Реактивное управление идентификацией и контролем доступа в контейнерных средах на базе прокси-сервера Envoy

И.А. Зянчурин, О.Р. Лапонина

Аннотация **– В контейнерных и микросервисных системах доступ всё чаще предоставляется не к одиночному запросу, а к долгоживущему потоку HTTP/2(gRPC, Google Remote Procedure Call). В таких условиях авторизация только в момент открытия соединения не гарантирует своевременное прекращение доступа после события отзыва.**

Представлен реализованный реактивный механизм управления идентификацией и доступом для сервисной сетки на базе прокси-сервера Envoy.

Локальный провайдер идентификации публикует метаданные OIDC (OpenID Connect) и JWKS (JSON Web Key Set) и выдаёт токен доступа с полями sub, sid и jti, то есть с идентификаторами субъекта, сессии и токена. Благодаря этому все итоговые сценарии используют один и тот же подход к идентификации, а сравнение не опирается на заранее зафиксированные идентификаторы. Envoy принимает стандартизованные события Shared Signals Framework (SSF) и Continuous Access Evaluation Profile (CAEP) в формате Security Event Token (SET), нормализует идентификаторы субъекта, сессии и токена (sub, sid, jti), хранит состояние запрета в Redis и адресно завершает уже активный поток gRPC в плоскости данных Envoy.

Для оценки построен воспроизводимый Kubernetes-стенд с единым локальным провайдером идентификации, публикующим метаданные OIDC/JWKS, и сравнением с тремя архитектурами с открытым исходным кодом: OPA (Open Policy Agent) + Envoy, Istio CUSTOM с режимом подключения внешней авторизации в Istio, и OpenFGA. Основная серия включает 360 запусков: четыре архитектурных подхода, три профиля нагрузки и 30 повторов на конфигурацию. Разработанный подход во всех повторах завершал уже активный поток, обеспечивал полный запрет повторного открытия после отзыва доступа, удерживал окно риска на уровне 0 сообщений в низком и среднем профилях и около 0.9 сообщения в высоком профиле, тогда как сравниваемые архитектуры блокировали только повторное открытие. Практическая значимость работы состоит в воспроизводимом способе перевода стандартизованных событий безопасности в немедленное принуждение политики на уровне сетевого прокси.

Ключевые слова **– реактивная авторизация, сервисная сетка, Envoy, SSF/CAEP/SET, gRPC, управление использованием, Kubernetes.**

Статья получена 22 апреля 2026.
И.А. Зянчурин – МГУ имени М.В. Ломоносова (email: ingv0rr@yandex.com).
О.Р. Лапонина – МГУ имени М.В. Ломоносова (email: laponina@oit.cmc.msu.ru).

I. Введение

Современные микросервисные системы всё чаще разрабатываются с использованием сервисной сетки, которая реализует сетевые и защитные функции в плоскости данных [1, 2, 3]. В такой архитектуре широко используются долгоживущие потоки gRPC (Google Remote Procedure Call) поверх HTTP/2. Если контекст безопасности меняется уже после открытия потока, классическая авторизация в момент запроса перестаёт быть достаточной: доступ уже должен быть прекращён, но фактически поток ещё продолжает существовать.

В работах по управлению использованием ресурсов (usage control) и непрерывной авторизации (continuous authorization) отмечается, что контроль доступа в динамической среде должен рассматриваться как процесс, продолжающийся и после первоначального предоставления доступа [7, 8, 9, 11]. Для таких систем значимым становится не только бинарный ответ «разрешить или запретить» в момент создания потока, но и время между событием отзыва разрешения и фактическим прекращением текущего использования ресурса.

Стандарты Shared Signals Framework (SSF) и Continuous Access Evaluation Profile (CAEP) вводят интероперабельный событийный слой, а формат Security Event Token (SET) делает возможным передачу таких событий между разными системами [4, 5, 6]. Однако сами стандарты описывают сигнал, а не механизм его немедленного применения в плоскости данных. Для уменьшения временного лага между событием безопасности и реализацией требуемых действий в плоскости данных предназначен предлагаемый механизм.

На практике эта задача становится особенно острой для долгоживущих потоковых взаимодействий. После разрешения потока система может в течение заметного времени продолжать передавать данные, хотя политика доступа уже изменилась. Поэтому важен не только факт блокировки открытия нового потока, но и время, за которое событие изменения прав доступа действительно достигает уже активного потока.

SSF задаёт общую модель обмена сигналами безопасности между системами, CAEP специализирует эту модель для событий, которые должны повлиять на уже выданный доступ, а SET задаёт форму представления самого события в виде подписанного токена.

В статье описан реактивный механизм управления идентификацией и доступом для сервисной сетки на базе прокси-сервера Envoy. В статье также показано, что событийно-управляемое принуждение выполнения требуемых действий над уже активными gRPC-потоками уменьшает окно риска по сравнению с распространёнными архитектурами авторизации с открытым исходным кодом.

Рассматриваемая постановка задачи является достаточно узкой. Наша цель состояла не в изобретении нового типа событий или в создании ещё одного движка авторизации, а в минимизации времени между возникновением стандартного события безопасности и его реализацией в плоскости данных для уже открытого потока.

Разработанный механизм состоит в следующем:

• реализован реактивный контур: приёмник событий, Redis-слой состояния и модуль принуждения в Envoy со связыванием активного потока по идентификаторам **`sub`**, **`sid`** и **`jti`**;

• построен единый воспроизводимый Kubernetes-стенд с локальным провайдером идентификации, публикующим OIDC/JWKS-метаданные;

• выполнено сравнение с тремя открытыми архитектурами: OPA (Open Policy Agent) + Envoy, Istio CUSTOM с режимом подключения внешней авторизации, и OpenFGA;

• выполнено краткое формальное описание на TLA+ (Temporal Logic of Actions) и проверена корректность базовых свойств.

II. Существующие решения

Shared Signals Framework (SSF) и Continuous Access Evaluation Profile (CAEP) используются как стандартизованный входной слой. SSF задаёт общую модель обмена, описывает источник и субъект события и определяет требования к передаче сигнала между компонентами. CAEP использует эту модель для событий, влияющих на продолжение уже выданного доступа, а SET определяет формат представления самого события как подписанного токена [4, 5, 6]. Тем самым в связке SSF отвечает за общий протокол и структуру обмена, CAEP – за семантику событий пересмотра доступа, а SET – за форму конкретного сообщения.

Проблема непрерывного пересмотра доступа подробно рассмотрена в литературе по управлению использованием ресурсов (usage control). Базовая работа Sandhu и Park вводит идею контроля использования как расширения традиционного управления доступом, а более поздняя линия UCON+ и обзор подходов к usage control показывают, что в центре внимания должны находиться не только политики доступа, но и механизмы их исполнения [7, 8, 9]. Работа по ACE и уведомлениям об отзыве прав отдельно подчёркивает значимость времени распространения события отзыва и интервала рассогласования между политикой и фактическим доступом [10, 11].

Из этих работ следуют два принципиальных вывода. Во-первых, контроль доступа должен продолжаться после первоначального разрешения для отслеживания изменения состояния субъекта, условий внешнего окружения или возникновения события отзыва. Во-вторых, практически значимой метрикой становится не только ответ системы на новый запрос, но и длительность рассогласования между уже изменившейся политикой и фактически продолжающимся доступом.

Для контейнерных приложений сервисная сетка рассматривается как естественная инфраструктура, в которую вынесены защитные функции и реализация авторизации на уровне прокси [1, 2, 3]. Поэтому точка применения политики в прокси-сервере Envoy логически соответствует как облачной микросервисной архитектуре, так и требованиям, чтобы управление доступом осуществлялось без модификации бизнес-сервисов.

В существующих решениях с открытым кодом OPA + Envoy реализует внешнюю авторизацию в момент запроса [12]; Istio CUSTOM, то есть режим подключения внешнего сервиса авторизации в сервисной сетке Istio, переносит ту же идею в сервисную сетку [13]; OpenFGA представляет способ централизованной детализированной авторизации, восходящий к Zanzibar, то есть модели отношений доступа [14, 15]. Тем самым существующие решения представляют три архитектурных подхода: прокси-авторизацию, авторизацию в сервисной сетке и централизованную детализированную авторизацию. Разработанный механизм добавляет к этой технологической базе реактивное адресное завершение уже активного gRPC-потока в плоскости данных.

Для сравнения выбраны именно три архитектуры с открытым исходным кодом, представляющие три класса решений: авторизация на уровне прокси, авторизация в сервисной сетке и централизованная детализированная авторизация. Такой выбор позволяет обсуждать влияние архитектуры на результат, а не только особенности конкретных реализаций.

Для формального описания свойств разработанного механизма и проверки корректности этих свойств используется TLA+ (Temporal Logic of Actions), язык спецификации дискретных систем и проверки их свойств с помощью model checking. TLA+ применяется для проверки модели переходов потока после события отзыва и тем самым усиливает экспериментальную часть, доказывая ее корректность [16, 8].

III. Архитектура системы

A. Общая архитектура системы

Разработанная система состоит из локального провайдера идентификации, приёмника событий, Redis как хранилища данных типа «ключ–значение», реактивного модуля Envoy и сервиса gRPC, а также нагрузочного клиента. Локальный провайдер идентификации публикует метаданные OIDC (OpenID Connect) и JWKS (JSON Web Key Set) и выдаёт токен доступа с полями **`sub`**, **`sid`** и **`jti`**, то есть с идентификаторами субъекта, сессии и токена. Благодаря этому все итоговые сценарии используют один и тот же

подход к идентификации, а сравнение не опирается на заранее зафиксированные идентификаторы.

B. Приёмник событий и Redis как компонент поддержки состояния и доставки событий

Приёмник событий принимает SSF/CAEP/SET-событие, проверяет структуру полезной нагрузки, извлекает идентификаторы **`sub`**, **`sid`** и **`jti`** и записывает новое состояние в Redis. Redis используется не только как внешнее хранилище, но и как единый слой оперативного состояния и доставки событий. Во внешнем хранилище поддерживаются ключи запрета вида **`deny:sub:<value>`**, **`deny:sid:<value>`** и **`deny:jti:<value>`** с TTL, а для дедупликации событий используется отдельный ключ вида **`event:<event_id>`**. После записи состояния событие публикуется в канал **`auth_events`**. В результате Redis одновременно играет роль низколатентного хранилища запрета и слоя доставки сигнала отзыва.

C. Реактивный модуль Envoy

Реактивный модуль реализован в Envoy как расширение для HTTP/2(gRPC)-потоков. В момент открытия потока модуль извлекает идентификатор из уже проверенного токена доступа, проверяет соответствие между идентификатором потока и набором **`sub`**, **`sid`** и **`jti`** во внутреннем состоянии исполнения Envoy и добавляет поток во внутренний индекс активных потоков. При получении события из **`auth_events`** модуль обновляет локальный кэш отозванных идентификаторов, находит совпадающие активные потоки по **`sub`**, **`sid`** и **`jti`** и адресно завершает их средствами самого Envoy, без участия бизнес-логики сервиса.

Разделение состояния между Redis и Envoy принципиально. Во внешнем хранилище сохраняется запрет, который нужен и для контроля повторного открытия, и для выполнения сравнения с другими архитектурными подходами. Во внутреннем состоянии Envoy поддерживается только список активных потоков, поскольку именно он нужен для немедленного воздействия на уже идущее сетевое взаимодействие. Такая схема соответствует прокси-ориентированному принуждению и не требует постоянного хранения списка завершённых потоков вне плоскости данных.

Рисунок 1. Общая архитектура разработанной системы

D. Связывание потока с идентификатором и адресное завершение потока

Связывание потока с идентификатором использует уже верифицированный токен. После успешной проверки подписи и извлечения утверждений токена модуль получает значения **`sub`**, **`sid`** и **`jti`**, нормализует их и использует как ключи для регистрации активного потока. Такое решение позволяет одинаково обрабатывать как сценарий немедленного завершения текущего потока, так и сценарий последующего отказа при новой попытке доступа.

Завершение активного потока производится с помощью самого Envoy. После обнаружения совпадения между событием отзыва и локальным индексом модуль инициирует адресное прекращение передачи для соответствующего потока. Это важно с архитектурной точки зрения: вышестоящее приложение не должно знать о деталях политики отзыва и не должно самостоятельно реализовывать обработку каждого типа сигнала безопасности.

IV. Экспериментальный стенд и методика сравнения

Экспериментальный стенд развёрнут в локальном Kubernetes-кластере kind. Во всех режимах использовались один и тот же gRPC-сервис, один нагрузочный клиент, одинаковая топология «доступа клиента через Envoy к gRPC-сервису» и одинаковый сценарий события отзыва через локальный провайдер идентификации. Это позволяет интерпретировать различия именно как различия архитектуры принуждения [10, 17].

Одинаковая топология принципиальна для корректности полученных результатов. Во всех архитектурных подходах использовались один и тот же вышестоящий сервис, одинаковый тип токена, один и тот же момент отправки события отзыва и единое окно

наблюдения. Поэтому в эксперименте менялась не прикладная логика сервиса, а именно архитектурные решения исполнения политики доступа.

Аппаратно-программная среда была фиксирована, поскольку метрики считались в миллисекундах. Эксперименты выполнялись в среде WSL2 на машине с процессором AMD Ryzen 9 7900X и 96 ГБ оперативной памяти DDR5-6800. Внутри WSL2, Docker и кластера kind были доступны 24 логических вычислительных потока и около 46 ГБ памяти.

Каждый прогон строился одинаково: клиент открывал поток, система выходила на устойчивую передачу, затем в фиксированный момент **`t_revoke`** публиковалось событие отзыва сессии типа **`session-revoked`**, после чего измерялись время завершения текущего потока, число сообщений после события отзыва и результат повторной попытки открыть поток с теми же идентификаторами.

Рассматривались четыре архитектурных решения: OPA (Open Policy Agent) + Envoy, Istio CUSTOM, OpenFGA и разработанный реактивный модуль. Использовались три варианта нагрузки: низкий – 1 сообщение / 200 мс, средний – 1 сообщение / 50 мс, высокий – 1 сообщение / 10 мс. На каждую комбинацию «архитектурное решение и вариант нагрузки» приходилось 30 повторов; тем самым основная таблица сравнения содержит 360 запусков.

Были выполнены также дополнительные локальные проверки чувствительности, однако они не рассматриваются как самостоятельные проверки. Их задача – подтвердить интерпретацию основного эффекта, но не заменять сравнение с реальными архитектурами.

Основными метриками были задержка применения политики (latency-to-enforce), окно риска Δ в миллисекундах, окно риска Δ по числу сообщений после события отзыва, доля запрещённых повторных открытий (post-revoke deny), частота ложных остановок (false termination rate), частота пропущенных отзывов (missed revocation rate) и накладные расходы без учета событий. Такой выбор метрик соответствует работам этого типа по continuous authorization и уведомлениям об отзыве прав, где анализируется не только исходное решение о доступе, но и продолжительность остаточного доступа после изменения контекста [10, 9, 11].

Используемые метрики интерпретируются следующим образом. Задержка применения политики показывает время между событием отзыва и фактическим исполнением запрета. Δ в миллисекундах характеризует временную длину окна риска, а Δ по числу сообщений рассматривает тот же эффект с точки зрения семантики самого потока. Метрика post-revoke deny отвечает на вопрос, умеет ли система блокировать повторное открытие. Наконец, частота ложных остановок и частота пропущенных отзывов характеризует, не только скорость отзыва доступа, но и корректность такого отзыва.

Такая методика одновременно проверяет основную гипотезу и практическую пригодность разработанного механизма. Реактивный подход должен не только уменьшать окно риска, но и завершать уже активный поток без генерации ложных остановок при отсутствии соответствующего события.

Статистика собиралась по 30 повторам на конфигурацию. Для каждой метрики сохранялись результаты прогонов, после чего вычислялись средние значения, медианы (p50), 95-й перцентиль (p95) и 95%-доверительные интервалы. Приводятся только наиболее информативные агрегаты, тогда как полные сырые результаты и вспомогательные отчёты вынесены в открытый репозиторий проекта.

Принципиально важно различать основную серию и полный экспериментальный корпус. Основная серия экспериментов, на которой построены выводы статьи, содержит 360 запусков и сравнивает только четыре архитектурных решения. Полный корпус включает 540 прогонов, поскольку дополнительно содержит внутренние проверки чувствительности. Такое разграничение позволяет одновременно обеспечивать прозрачность основного сравнения и иметь более широкий набор вспомогательных проверок.

Таблица 1. Архитектуры сравнения и профили нагрузки

Архитектурное решение	Описание	Профили нагрузки
OPA + Envoy	Авторизация при открытии потока через встроенный механизм Envoy ext_authz	низкий / средний / высокий
Istio CUSTOM	Внешняя авторизация в сервисной сетке	низкий / средний / высокий
OpenFGA	Централизованная детализированная авторизация	низкий / средний / высокий
Разработанный реактивный подход	Адресное завершение уже активного потока в Envoy	низкий / средний / высокий

V. Результаты

Основной результат показан на рис. 2. Во всех профилях нагрузки разработанный реактивный режим имеет существенно меньшее окно риска, чем остальные три архитектурных решения. Причина этого кроется в различных архитектурных подходах: разработанный механизм может воздействовать на уже активный поток, тогда как OPA + Envoy, Istio CUSTOM и OpenFGA принимают решение только при получении нового запроса или при повторном открытии.

Для разработанного режима во всех 90 прогонах наблюдалось завершение активного потока после получения события отзыва, а доля запрещённых повторных открытий составила 1.0. Средняя задержка применения политики находилась на уровне 256.5 мс в профиле высокой нагрузки, 256.7 мс в профиле средней нагрузки и 256.7 мс в профиле низкой нагрузки. Окно риска Δ по числу сообщений было равно 0 в профилях низкой и средней нагрузки и 0.9 сообщения в профиле высокой нагрузки.

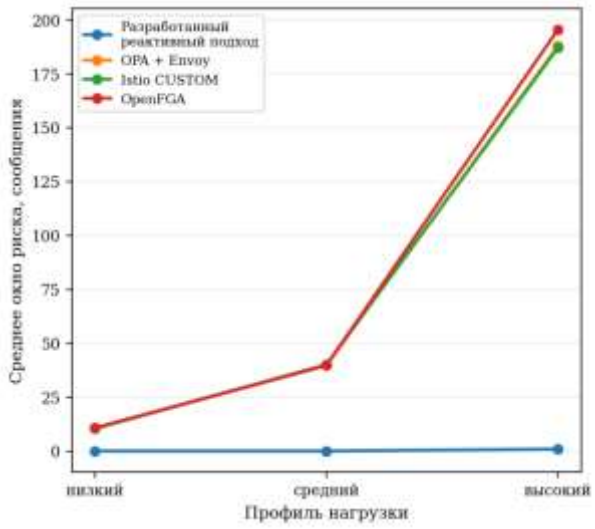

Рисунок 2. Сравнение окна риска между разработанным реактивным подходом и основными открытыми архитектурами

Существенно, что преимущество сохранялось во всех трёх профилях нагрузки. В профиле низкой нагрузки среднее окно риска составило 256.7 мс при нулевом числе сообщений после события отзыва; в профиле средней нагрузки – 256.7 мс, также без сообщений после события отзыва; в профиле высокой нагрузки – 256.5 мс и примерно одно сообщение. Это показывает, что разработанный механизм остаётся эффективным и при редкой, и при интенсивной передаче данных.

Если сопоставить эти значения с профилем потока, различие становится ещё нагляднее. В профилях низкой и средней нагрузки реактивный подход практически успевает применить событие отзыва до прохождения следующего сообщения. В профиле высокой нагрузки из-за частоты порядка 100 сообщений в секунду после события отзыва в среднем успевает пройти около одного сообщения, что по-прежнему на порядки меньше, чем в других архитектурных решениях.

Для архитектурных решений, с которыми проводилось сравнение, картина оказалась иной. Все они воспроизводимо запрещали повторное открытие после поступления события отзыва, но не прерывали уже активный поток в пределах окна наблюдения 2 с. В профиле высокой нагрузки среднее окно риска по числу сообщений составило 188.0 сообщения для OPA + Envoy, 187.1 для Istio CUSTOM и 195.6 для OpenFGA. По времени это соответствовало окну риска около 2021–2025 мс, то есть практически полному окну наблюдения.

Эта разница не сводится к наличию или отсутствию внешнего движка авторизации. И OPA + Envoy, и Istio CUSTOM, и OpenFGA корректно реализуют отказ при новой попытке доступа после события отзыва, однако уже ранее разрешённый поток остаётся вне зоны немедленного воздействия. Следовательно, решающим фактором является не просто внешняя точка принятия решения, а наличие механизма, который умеет адресно завершать уже активный поток в плоскости данных.

С практической точки зрения это означает, что авторизация в момент запроса решает задачу последующего отказа в новом открытии, но не задачу минимизации остаточного доступа в уже идущем сеансе. Именно поэтому в основной серии сопоставляются разные OSS-архитектуры авторизации, а не только локальные варианты одной и той же реализации.

В профилях меньшей нагрузки различие сохраняется на том же качественном уровне. При 5 сообщениях в секунду режимы сравнения оставляли после события отзыва примерно по 10–11 сообщений, а при 20 сообщениях в секунду – около 40 сообщений. Это показывает, что проблема не связана только с экстремальной нагрузкой; она возникает в любом случае, когда система не умеет воздействовать на уже активный поток между двумя новыми точками авторизации.

Дополнительные проверки корректности показали нулевую частоту ложных остановок и нулевую частоту пропущенных отзывов. По измерениям накладных расходов без событий реактивный вариант Envoy использовал около 19 мCPU и 18 МБ памяти, тогда как базовый Envoy – около 20 мCPU и 18 МБ. Следовательно, уменьшение окна риска не достигается ценой существенного роста фоновых накладных расходов.

Итоговые сценарии с локальным провайдером идентификации подтвердили ту же закономерность на полном пути с токеном доступа. Разработанный механизм завершал текущий поток и запрещал повторное открытие, тогда как при других архитектурных решениях текущий поток оставлялся активным, но отклонялись новые попытки доступа. Это исключает объяснение результатов через искусственно заданные идентификаторы и показывает, что эффект сохраняется в полном контуре аутентификации и авторизации.

Основной результат статьи не сводится к сокращению задержек. Ключевой вывод состоит в том, что только архитектура с адресным воздействием в плоскости данных на уже активный поток устраняет сам механизм накопления остаточного доступа после события отзыва. Все архитектурные решения обеспечивают последующий запрет доступа, но не немедленное прекращение уже начатого использования ресурса.

Таблица 2. Ключевые результаты в профиле высокой нагрузки (100 сообщений/с)

Архитектурное решение	Поток завершён	Δ, мс	Δ, сообщения	Повторное открытие запрещено
Разработанный реактивный подход	да	256.5	0.9	1.0
OPA + Envoy	нет	2021.3	188.0	1.0
Istio CUSTOM	нет	2027.9	187.1	1.0
OpenFGA	нет	2024.6	195.6	1.0

VI. Обсуждение

Полученные результаты позволяют чётко разделить две задачи, которые в практических системах часто

смешиваются. Первая задача – принять новое решение о доступе после события отзыва. С ней существующие архитектурные решения с открытым исходном кодом справляются корректно. Вторая задача – немедленно распространить это решение на уже начатое использование ресурса. Именно здесь проявляется отличие разработанного подхода от подходов авторизации в момент запроса, и именно этот эффект соответствует постановке, обсуждаемой в литературе по continuous authorization и распространению сигналов отзыва [10, 11, 8].

Разработанный подход следует трактовать как архитектурный слой реактивного принуждения, дополняющий существующие системы OPA, Istio или OpenFGA. Эти системы централизуют принятие решения, упрощают управление политиками и обеспечивают повторяемую авторизацию новых запросов. В рассматриваемой постановке различие связано с тем, что сама точка исполнения политики остаётся у них на входе запроса. Поэтому реактивный подход уместен прежде всего там, где требуется воздействовать на уже активный поток.

Значение имеет и выбранное разделение состояния. Redis хранит запрет и обеспечивает доставку события, а Envoy удерживает только локальный индекс уже активных потоков. Такая композиция делает внешний слой достаточно простым и переносимым между разными архитектурными решениями, но при этом оставляет самое срочное действие – завершение текущего потока – в плоскости данных. Это согласуется с представлением о сервисной сетке как об инфраструктуре, в которой общие защитные механизмы должны выноситься в прокси и применяться без участия бизнес-сервиса [1, 2, 3].

У описанного подхода есть и ограничения. Основной эксперимент выполнен на локальном стенде kind, состоящим из одного узла, поэтому не утверждается аппаратная или кластерная инвариантность миллисекундных значений. Кроме того, в основной серии использовалось ограниченное подмножество событий, прежде всего событие отзыва сессии **`session-revoked`**, а в сравнении рассматривались только три наиболее репрезентативные OSS-архитектуры. Эти ограничения не отменяют основного вывода статьи, но задают его корректные границы: исследуется воспроизводимая архитектурная разница между реактивным принуждением и авторизацией в момент запроса, а не полный охват всех возможных провайдеров идентификации, типов событий и топологий развёртывания.

Практическая полезность результата при этом остаётся высокой. Даже на тестовом стенде использование стандартизованных событий SSF/CAEP/SET совместно с Envoy и Redis позволяет построить воспроизводимый реактивный механизм управления идентификацией и доступом без изменения прикладного gRPC-сервиса. Поэтому разработанный механизм может служить исследовательским и инженерным шаблоном для тех контейнерных систем, где требуется сократить окно риска не только для новых запросов, но и для уже активных потоков.

С точки зрения переносимости архитектура остаётся достаточно общей. Приёмник событий можно связать не только с локальным провайдером идентификации, но и с внешним провайдером, если он публикует совместимые события или позволяет их нормализовать. Аналогично, внутренний механизм Envoy зависит не от конкретного бизнес-сервиса, а от того, что доступ реализован через HTTP/2(gRPC)-поток и идентичность может быть связана с **`sub`**, **`sid`** или **`jti`**. Тем самым показана не тестовая реализация, а переносимая архитектурная схема.

VII. Воспроизводимость артефактов

Отдельным результатом статьи является воспроизводимый набор программ и результатов экспериментов. В открытом репозитории опубликованы исходный код разработанного режима, конфигурации Kubernetes-стенда, сценарии запуска приёмника событий и локального провайдера идентификации, сырые CSV/JSON-результаты прогонов, агрегированные отчёты, графики и минимальный TLA+-артефакт [18].

Литература по авторизации в микросервисных архитектурах неоднократно подчёркивает дефицит открытых воспроизводимых исследований, в которых доступны не только итоговые числа, но и полный экспериментальный контур [17]. Воспроизводимость обеспечивается единым стендом, фиксированной матрицей архитектурных подходов и профилей, публикацией исходных результатов и явным разделением между основной серией сравнений и вспомогательными проверками чувствительности.

Благодаря этому любой исследователь в дальнейшем может повторить как минимум три уровня проверки: функциональную проверку реактивного прерывания потока, основную серию вычислительных экспериментов и минимальную формальную проверку свойств политики. Тем самым разработанная система выступает не только как предмет исследования, но и как готовый стенд для последующих сравнений и расширений.

VIII. Границы применимости подхода

Разработанный подход ориентирован прежде всего на те контейнерные системы, где доступ предоставляется не к одному короткому запросу, а к долгоживущему каналу взаимодействия. Именно в таких случаях возникает разрыв между моментом изменения политики и фактическим прекращением использования ресурса. Поэтому подход в первую очередь предназначен для потоковых gRPC- и других HTTP/2-сценариев, в которых после события отзыва может продолжаться передача данных без новой авторизации.

Напротив, для полностью короткоживущих запросов, где каждое взаимодействие и так проходит через новую проверку, преимущество реактивного завершения уже активного потока существенно меньше. В этом смысле разработанный механизм не подменяет традиционную внешнюю авторизацию, а дополняет её там, где одной проверки на входе недостаточно. Практически это

означает, что разработанный механизм следует применять не как универсальную замену существующим политикам доступа, а как специальный слой для долгоживущих взаимодействий.

Ещё одна граница применимости связана с событиями и идентичностью. Предложенный подход опирается на то, что событие отзыва может быть нормализовано к **`sub`**, **`sid`** или **`jti`** и что Envoy способен связать эти идентификаторы с уже открытым потоком. Если такая связь доступна, разработанный механизм переносим и на другие системы. Если же идентичность не доходит до прокси или изменение контекста не выражается через событие, пригодное для нормализации, то потребуется иная схема принуждения. Тем самым очерчивается корректная область, в которой полученные результаты следует считать обоснованными.

IX. Краткая формальная проверка

Для модели потока использовался автомат состояний с последовательностью «новый поток – активный поток – получено событие отзыва – завершённый или запрещённый поток». В этой модели фиксировались три свойства: после события отзыва уже допущенный поток должен быть завершён за конечное ограниченное время; после события отзыва повторное открытие потока с теми же идентификаторами должно оставаться запрещённым; без соответствующего события ложное завершение недопустимо. Такой способ формализации соответствует работам по применению TLA+ к usage control [16, 8]. Для минимального формального артефакта нарушения этих свойств не обнаружены. Формальная часть не подменяет вычислительный эксперимент, а подтверждает согласованность логики переходов, лежащей в основе реактивного подхода.

Первое свойство задаёт требование своевременного завершения уже существующего потока после события отзыва, второе – устойчивый запрет повторного открытия, а третье исключает ложное срабатывание без соответствующего события. В совокупности эти свойства покрывают именно те аспекты корректности, которые затем наблюдаются и в вычислительном эксперименте: прекращение текущего потока, отказ в новом открытии и отсутствие ошибочных остановок.

X. Заключение

Представлен разработанный реактивный механизм управления идентификацией и доступом для контейнерных сред на базе Envoy, который связывает стандартизованные SSF/CAEP/SET-события с уже активными gRPC-потоками и применяет политику непосредственно в плоскости данных. Реализованы приёмник событий, Redis-слой состояния и доставки, модуль принуждения в Envoy, локальный провайдер идентификации и воспроизводимый Kubernetes-стенд.

Вычислительные эксперименты показали, что разработанный режим завершает активный поток после события отзыва и существенно уменьшает окно риска по сравнению с OPA + Envoy, Istio CUSTOM и OpenFGA, которые блокируют только повторное открытие. Практическая значимость результата состоит в воспроизводимом способе перевода стандартизованных событий безопасности в немедленное принуждение на уровне сетевого прокси. Исходный код и экспериментальные артефакты опубликованы в открытом репозитории [18].

Благодарности

Авторы благодарят участников семинара кафедры Информационной безопасности за критику и ценные обсуждения. Вопросы кибербезопасности являются одним из основных научных направлений кафедры ИБ факультета ВМК МГУ имени М.В. Ломоносова и рассматривались во множестве магистерских диссертаций и научных работ [19, 20, 21]. Также, традиционно, отмечаем работы В.П. Куприяновского и его соавторов, положивших начало цифровой тематике в журнале INJOIT [22, 23].

Библиография

[1] Ferraiolo D., Gavrila S., Kuhn R. Service Mesh Proxy Models for Cloud-Native Applications. NIST SP 800-233. 2024.
[2] Nadalin A. et al. Building Secure Microservices-based Applications Using Service-Mesh Architecture. NIST SP 800-204A. 2021.
[3] Nadalin A. et al. Attribute-based Access Control for Microservices-based Applications Using a Service Mesh. NIST SP 800-204B. 2021.
[4] OpenID Foundation. OpenID Shared Signals Framework Specification 1.0. 2025.
[5] OpenID Foundation. OpenID Continuous Access Evaluation Profile 1.0. 2025.
[6] Hunt P., Jones M. Security Event Token (SET). RFC 8417. IETF. 2018.
[7] Sandhu R., Park J. Usage Control: A Vision for Next Generation Access Control. Mathematical Methods, Models, and Architectures for Computer Network Security. Springer. 2003.
[8] Park J., Sandhu R. et al. UCON+: Comprehensive Model, Architecture and Implementation for Usage Control and Continuous Authorization. Data and Applications Security and Privacy XXXVI. Springer. 2023.
[9] Ayoade G. et al. A Comprehensive Review of Usage Control Frameworks. Computer Standards & Interfaces. 2024.
[10] Gerdes S. et al. Using the ACE Framework to Enforce Access and Usage Control with Notifications of Revoked Access Rights. International Journal of Information Security. 2024.
[11] Joumaa H., Petrovska A., Hariri A., Dimitrakos T., Crispo B. Continuous Authorization Architecture for Dynamic Trust Evaluation. Trust Management XIV. Springer. 2024.
[12] Open Policy Agent. OPA-Envoy Plugin. 2026. URL: https://www.openpolicyagent.org/docs/latest/envoy-introduction/
[13] Istio Authors. External Authorization. 2025. URL: https://istio.io/latest/docs/tasks/security/authorization/authz-custom/
[14] OpenFGA Authors. OpenFGA Documentation. 2026. URL: https://openfga.dev/docs
[15] Pang R. et al. Zanzibar: Google's Consistent, Global Authorization System. USENIX Annual Technical Conference. 2019.
[16] Mouelhi T. et al. Specifying and Verifying Usage Control Models and Policies in TLA+. International Journal on Software Tools for Technology Transfer. 2021.
[17] Aloufi B. et al. Authentication and Authorization in Microservices Architecture: A Systematic Literature Review. Applied Sciences. 2022.
[18] Зянчурин И.А. Reactive Mesh AuthZ: исходный код, сценарии запуска и экспериментальные артефакты. Репозиторий GitHub. 2026. URL: https://github.com/Dark-Avery/Reactive-Mesh-AuthZ
[19] Ульби, Т. В., and О. Р. Лапонина. "Модуль управление доступом на основе атрибутов для веб-запросов из разных источников." International Journal of Open Information Technologies 11.5 (2023): 128-136.
[20] Ковтун, Д. П., and О. Р. Лапонина. "Использование управления доступом на основе атрибутов и mTLS в микросервисной архитектуре." International Journal of Open Information Technologies 13.6 (2025): 75-85.

[21] Намиот, Д. Е. Схемы атак на модели машинного обучения / Д. Е. Намиот // International Journal of Open Information Technologies. – 2023. – Т. 11, № 5. – С. 68-86. – EDN YVRDOB.
[22] Цифровая железная дорога - прогнозы, инновации, проекты / В. П. Куприяновский, Г. В. Суконников, П. М. Бубнов [и др.] // International Journal of Open Information Technologies. – 2016. – Т. 4, № 9. – С. 34-43. – EDN WIQHXX.
[23] Волков, А. А. О задачах создания эффективной инфраструктуры среды обитания / А. А. Волков, Д. Е. Намиот, М. А. Шнепс-Шнеппе // International Journal of Open Information Technologies. – 2013. – Т. 1, № 7. – С. 1-10. – EDN ROMIZX.

Reactive Identity and Access Control in Container Environments Based on Envoy Proxy Server

I.A. Zyanchurin, O.R. Laponina

***Abstract* – In containerized and microservice systems, access is increasingly granted not to a single request, but to a long-lived HTTP/2 (gRPC, Google Remote Procedure Call) stream. In such a context, authorization only at the moment a connection is opened does not guarantee timely termination of access after a revocation event. We present a reactive identity and access management mechanism implemented for a service mesh based on the Envoy proxy server. The local identity provider publishes OIDC (OpenID Connect) and JWKS (JSON Web Key Set) metadata and issues an access token with the sub, sid, and jti fields—that is, the subject, session, and token identifiers. This ensures that all end-user scenarios use the same authentication approach, and comparisons do not rely on pre-defined identifiers. Envoy accepts standardized Shared Signals Framework (SSF) and Continuous Access Evaluation Profile (CAEP) events in the Security Event Token (SET) format, normalizes subject, session, and token identifiers (sub, sid, jti), stores the denied state in Redis, and specifically terminates an active gRPC flow in the Envoy data plane. For evaluation, a reproducible Kubernetes rig was built with a single local identity provider publishing OIDC/JWKS metadata and compared against three open-source architectures: OPA (Open Policy Agent) + Envoy, Istio CUSTOM with Istio's external authorization mode enabled, and OpenFGA. The main series includes 360 runs: four architectural approaches, three load profiles, and 30 iterations per configuration. The developed approach terminated the active thread in all iterations, ensured complete blocking of reopening after access revocation, and kept the risk window at 0 messages in the low and medium profiles and approximately 0.9 messages in the high profile, while the compared architectures only blocked reopening. The practical significance of this work lies in the reproducible method for translating standardized security events into immediate policy enforcement at the network proxy level.**

***Keywords* – reactive authorization, service mesh, Envoy, SSF/CAEP/SET, gRPC, usage control, Kubernetes.**

References

[1]Ferraiolo D., Gavrila S., Kuhn R. Service Mesh Proxy Models for Cloud-Native Applications. NIST SP 800-233. 2024.
[2]Nadalin A. et al. Building Secure Microservices-based Applications Using Service-Mesh Architecture. NIST SP 800-204A. 2021.
[3]Nadalin A. et al. Attribute-based Access Control for Microservices-based Applications Using a Service Mesh. NIST SP 800-204B. 2021.
[4]OpenID Foundation. OpenID Shared Signals Framework Specification 1.0. 2025.
[5]OpenID Foundation. OpenID Continuous Access Evaluation Profile 1.0. 2025.
[6]Hunt P., Jones M. Security Event Token (SET). RFC 8417. IETF. 2018.
[7]Sandhu R., Park J. Usage Control: A Vision for Next Generation Access Control. Mathematical Methods, Models, and Architectures for Computer Network Security. Springer. 2003.
[8]Park J., Sandhu R. et al. UCON+: Comprehensive Model, Architecture and Implementation for Usage Control and Continuous Authorization. Data and Applications Security and Privacy XXXVI. Springer. 2023.
[9]Ayoade G. et al. A Comprehensive Review of Usage Control Frameworks. Computer Standards & Interfaces. 2024.
[10] Gerdes S. et al. Using the ACE Framework to Enforce Access and Usage Control with Notifications of Revoked Access Rights. International Journal of Information Security. 2024.
[11] Joumaa H., Petrovska A., Hariri A., Dimitrakos T., Crispo B. Continuous Authorization Architecture for Dynamic Trust Evaluation. Trust Management XIV. Springer. 2024.
[12] Open Policy Agent. OPA-Envoy Plugin. 2026. URL: https://www.openpolicyagent.org/docs/latest/envoy-introduction/
[13] Istio Authors. External Authorization. 2025. URL: https://istio.io/latest/docs/tasks/security/authorization/authz-custom/
[14] OpenFGA Authors. OpenFGA Documentation. 2026. URL: https://openfga.dev/docs
[15] Pang R. et al. Zanzibar: Google's Consistent, Global Authorization System. USENIX Annual Technical Conference. 2019.
[16] Mouelhi T. et al. Specifying and Verifying Usage Control Models and Policies in TLA+. International Journal on Software Tools for Technology Transfer. 2021.
[17] Aloufi B. et al. Authentication and Authorization in Microservices Architecture: A Systematic Literature Review. Applied Sciences. 2022.
[18] Zyanchurin I.A. Reactive Mesh AuthZ: iskhodnyj kod, scenarii zapuska i eksperimental'nye artefakty. Repozitorij GitHub. 2026. URL: https://github.com/Dark-Avery/Reactive-Mesh-AuthZ
[19] Ul'bi, T. V., and O. R. Laponina. "Modul' upravlenie dostupom na osnove atributov dlya veb-zaprosov iz raznyh istochnikov." International Journal of Open Information Technologies 11.5 (2023): 128-136.
[20] Kovtun, D. P., and O. R. Laponina. "Ispol'zovanie upravleniya dostupom na osnove atributov i mTLS v mikroservisnoj arhitekture." International Journal of Open Information Technologies 13.6 (2025): 75-85.
[21] Namiot, D. E. Skhemy atak na modeli mashinnogo obucheniya / D. E. Namiot // International Journal of Open Information Technologies. – 2023. – T. 11, № 5. – S. 68-86. – EDN YVRDOB.
[22] Cifrovaya zheleznaya doroga - prognozy, innovacii, proekty / V. P. Kupriyanovskij, G. V. Sukonnikov, P. M. Bubnov [i dr.] // International Journal of Open Information Technologies. – 2016. – T. 4, № 9. – S. 34-43. – EDN WIQHXX.
[23] [23] Volkov, A. A. O zadachah sozdanija jeffektivnoj infrastruktury sredy obitanija / A. A. Volkov, D. E. Namiot, M. A. Shneps-Shneppe // International Journal of Open Information Technologies. – 2013. – T. 1, # 7. – S. 1-10. – EDN ROMIZX.