УДК 004.054

Королев Юрий Алексеевич

магистрант, Университет ИТМО, город Санкт-Петербург

Любичев Ефим Денисович

магистрант, Университет ИТМО, город Санкт-Петербург

Государев Илья Борисович

доцент, кандидат педагогических наук, Университет ИТМО,

город Санкт-Петербург

АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ МЕХАНИЗМОВ АУТЕНТИФИКАЦИИ ЧЕРЕЗ ЕСИА В ВЕБ-ПЛАТФОРМЕ АРЕНДЫ НЕДВИЖИМОСТИ

Аннотация

В статье рассматривается проблема автоматизированного тестирования механизмов аутентификации через Единую систему идентификации и аутентификации (ЕСИА) в веб-платформе аренды недвижимости. Проанализированы архитектурные особенности интеграции на основе протоколов OAuth 2.0 и OpenID Connect, раскрыты вопросы криптографической валидации токенов и управления состояниями в распределенной среде. Предложен воспроизводимый подход к построению изолированного тестового контура, позволяющий выявлять дефекты обработки JWT, параметров state и nonce, а также

сценарии отказа или деградации внешнего провайдера. Представлены результаты апробации методики и обоснована ее практическая применимость.

Ключевые слова:

ЕСИА, OpenID Connect, OAuth 2.0, JWT, автоматизированное тестирование, информационная безопасность, аренда недвижимости.

Платформы аренды недвижимости эволюционируют от витрин объявлений к средам, в которых пользователи совершают значимую последовательность действий: подтверждают личность, связывают контактные данные, резервируют объект, подписывают документы, оплачивают комиссии. В таких системах аутентификация перестает быть вспомогательной функцией и становится частью доверенной цепочки принятия решений. Использование ЕСИА в качестве поставщика идентификации снижает барьер входа для пользователей, позволяет использовать подтвержденные атрибуты личности и упрощает выполнение комплаенс-процедур. Вместе с тем интеграция с государственным провайдером предъявляет к реализации повышенные требования: ошибки в обработке перенаправлений, токенов и параметров защиты приводят не только к отказам входа, но и к более серьезным последствиям – подмене аккаунтов, некорректной привязке персональных данных, обходу ограничений доступа к функциональности. В отличие от локальной схемы аутентификации, где разработчик контролирует весь жизненный цикл сессии, при входе через ЕСИА значительная часть логики выполняется вне домена платформы. Это меняет природу тестирования. Ручная проверка «входа в браузере» фиксирует лишь функциональный факт успешной авторизации, но почти не затрагивает корректность валидации токенов, поведение при деградации внешнего контура, устойчивость к повторным запросам и безопасность контекста выполнения. Следовательно, необходима методика автоматизированного тестирования, которая одновременно проверяет протокол,

криптографию и управление состояниями, при этом оставаясь воспроизводимой в изолированной среде и достаточно близкой к реальному взаимодействию.

Целью работы является разработка методики автоматизированного тестирования механизмов аутентификации через ЕСИА в веб-платформе аренды недвижимости и показать, какие классы дефектов выявляются только при систематическом моделировании внешнего провайдера и негативных сценариев. Задачи исследования включают выделение критериев корректности реализации OpenID Connect-входа, разработку тестового контура имитации ЕСИА, описание набора проверок на уровне протокольного обмена и представление результатов апробации.

Для формализации требований к корректности реализации протокола OpenID Connect в тестовой методике введена система критериев проверки (таблица 1). Каждый критерий отражает отдельное свойство безопасности или корректности протокольного взаимодействия и связывается с конкретным тестовым сценарием и ожидаемым поведением системы. Это позволяет перевести требования спецификаций в набор воспроизводимых автоматизированных проверок.

Таблица 1 — Критерии корректности обработки OIDC-аутентификации

Критерий корректности	**Проверка**	**Ожидаемое поведение**	**Класс дефекта**
issuer (iss)	id_token подписан корректно, но содержит другой iss	токен отклоняется, сессия не создаётся	конфигурация доверенного провайдера
audience (aud)	токен содержит другой client_id	отказ аутентификации	нарушение привязки клиента

Критерий корректности	Проверка	Ожидаемое поведение	Класс дефекта
exp/iat	генерация токена с истекшим exp или будущим iat	отказ валидации	ошибка обработки времени
clock skew	токен валиден, но близок к границе времени	принимается только при допустимом CLOCK_SKEW	некорректная синхронизация времени
nonce mismatch	возвращён id_token с nonce другого запроса	отказ входа	подмена ответа авторизации
state mismatch	параметр state не совпадает с сохранённым	отказ входа	CSRF-уязвимость
authorization code reuse	повторный обмен одного code	token endpoint возвращает invalid_grant	отсутствие одноразовости
jwks rotation	подпись токена новым ключом	обновление JWKS и успешная валидация	некорректное кэширование ключей
missing JWKS key	ключ отсутствует в JWKS	отказ валидации	несогласованность ключей

Критерий корректности	Проверка	Ожидаемое поведение	Класс дефекта
timeout JWKS	искусственная задержка ответа JWKS	использование кэша или контролируемый отказ	устойчивость к деградации
timeout token endpoint	задержка token endpoint	корректная обработка таймаута и retry	сетевые сбои
invalid_grant	обмен code с нарушенным параметром	отказ входа	ошибка обработки протокола

Интеграцию с ЕСИА целесообразно рассматривать как распределенный процесс, в котором у платформы есть две разные роли. С одной стороны, она выступает клиентом авторизационного сервера (получает code и обменивает его на токены). С другой – она выступает издателем собственной сессии и точкой применения политики доступа, поскольку именно она решает, что означает успешный вход для конкретного домена: арендатора, арендодателя, агента или сотрудника поддержки. Классическая последовательность OpenID Connect состоит из перенаправления на сторону провайдера, получения authorization code, обмена кода на токены на серверной стороне и дальнейшей валидации id_token.

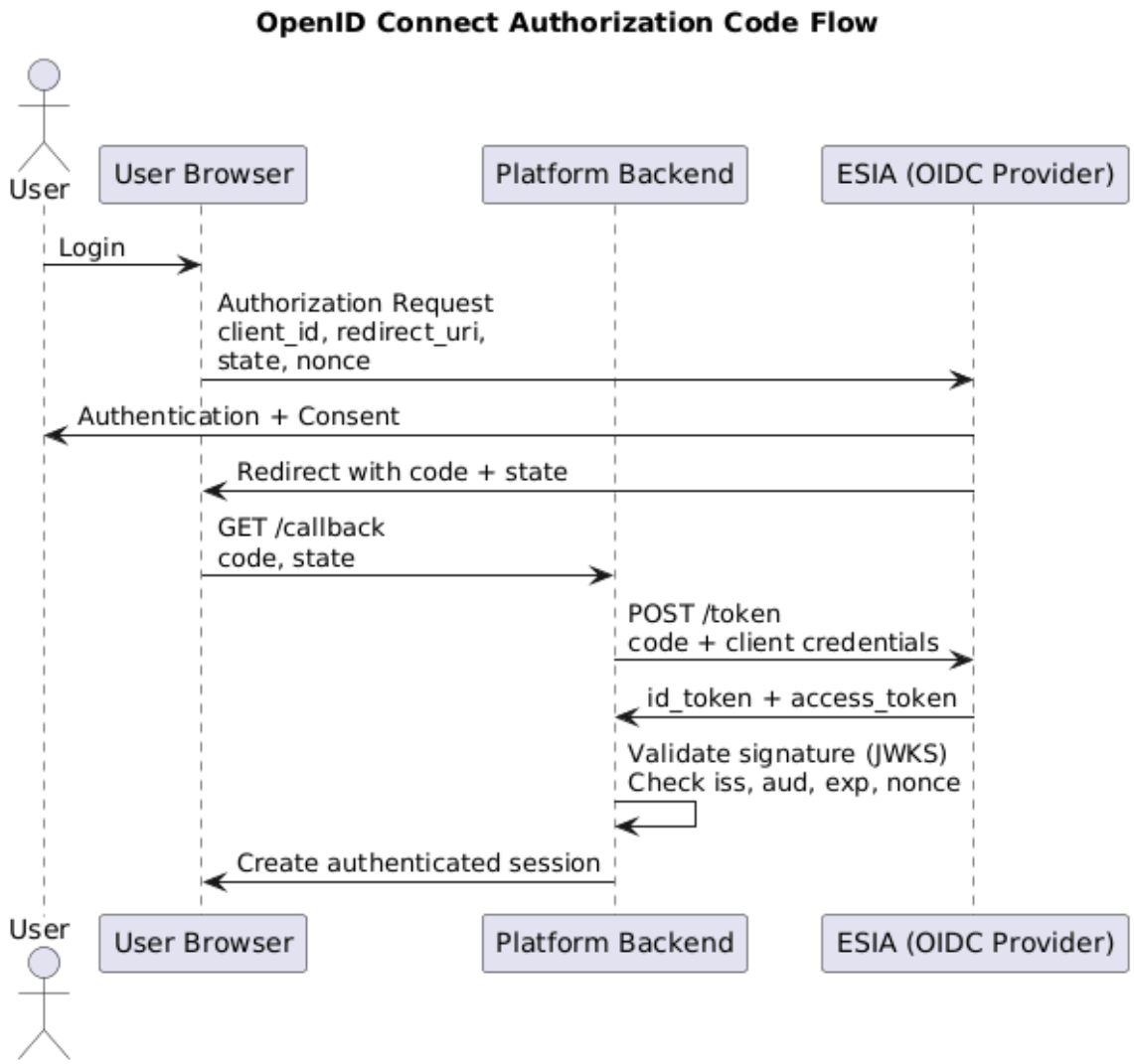

Рисунок 1 – Последовательность взаимодействия при аутентификации пользователя через OpenID Connect

На практике именно последняя стадия чаще всего реализуется поверхностно: разработчики ограничиваются проверкой подписи и наличия нескольких полей, тогда как корректность требует согласованности множества утверждений и параметров протокола. Валидация должна включать проверку издателя, получателя, срока действия, допустимого времени, одноразовости контекста и привязки авторизационного ответа к инициирующей сессии (nonce и state). Ошибка в любой из этих проверок может дать редкий, но опасный класс уязвимостей: от повторного использования токена до подмены результата входа в случае атаки с

межсайтовым запросом. Дополнительная сложность для предметной области аренды недвижимости связана с тем, что после первичного входа платформа часто связывает аккаунт с набором действий повышенной значимости: публикация объявления, просмотр данных другой стороны сделки, подписание договора, оформление платежа. Поэтому требования к тестированию аутентификации должны учитывать не только «вход выполнен», но и то, что после входа корректно сформированы роли, атрибуты пользователя и контекст безопасности. В частности, система должна корректно обрабатывать смену статуса пользователя, повторный вход с другим уровнем подтверждения, ситуацию частичного согласия на передачу данных и последующее обновление профиля. Наконец, в отличие от логина по паролю, при входе через ЕСИА платформе важно корректно отрабатывать деградацию внешнего контура: временную недоступность, рост задержек, частичные ошибки при получении ключей подписи и нестабильность сетевого маршрута. Это переносит часть тестовых задач из области функционального тестирования в область тестирования надежности и устойчивости. Автоматизация здесь особенно важна: воспроизвести редкую комбинацию таймаутов и повторов вручную практически невозможно.

Важно разграничить понятия: тестирование интеграции с реальным контуром ЕСИА и тестирование корректности реализации протокола в коде платформы. Первое дает уверенность в совместимости окружений и настройках (в том числе URI, сертификаты, доступность), но плохо воспроизводимо: оно зависит от внешних факторов, не подходит для ежедневного прогона и ограничено условиями доступа. Второе позволяет строго контролировать входные данные, симулировать ошибки и строить систематические негативные проверки, но требует корректного моделирования поведения провайдера. Из этого следует рациональная стратегия: основную массу автоматизированных проверок нужно строить вокруг имитации (mock-провайдера) или тестового стенда провайдера, а реальные интеграционные прогоны оставить как периодическую верификацию совместимости. При этом

задача имитации – воспроизвести существенные свойства ЕСИА как OpenID Connect-провайдера на уровне, достаточном для проверки безопасности и корректности. Это означает, что тестовый контур должен уметь выпускать подписанные JWT, публиковать набор публичных ключей (JWKS), поддерживать ротацию ключей, выдавать различные варианты claims и управлять жизненным циклом authorization code так, чтобы его нельзя было использовать повторно. На практике оптимально формировать тестовую пирамиду таким образом, чтобы нижний уровень проверял чистую валидацию токена как функцию (к примеру, валидация подписи и claims на уровне библиотек), средний уровень проверял серверный обмен code-token и обработку ошибок внешнего API, а верхний уровень проверял «конец-в-конец» сценарий в браузере с реальными перенаправлениями и cookie. Подобный подход позволяет обнаруживать дефекты в наиболее ранней точке, а также локализовать причину: ошибка в криптографической проверке не должна маскироваться проблемой фронтенд-навигации, и наоборот.

Ключевой вопрос выбранной методики – насколько имитация провайдера соответствует существенным свойствам реального взаимодействия. Если mock выдает упрощенный токен без подписи, тесты превращаются в проверку «обработки JSON», что не дает научно значимого результата. Поэтому модель провайдера должна быть криптографически полноценной: токен подписывается приватным ключом, а платформа проверяет подпись по опубликованному JWKS. Это позволяет воспроизводить типовые случаи: неверная подпись, подпись другим ключом, отсутствие нужного ключа в JWKS, ситуация ротации ключей, когда JWKS обновился, а кэш платформы еще нет. С научной точки зрения важно контролировать параметры времени. Типичный скрытый дефект – некорректное использование системного времени сервера при проверке exp/iat, особенно в контейнерных окружениях или при рассинхронизации времени на стендах. Тестовый контур должен позволять управлять «текущим временем» в токенах и создавать ситуации, где токен формально валиден, но пограничен по времени.

Такие проверки выявляют ошибки округления, неверно настроенный допустимый сдвиг и дефекты кэширования сессий. Отдельного внимания требует связка state-nonce (рис.2). В реальных атаках злоумышленник стремится предоставить платформе «корректный» authorization response, полученный в другом контексте. Если платформа не привязывает state к инициирующей сессии и не проверяет nonce, возникает окно для подмены входа. Подобные дефекты трудно выявить, если тест ограничивается успешной авторизацией.

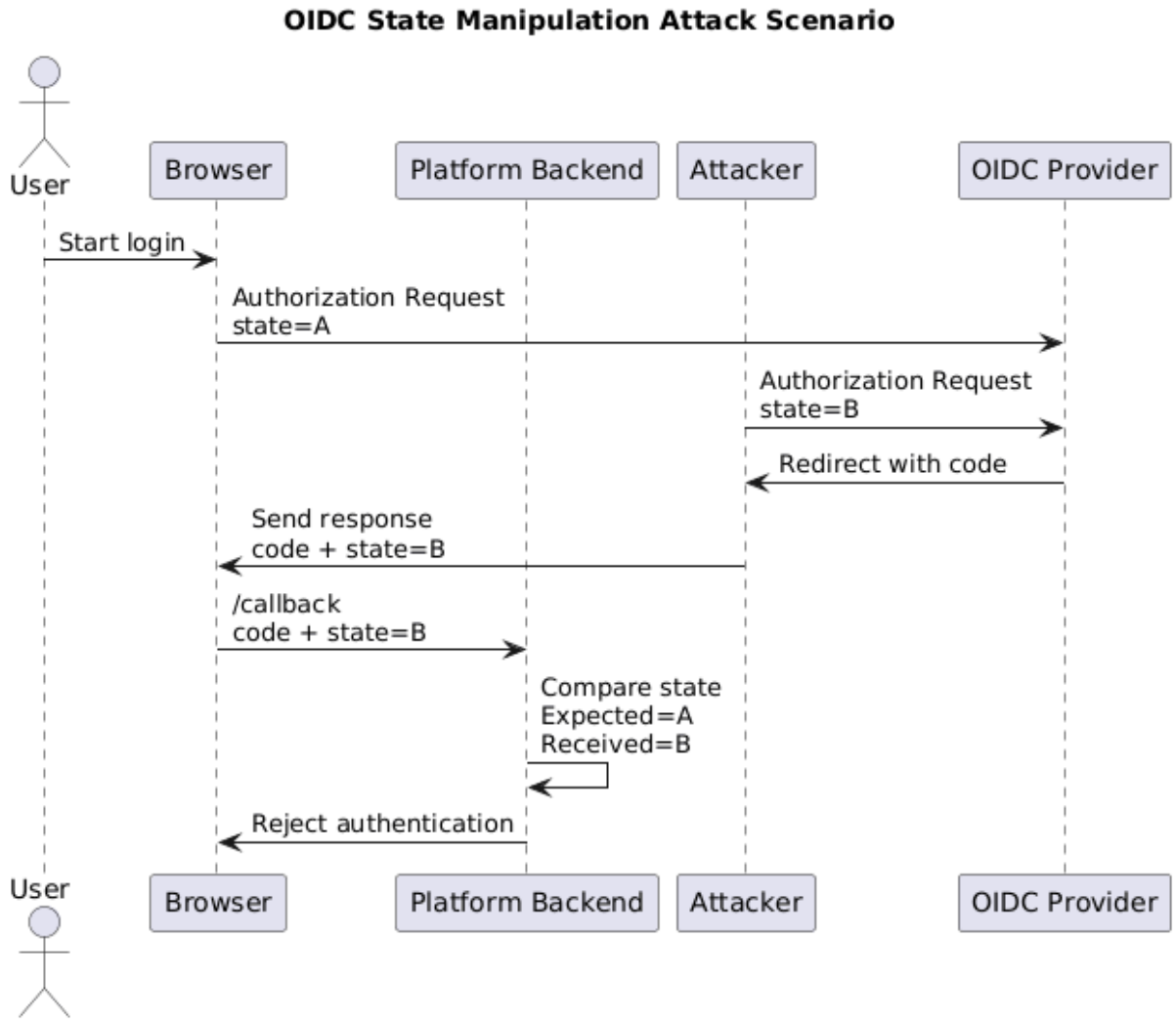

Рисунок 2 – Моделирование атаки подмены параметра state

В предложенной методике такие ситуации моделируются как последовательность: инициируется запрос входа, затем возвращается ответ с

корректным code, но с чужим state или nonce. Ожидаемое поведение – отказ в создании сессии и запись события безопасности. Поскольку предметная область предполагает разные уровни доступа, тестовый контур должен уметь выпускать id_token с различными атрибутами пользователя и уровнями подтверждения, чтобы платформа могла корректно отрабатывать правила бизнес-политики. Неправильная интерпретация claims в такой системе приводит к неочевидным дефектам: пользователь вошел правильно, но получил роль не своего типа, а значит – увидел чужие контакты или получил доступ к публикации. В методике это отражается в тестах, связывающих id_token и локальную модель; пользователя и набор прав.

При автоматизированном тестировании входа через ЕСИА необходимо разделить проверку протокольного взаимодействия и проверку пользовательского потока. Протокольная часть выполняется между backend платформы и провайдером и должна тестироваться на уровне HTTP-клиента/серверных компонентов. Это позволяет точно контролировать параметры запросов и ответов, коды ошибок и ретраи. В этой части проверяется, что платформа правильно обменивает authorization code на токены только один раз, корректно обрабатывает ошибки типа invalid_grant, распознает отсутствие нужного ключа подписи и не принимает токен, не проходящий по issuer/audience. Пользовательская часть выполняется в браузере и имеет другой источник дефектов: неверные redirect URI, проблемы с cookie-политикой (SameSite), несовпадение доменов, потеря параметров state при навигации, влияние блокировщиков. Здесь потребуется E2E-автоматизация, но она должна быть экономной и целевой: проверять именно наличие ключевых свойств безопасности и целостности состояния. В платформе аренды критично подтвердить, что после завершения входа пользователь не оказывается в промежуточном «полуаутентифицированном» состоянии, когда UI считает его вошедшим, а backend – нет. Такое различие типично при неправильном управлении сессионными cookie и токен-storage. Отдельный класс дефектов проявляется в конкурентных условиях. Сценарии включают двойной клик по кнопке входа,

повторную отправку запроса браузером, одновременное открытие вкладок. Если backend не делает обмен code и token идемпотентным и не защищает хранилище одноразовых кодов, возникают гонки, которые влекут случайные ошибки входа или, хуже, некорректную привязку сессии. В методике эти случаи воспроизводятся нагрузочными и стохастическими тестами: множество параллельных попыток завершить вход с одним и тем же code, а также серия входов в коротком интервале времени с различными code, чтобы проверить отсутствие «перемешивания» сессий. Надежность требует тестировать поведение при деградации. Это означает, что тесты должны уметь вводить задержки и ошибки на стороне mock-провайдера: таймауты, медленный ответ JWKS, временная недоступность token endpoint. При этом правильная реакция платформы зависит от ее архитектуры. Если платформа кэширует JWKS, важно убедиться, что она использует кэш при кратковременной недоступности и не принимает токены с неизвестным ключом без разбора. Если реализованы повторы запросов, стоит проверить, что повторы ограничены, а ошибки правильно классифицируются (повторять можно сетевые таймауты, но бессмысленно повторять invalid_grant). На уровне научного описания это оформляется как проверка стратегии деградации: что система делает при временной недоступности внешнего компонента, и как это влияет на пользователя и безопасность.

Методика была апробирована на прототипе веб-платформы аренды недвижимости с выделенным backend-компонентом аутентификации и отдельным модулем управления пользователями. В тестовом контуре реализована имитация OIDC-провайдера с выпуском подписанных JWT и публикацией JWKS, а также набором управляемых отказов (рис.3). Для обеспечения воспроизводимости тестирования используется контейнеризированный тестовый контур, развернутый с использованием Docker Compose. Архитектура стенда включает следующие компоненты:

1. oidc-mock-provider — сервис имитации провайдера OpenID Connect; реализует endpoints /authorize, /token, /jwks.json.
2. auth-backend — backend-компонент платформы, выполняющий обмен authorization code на токены и их валидацию.
3. frontend — веб-клиент, инициирующий поток авторизации.
4. test-runner — контейнер с автоматизированными тестами.

Контейнер провайдера генерирует ключевую пару RSA и публикует открытый ключ через JWKS endpoint. Ротация ключей моделируется сменой значения KEY_ID и обновлением набора ключей, что позволяет воспроизводить сценарии устаревшего кэша JWKS на стороне платформы.

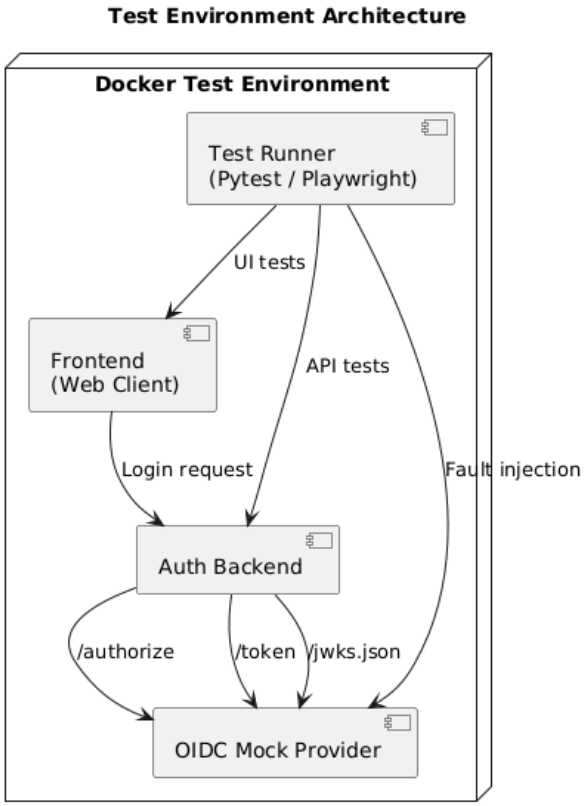

Рисунок 3 – Архитектура контейнеризированного тестового стенда

Особое внимание уделялось воспроизводимости: тесты выполнялись в контейнерном окружении, а параметры времени и ключей фиксировались в

конфигурации. Результаты в таблице 2 показали, что наибольшую практическую ценность дают тесты, которые систематически атакуют границы корректности: подмена state-nonce, ротация ключей, пограничные значения exp/iat, повторное использование authorization code.

Таблица 2 — Распределение обнаруженных дефектов

Тип дефекта	**Количество**
ошибки валидации времени токена	3
несогласованность кэша JWKS	2
ошибки обработки state/nonce	2
некорректная обработка повторного authorization code	1

Наиболее критичным оказался дефект обработки clock skew: при увеличении сетевой задержки более 800 мс часть токенов ошибочно считалась просроченной. Исправление заключалось в введении допустимого временного сдвига (CLOCK_SKEW=60s) и синхронизации времени контейнеров через NTP. В ходе апробации выявлялись дефекты, которые при ручном тестировании выглядели как «редкие сбои входа»: несогласованность кэша JWKS приводила к ситуационным отказам после смены ключа, а неправильная обработка clock skew давала спорадические ошибки у части пользователей при нагрузке и увеличении задержек. Также обнаруживались логические дефекты доменной привязки: при определенной комбинации claims платформа создавала локального пользователя с неверным уровнем доступа, что проявлялось только в связке «вход, получение профиля, проверка прав». Значимым наблюдением является то, что корректность аутентификации в распределенной системе имеет многокомпонентный характер: криптографическая валидность токена необходима, но недостаточна. Не менее важны и корректность управления состояниями (state-nonce, одноразовость code) и устойчивость к деградации внешнего провайдера. Автоматизация позволяет формализовать эти свойства и превратить их в повторяемые проверки, что

повышает качество регрессии и снижает вероятность дефектов, уходящих в эксплуатацию.

Аутентификация через ЕСИА в веб-платформе аренды недвижимости является объектом тестирования, где ошибки проявляются на стыке протоколов, криптографии, сетевого взаимодействия и доменной модели доступа. Предложенный подход к автоматизированному тестированию опирается на воспроизводимую имитацию внешнего провайдера с криптографически полноценными токенами и управляемыми отказами, что позволяет проверять не только успешный вход, но и безопасность связки state/nonce, корректность валидации claims, одноразовость авторизационных кодов и поведение системы при деградации. Апробация на прототипе показала, что именно негативные и пограничные сценарии дают максимальный эффект по обнаружению дефектов, которые иначе воспринимаются как «нестабильность интеграции». Практическая ценность работы состоит в переносе тестирования входа через ЕСИА из разряда разрозненных ручных проверок в область систематически воспроизводимых экспериментов, совместимых с CI/CD и ориентированных на свойства безопасности и надежности, значимые для цифровых сервисов в сфере недвижимости.

Разработанная методика автоматизированного тестирования позволяет выявлять ошибки реализации протокола OpenID Connect на ранних этапах разработки и повышает надежность интеграции с ЕСИА в веб-платформах.

Список литературы

1. Дрянкова Д. А. Реализация аутентификации и авторизации в FastAPI с использованием OpenID Connect / Д. А. Дрянкова // Электронный научный журнал «Дневник науки». – 2025. – № 10. – С. 1–10.
2. Старосельский А. К. Авторизация и обмен данными в веб-приложениях с использованием протокола OAuth 2.0 / А. К. Старосельский // Научная

публикация. – 2023. – URL: https://cyberleninka.ru/article/n/avtorizatsiya-i-obmen-dannymi-v-veb-prilozheniyah-s-ispolzovaniem-protokola-oauth-2-0

3. Ляшов Е. И. Виды аутентификации в современных веб-приложениях / Е. И. Ляшов // Научная статья. – 2024. – URL: https://cyberleninka.ru/article/n/vidy-autentifikatsii-v-sovremennyh-veb-prilozheniyah
4. Сайидов Ф. А. Внедрение единой системы авторизации и ее безопасность: интеграция с OAuth 2.0 и OpenID Connect / Ф. А. Сайидов // Научная публикация. – 2024. – URL: https://cyberleninka.ru/article/n/vnedrenie-edinoy-sistemy-avtorizatsii-v-obrazovatelnuyu-ekosistemu-integratsiya-i-bezopasnost
5. Ястребков А. С. Аутентификация и авторизация пользователей в распределенных микросервисных приложениях / А. С. Ястребков // Научная статья. – 2020. – URL: https://elibrary.ru/item.asp?id=44307683
6. Dalimunthe S., Reza J., Marzuki A. RESTful API Security Using JSON Web Token (JWT) with HMAC-SHA512 Algorithm in Session Management // Journal of Applied Engineering and Technological Science. – 2026. – Vol. 3, No. 2.
7. Fett D., Küsters R., Schmitz G. A comprehensive formal security analysis of OAuth 2.0 / D. Fett, R. Küsters, G. Schmitz // arXiv. – 2016
8. Rana M., Pandey A., Mishra A., Kandu V. Enhancing Data Security: A Comprehensive Study on the Efficacy of JSON Web Token (JWT) and HMAC SHA-256 Algorithm for Web Application Security // International Journal on Recent and Innovation Trends in Computing and Communication. – 2024. – Vol. 11, Issue 9. – P. 4409–4416.
9. Hardt, D. The OAuth 2.0 Authorization Framework [Электронный ресурс] / D. Hardt // RFC 6749. – IETF, 2012. – URL: https://datatracker.ietf.org/doc/html/rfc6749
10. Jones, M. JSON Web Token (JWT) [Электронный ресурс] / M. Jones, J. Bradley, N. Sakimura // RFC 7519. – IETF, 2015. – URL: https://datatracker.ietf.org/doc/html/rfc7519

11. Sakimura, N. OpenID Connect Core 1.0 [Электронный ресурс] / N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, C. Mortimore. – OpenID Foundation, 2014. – URL: https://openid.net/specs/openid-connect-core-1_0.html
12. Sakimura, N. Proof Key for Code Exchange by OAuth Public Clients [Электронный ресурс] / N. Sakimura, J. Bradley, N. Agarwal // RFC 7636. – IETF, 2015. – URL: https://datatracker.ietf.org/doc/html/rfc7636
13. Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации. Единая система идентификации и аутентификации (ЕСИА): документация по интеграции информационных систем [Электронный ресурс]. – URL: https://esia.gosuslugi.ru