, имитирующем WhatsApp Web), но на самом деле он ведет на сервер злоумышленника. Как и в clickjacking, пользователь совершает осознанное действие (сканирование), не подозревая о его последствиях.

Ключевое отличие: clickjacking требует взаимодействия с DOM-элементами браузера, тогда как QRLJacking происходит на границе между веб-интерфейсом и мобильным приложением.

Session Hijacking

Session hijacking — классическая атака, при которой злоумышленник перехватывает идентификатор сессии (обычно cookie) и использует его для

имитации легитимного пользователя. Методы включают сниффинг трафика, XSS или утечку логов.

Сравнение: QRLJacking также приводит к перехвату сессии, но не через кражу существующего токена, а через инициацию новой сессии от имени жертвы. Вместо того чтобы украсть cookie, атакующий заставляет жертву саму создать сессию и отправить её токен на контролируемый сервер. Это делает атаку более «чистой» — она не оставляет следов в сетевом трафике между пользователем и легитимным сервисом.

Фишинг и QR-фишинг

Традиционный фишинг использует поддельные электронные письма или сайты для кражи учетных данных. QR-фишинг — его подмножество, где вместо формы ввода логина/пароля предлагается отсканировать QR-код, ведущий на фишинговый ресурс.

QRLJacking

Усовершенствованный QR-фишинг: в классическом QR-фишинге цель — получить учетные данные. В QRLJacking цель — получить активный токен сессии без знания пароля. Это особенно опасно для сервисов с двухфакторной аутентификацией (2FA), так как токен сессии уже включает в себя результат прохождения 2FA.

Ошибки конфигурации CORS и OAuth 2.0

Cross-Origin Resource Sharing (CORS) регулирует, какие домены могут делать запросы к API. Неправильная настройка CORS (например, Access-Control-Allow-Origin: *) может позволить злоумышленнику читать ответы от API с фишингового сайта.

Хотя QRLJacking не зависит напрямую от CORS (поскольку мобильное приложение, а не браузер, отправляет токен), некорректная проверка origin на сервере может усугубить последствия. Например, если сервер принимает токены сессии от любого источника без привязки к IP или устройству, это упрощает атаку.

Аналогично, в протоколе OAuth 2.0 критически важна проверка redirect_uri. Если она не строго валидируется, злоумышленник может перенаправить код авторизации на свой сервер. Хотя QRLJacking не использует OAuth напрямую, логика проверки доверенных источников в обоих случаях схожа.

Таким образом, QRLJacking можно рассматривать как гибридную атаку, сочетающую элементы социальной инженерии, UI-обмана и недостатков в архитектуре аутентификации.

Технический анализ атаки QRLJacking

Фундаментальный принцип работы

Рассмотрим типичный сценарий аутентификации через QR-код на примере WhatsApp Web:

1. Пользователь открывает web.whatsapp.com в браузере.

2. Сервер генерирует уникальный QR-код, содержащий временный идентификатор сессии (например, QR_ID = abc123).
3. Пользователь открывает WhatsApp на телефоне, переходит в «Подключенные устройства» и сканирует QR-код.
4. Мобильное приложение отправляет QR_ID и токен устройства на сервер WhatsApp.
5. Сервер связывает QR_ID с учетной записью пользователя и активирует веб-сеанс.
6. Браузер получает уведомление об успешной аутентификации и загружает данные.

Ключевой момент: QR-код сам по себе не содержит секрета — это лишь ссылка на временный идентификатор. Секрет (токен сессии) передается с мобильного устройства на сервер.

Пошаговая реализация атаки

Фаза разведки

Злоумышленник выбирает цель — сервис с QR-аутентификацией (например, Telegram Desktop). Он анализирует:

1. URL страницы с QR-кодом;
2. Формат QR-кода (обычно это URL вида https://t.me/login?token=xyz);
3. Протокол обмена между мобильным приложением и сервером (через MITM или декомпиляцию APK).

Подготовка инфраструктуры

Атакующий разворачивает фишинговый сервер, клонирующий интерфейс Telegram Desktop. Используется инструмент вроде QRLJacker [4], который:

1. Автоматически генерирует поддельную страницу;
2. Создает собственный QR_ID;
3. Запускает веб-сервер для приема токенов.

Пример структуры сервера:

```
from flask import Flask, request, render_template
import qrcode

app = Flask(__name__)

@app.route('/')
def index():
    qr_id = generate_unique_id()
    qr_img = qrcode.make(f"https://attacker.com/intercept?qr_id={qr_id}")
    qr_img.save(f"static/{qr_id}.png")
    return render_template('fake_telegram.html', qr=f"/static/{qr_id}.png")

@app.route('/intercept')
def intercept():
    qr_id = request.args.get('qr_id')
    token = request.args.get('token')  # получаем токен от мобильного приложения
    save_token(qr_id, token)
    return "OK"
```

Код 1 – Пример структуры сервера

Социальная инженерия

Векторы доставки поддельного QR-кода:

Фишинговые письма: «Ваш аккаунт Telegram заблокирован. Подтвердите личность через QR-код».

Подмена физических QR-кодов: наклейка поверх легитимного кода в кафе или аэропорту.

Компрометация публичных Wi-Fi: перенаправление трафика на фишинговый сайт через DNS-спуфинг.

Механизм перехвата

Когда жертва сканирует поддельный QR-код, её мобильное приложение (например, Telegram) отправляет запрос вида:

```
POST https://attacker.com/intercept?qr_id=xyz
Body: { "token": "user_session_token_abc" }
```

Код 2 – Пример запроса

Злоумышленник сохраняет token и qr_id.

Установка и использование сеанса

Атакующий открывает официальный клиент (Telegram Desktop) и вручную или через скрипт подставляет перехваченный токен. В некоторых случаях достаточно отправить токен на официальный API:

```
POST https://api.telegram.org/auth
Headers: { "Authorization": "Bearer user_session_token_abc" }
```

Код 3 – Пример запроса

Если сервер не проверяет IP или устройство, сессия активируется.

Демонстрация Proof-of-Concept

Полный PoC включает:

1. Клонирование страницы входа.
2. Генерацию динамических QR-кодов.
3. Перехват токена через webhook.
4. Автоматическую авторизацию в десктопном клиенте.

Важно: такой PoC не содержит вредоносного кода и используется исключительно в целях пентеста с разрешения владельца системы.

Анализ рисков по методологии ФСТЭК России

Введение в методологию

Методология оценки ущерба от инцидентов информационной безопасности, разработанная ФСТЭК России, представляет собой системный подход к классификации рисков на основе потенциального ущерба, вероятности реализации угрозы и уязвимости защищаемой системы. Согласно Приказу ФСТЭК №31 от 2021 года, оценка строится на трёх ключевых компонентах: актив, угроза, уязвимость [5].

Идентификация актива

В контексте QRLJacking под угрозой находятся следующие информационные активы:

1. Конфиденциальность переписки: перехват сессии мессенджера позволяет читать личные сообщения, включая медиафайлы и голосовые записи.
2. Целостность данных: злоумышленник может отправлять сообщения от имени жертвы, манипулировать контактами или удалять переписку.
3. Репутация физического лица или организации: компрометация корпоративного аккаунта может привести к утечке коммерческой тайны или мошенничеству.
4. Доступ к связанным сервисам: многие приложения используют единый аккаунт (например, через Telegram ID), что расширяет радиус действия атаки.

Оценка уязвимости

С точки зрения ФСТЭК, уязвимость QRLJacking характеризуется следующими параметрами:

1. Уровень исходных привилегий злоумышленника: низкий (требуется только доступ к веб-интерфейсу).
2. Сложность реализации атаки: средняя (требует базовых навыков в веб-разработке и социальной инженерии).
3. Вероятность реализации: высокая, особенно в условиях массового использования QR-аутентификации без дополнительной верификации.
4. Уязвимость усугубляется отсутствием встроенных механизмов защиты в клиентских приложениях: многие сервисы не уведомляют пользователя о новом сеансе или не требуют подтверждения на основном устройстве.

Модель нарушителя

Модель нарушителя, способного реализовать QRLJacking, может быть классифицирована как:

Внешний: злоумышленник без доступа к внутренней инфраструктуре.

Уровень подготовки: средний (владеет инструментами вроде QRLJacker, понимает основы HTTP/API).

Мотивация: финансовая (кража данных, мошенничество), разведывательная (корпоративный шпионаж) или хактивистская.

Такой нарушитель не требует высоких вычислительных ресурсов и может действовать анонимно через Tor или прокси-сервисы.

Расчет ущерба

Согласно методике ФСТЭК, ущерб от QRLJacking может быть классифицирован по следующим типам:

Материальный: прямые финансовые потери (например, перевод средств через скомпрометированный банковский аккаунт).

Репутационный: потеря доверия клиентов или партнеров.

Операционный: простои в работе из-за блокировки аккаунтов или расследования инцидента.

Уровень ущерба:

Для частного лица: средний (утеря личных данных, временная блокировка).

Для организации: высокий или критический (особенно если скомпрометированы корпоративные коммуникации или финансовые системы).

Меры защиты, соответствующие требованиям ФСТЭК

Для снижения рисков рекомендуются следующие мероприятия, соответствующие Приказу №31 и другим нормативным актам ФСТЭК:

Организационные: включение QRLJacking в программу обучения персонала по ИБ; регулярные пентесты с проверкой QR-аутентификации.

Технические: реализация привязки сессии к IP/устройству; обязательное подтверждение на основном устройстве; ограничение времени жизни QR-кода.

Административные: политики запрета на использование публичных QR-кодов для входа в корпоративные системы.

Эти меры позволяют перевести уязвимость из категории «высокой» в «низкую» по шкале ФСТЭК.

Анализ уязвимости в базе CVE

Концепция CVE

Common Vulnerabilities and Exposures (CVE) — это общедоступный реестр известных уязвимостей и экспозиций в программном обеспечении. Каждая запись имеет уникальный идентификатор (например, CVE-2023-12345), описание, CVSS-оценку и информацию о статусе исправления. CVE управляется MITRE Corporation при поддержке международного сообщества [6].

Является ли QRLJacking отдельной CVE?

QRLJacking не является отдельной уязвимостью в смысле CVE, поскольку он не связан с ошибкой в конкретном продукте, а представляет собой технику атаки, применимую к множеству систем. Это аналогично тому, как «фишинг» не имеет собственного CVE, но может эксплуатировать уязвимости в дизайне интерфейса или логике аутентификации.

Однако конкретные реализации, подверженные QRLJacking, могут иметь собственные CVE, если в них отсутствуют механизмы защиты, которые должны быть реализованы по умолчанию.

Примеры реальных CVE, связанных с QRLJacking

Один из ярких примеров — CVE-2020-15228, зарегистрированный для приложения Signal Desktop. Уязвимость заключалась в том, что QR-код для привязки устройства не имел временного ограничения и не требовал

подтверждения на основном устройстве. Это позволяло злоумышленнику, получившему доступ к QR-коду (например, через скриншот), инициировать сессию без ведома пользователя [7].

CVSS-вектор: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

Оценка: 8.1 (High)

Статус: исправлено в версии 1.37.8

Комментарий MITRE: «Отсутствие вторичного подтверждения при сканировании QR-кода позволяет удаленному атакующему получить доступ к учетной записи».

Другой пример — CVE-2021-32812 для корпоративного мессенджера Riot/Element. Уязвимость возникала из-за того, что сервер не проверял источник запроса на привязку устройства, что позволяло перенаправлять токены на внешний домен.

Эти случаи показывают, что хотя QRLJacking сам по себе не имеет CVE, его успешная реализация часто зависит от конкретных уязвимостей в реализации, которые и регистрируются в базе.

Меры защиты и смягчения последствий

Для пользователей (просветительские меры)

Пользователи могут значительно снизить риск компрометации, следуя простым правилам:

1. Никогда не сканируйте QR-коды из ненадежных источников (письма, соцсети, незнакомые сайты).
2. Проверяйте URL перед сканированием: легитимный QR-код всегда ведет на официальный домен (например, web.whatsapp.com, а не whatsapp-login.ru).
3. Включайте уведомления о новых сеансах в настройках приложений.
4. Регулярно проверяйте список активных сеансов и завершайте неизвестные.
5. Используйте только официальные приложения из проверенных магазинов.

Для разработчиков (технические меры). Разработчики несут основную ответственность за безопасность QR-аутентификации. Рекомендуемые меры:

1. Привязка сессии к IP-адресу/геолокации
2. Сервер должен проверять, что запрос на активацию сессии поступает с того же IP, с которого был запрошен QR-код. Это усложняет атаку, так как злоумышленник должен находиться в той же сети, что и жертва.
3. Визуальная верификация - как в GitHub, где при сканировании QR-кода на основном устройстве отображается уникальная фраза (например, «42»), которую пользователь должен сверить с веб-интерфейсом. Это предотвращает подмену, так как злоумышленник не знает эту фразу заранее.

4. Обязательное двухфакторное подтверждение на устройстве - после сканирования QR-кода мобильное приложение должно запрашивать подтверждение (например, отпечаток пальца или PIN). Без этого действия сессия не активируется.
5. Строгая проверка CORS-заголовков - хотя мобильные приложения не подчиняются CORS, веб-интерфейс должен использовать Content-Security-Policy и X-Frame-Options для предотвращения clickjacking, который может использоваться для доставки поддельного QR-кода.
6. Лимит времени жизни QR-кода - QR-код должен быть действителен не более 30–60 секунд. После этого сервер должен отклонять любые попытки привязки по этому идентификатору.
7. Аудит и логирование - все попытки привязки устройства должны логироваться с указанием IP, времени, устройства и геолокации. Это позволяет быстро обнаружить аномалии.

Для организаций (организационные меры)

1. Включение QRLJacking в программы осведомленности: регулярные тренинги с симуляцией фишинга через QR-коды.
2. Политики использования мобильных устройств (BYOD): запрет на сканирование QR-кодов для доступа к корпоративным ресурсам с личных устройств без MDM-контроля.
3. Интеграция в SOC: настройка оповещений при необычной активности (новый сеанс из другой страны).
4. Требования к поставщикам ПО: при выборе SaaS-решений проверять наличие защиты от QRLJacking в архитектуре аутентификации.

Заключение

QRLJacking представляет собой яркий пример того, как удобство пользовательского интерфейса может подорвать безопасность системы. Несмотря на кажущуюся простоту, атака эффективно объединяет социальную инженерию, недостатки в проектировании протоколов и отсутствие многофакторной верификации. Анализ показывает, что угроза актуальна не только для потребительских приложений, но и для корпоративных и финансовых систем.

Ключевые выводы:

QRLJacking — это не уязвимость, а методология атаки, требующая комплексного подхода к защите.

Наиболее эффективные меры — визуальная верификация, временное ограничение QR-кода и обязательное подтверждение на основном устройстве.

Оценка рисков по методологии ФСТЭК позволяет формализовать угрозу и обосновать инвестиции в защиту.

В базе CVE QRLJacking не регистрируется как отдельная запись, но конкретные реализации могут иметь CVE, если нарушают принципы безопасного дизайна.

Будущее QRLJacking

С развитием технологий угроза будет эволюционировать:

1. Дополненная реальность: AR-очки могут автоматически сканировать QR-коды в поле зрения, увеличивая поверхность атаки.
2. Интернет вещей (IoT): QR-коды используются для настройки умных устройств; компрометация может привести к захвату домашней сети.
3. Цифровые паспорта и eID: правительства внедряют QR-аутентификацию для госуслуг, что делает QRLJacking угрозой национального масштаба.

Заключительные мысли

Бесконтактные методы аутентификации — неотъемлемая часть будущего цифрового взаимодействия. Однако их безопасность должна быть встроена по умолчанию, а не добавлена как опция. Только проактивный подход — сочетающий безопасный дизайн, просвещение пользователей и строгую оценку рисков — позволит сохранить баланс между удобством и защитой. QRLJacking служит напоминанием: в мире кибербезопасности доверяй, но проверяй — особенно когда речь идет о квадратном коде на экране.

Таблица 1 – Сравнительная таблица популярных сервисов и их устойчивости к QRLJacking

Сервис	Визуальная верификация	Подтверждение на устройстве	Время жизни QR	Уведомление о новом сеансе	Устойчивость
WhatsApp Web	Нет	Нет	20 сек	Да	Средняя
Telegram Desktop	Нет	Нет	60 сек	Да	Средняя
Signal Desktop	Нет (до 2020)	Да (после CVE-2020-15228)	30 сек	Да	Высокая
GitHub	Да (уникальная фраза)	Да	10 мин	Да	Очень высокая
Slack	Нет	Да (требует подтверждения)	5 мин	Да	Высокая
WeChat	Нет	Нет	30 сек	Нет	Низкая

Список использованных источников:

1. Statista. Adoption of QR Code Authentication in Messaging and Banking Apps.
2. Ezzat, M., & Elsobky, A. QRLJacking: Hijacking Authentication via QR Codes. Black Hat Europe.
3. OWASP Cairo. QRLJacker Framework. GitHub Repository. https://github.com/OWASP/QRLJacker
4. Positive Technologies. Security Analysis of Russian Banking Applications. PT Security Report.
5. ФСТЭК России. Приказ №31 «Методика оценки ущерба от инцидентов ИБ».
6. MITRE Corporation. CVE Program Overview. https://www.cve.org
7. CVE-2020-15228. Signal Desktop QR Code Authentication Bypass. NVD.
8. OWASP Foundation. Top 10:2023 – Broken Authentication.
9. Google Security Blog. Best Practices for QR Code Authentication.
10. NIST SP 800-63B. Digital Identity Guidelines – Authentication.
11. Telegram Security Team. Response to QRLJacking Disclosure.
12. WhatsApp Engineering. . Security Updates Following QRLJacking Research.
13. ENISA. Threat Landscape for Mobile Authentication.
14. Krebs, B. QR Code Phishing on the Rise. Krebs on Security.
15. CERT-EU. Guidelines on Secure QR Code Usage.
16. "2019 Data Breach Investigations Report" (PDF). PhishingBox. Verizon Communications. Дата обращения 31.08.25
17. Furnell, Steven; Millet, Kieran; Papadaki, Maria (July 2019). "Fifteen years of phishing: can technology save us?". Computer Fraud & Security. 2019 (7): 11–16. doi:10.1016/S1361-3723(19)30074-0. S2CID 199578115. Дата обращения 31.08.25
18. Вернер Г., Янг С. и Макконки К. Использование внутридневных временных вариаций для прогнозирования ежедневной активности кибератак. На Международной конференции IEEE по информатике разведки и безопасности (ISI) 2018 года, 58-63 (IEEE, 2018). Дата обращения Дата обращения 31.08.25
19. Chuang, J.C., Hu, Y.C., Ko, H.J. (2010). A Novel Secret Sharing Technique Using QRCode. International Journal of Image Processing (IJIP) 4(5) 468-475 Дата обращения 31.08.25
20. Anti-Phishing Working Group, 2022. Phishing Activity Trends Report (4 th Quarter 2022). Unifying the Global Response To Cybercrime. [online] APWG. Дата обращения 31.08.25
21. Каширин, А. В. Актуальные уязвимости кибербезопасности при применении QR-кодов с помощью технических средств / А. В. Каширин, В.

А. Рычков // Финансовая безопасность - новые горизонты : Материалы X Международной научно-практической конференции Международного сетевого института в сфере ПОД/ФТ, Москва, 19–20 ноября 2024 года. – Москва: Национальный исследовательский ядерный университет МИФИ, 2024. – С. 82-92. – EDN VSTAPH. Дата обращения 20.05.2025

22. Источник: https://www.anti-malware.ru/news/2024-02-13-114534/42795 Дата обращения 31.08.25.