# Использование управления доступом на основе атрибутов и mTLS в микросервисной архитектуре

Д.П. Ковтун, О.Р. Лапонина

***Аннотация* – Данная статья посвящена исследованию возможностей использования взаимной TLS-аутентификации (mTLS) совместно с управлением доступом на основе атрибутов (ABAC) в микросервисной архитектуре. В работе рассматриваются основные принципы mTLS, его роль в обеспечении безопасной аутентификации между сервисами, а также способы интеграции TLS-сертификатов в ABAC для принятия решений о доступе.**

**Предложенная модель предусматривает использование TLS-сертификатов для идентификации и авторизации субъектов. Рассматривается архитектурный подход, при котором ABAC реализуется в виде микросервисов, обеспечивающих гибкость, масштабируемость и совместимость с современными распределёнными системами.**

**Интеграция mTLS и ABAC позволит построить безопасную и целостную модель аутентификации и авторизации, обеспечивающую защиту критически важных данных и транзакций в микросервисных системах.**

**Основная идея предлагаемой модели заключается в использовании TLS-сертификатов не только как инструмента аутентификации в процессе mTLS, но и как надёжного источника атрибутов для системы управления доступом на основе атрибутов (ABAC).**

**Традиционно для авторизации использовались токены, такие как JWT, а сертификаты X.509 использовались исключительно для подтверждения подлинности клиента и сервера при установлении TLS-соединения. Однако JWT-токены уязвимы в случае перехвата и требуют дополнительных механизмов валидации. В предлагаемом подходе вся информация, необходимая для авторизации, встраивается в X.509-сертификаты и проверяется криптографически при каждом соединении.**

**Выполнена практическая реализация модели управления доступом на основе атрибутов (ABAC) с использованием механизма mutual TLS (mTLS) в инфраструктуре Kubernetes. В качестве компонентов архитектуры используются Ingress-NGINX в качестве PEP, Open Policy Agent (OPA) в качестве PDP, Rego в качестве языка для описания политик в OPA.**

***Ключевые слова* - микросервисная архитектура, mTLS, TLS-сертификат, ABAC, PEP, PDP, Ingress-NGINX, Open Policy Agent, OPA, Rego.**

## I. Введение

Управление доступом различных групп пользователей/субъектов к ресурсам системы играет ключевую роль в современных автоматизированных системах. Простота и гибкость механизма управления доступом актуальны в связи с растущим числом уязвимостей, связанных с нарушениями доступа. Данные уязвимости входят в перечень наиболее критических рисков безопасности веб-приложений [1].

В современных микросервисных архитектурах аутентификации и авторизации играет решающую роль в обеспечении безопасности данных. Традиционные механизмы, такие как управление доступом на основе ролей (RBAC), часто оказываются недостаточно гибкими в динамических и распределённых системах, где права доступа должны определяться на основе множества факторов, а не только предопределённых ролей пользователей.

Управление доступом на основе атрибутов (ABAC) предоставляет более гибкий и контекстно-зависимый подход к авторизации [2]. В отличие от RBAC, где доступ определяется принадлежностью статическим ролям, ABAC учитывает множество атрибутов — не только кто делает запрос и к каким данным или ресурсам запрашивается доступ, но и условия, при которых этот доступ возможен, и другие возможные атрибуты субъекта и ресурса. Такой подход особенно важен в микросервисных системах, где различные сервисы обмениваются данными, а доступ может зависеть от конкретного контекста запроса, типа данных и уровня доверия между сервисами.

Кроме того, в условиях распределённой архитектуры необходимо учитывать не только идентификацию запрашивающего субъекта, но и различные дополнительные характеристики целевого ресурса.

В данной работе предлагается модель, объединяющая взаимную TLS-аутентификацию (mTLS) и ABAC, при которой цифровые сертификаты используются не только для установления защищённого соединения, но и как источник атрибутов для принятия решений о доступе. Такой подход позволяет не только обеспечивать безопасное взаимодействие между микросервисами, но и учитывать контекст доступа при каждом запросе.

## II. Цели работы

Целью данной работы является исследование возможностей использования взаимной TLS-аутентификации (mTLS) совместно с управлением доступом на основе атрибутов (ABAC) в микросервисной архитектуре, а также разработка модели, в которой TLS-сертификаты используются для хранения атрибутов при принятии решений о доступе.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать принципы работы mTLS и его роль в обеспечении безопасного взаимодействия между микросервисами.
2. Проанализировать подход ABAC и его преимущества перед традиционными методами управления доступом в микросервисных системах.

3. Исследовать возможность использования сертификатов mTLS для хранения атрибутов при принятии решений о доступе.
4. Разработать модель интеграции mTLS и ABAC, обеспечивающую безопасную и гибкую авторизацию.
5. Определить возможные сценарии применения предложенной модели и оценить её преимущества для микросервисных систем.

## III. Угрозы безопасности в микросервисной архитектуре

Микросервисная архитектура предоставляет значительные преимущества в масштабируемости, гибкости и независимости развертывания отдельных компонентов системы. Однако такой подход усложняет обеспечение безопасности, поскольку традиционные методы защиты, разработанные для монолитных приложений, зачастую оказываются неэффективными.

Микросервисные системы подвержены ряду специфических угроз, среди которых можно выделить [3]:

- Сетевые атаки: атаки типа MITM (man-in-the-middle), подмена DNS, перехват JWT-токенов.
- Компрометация сервисов: если злоумышленник получает доступ к одному сервису, он может использовать его для атаки на другие сервисы.
- Ошибки в настройке контейнеров: небезопасные настройки Docker/Kubernetes могут привести к утечке данных или повышению привилегий.
- Проблемы аутентификации и авторизации: отсутствие строгих механизмов контроля доступа между сервисами.

Таким образом, для обеспечения безопасности в микросервисных системах, требуются следующие меры:

- Использование безопасных методов аутентификации и авторизации
- Обеспечение защищенного взаимодействия между сервисами
- Безопасность контейнеров
- Мониторинг системы

## IV. Модели управления доступом в микросервисной архитектуре

Управление доступом – это процесс контроля и ограничения доступа пользователей или систем к ресурсам на основе заданных правил, политик и условий. Он является важнейшим элементом информационной безопасности и позволяет защитить данные, сервисы и инфраструктуру от несанкционированного доступа и помогает решить некоторые из вышеперечисленных проблем, присущих микросервисной архитектуре.

Ключевой частью управления доступом являются механизмы управления доступом (ACM), которые определяют, какие субъекты могут взаимодействовать с определёнными ресурсами и при каких условиях. Они включают в себя:

- Политики управления доступом (Access Control Policies) – набор правил, определяющих, кто и к чему может получить доступ.
- Модели управления доступом (Access Control Models) – формальные структуры, описывающие принципы распределения прав и привилегий.
- Контрольные точки принятия решений (Policy Decision Points, PDP) – компоненты, отвечающие за оценку запросов на доступ на основе политик безопасности.
- Контрольные точки исполнения (Policy Enforcement Points, PEP) – механизмы, применяющие решения о доступе на практике, блокируя или разрешая взаимодействие.

В рамках ACM выделяют несколько ключевых моделей:

- Discretionary Access Control (DAC) – дискреционный контроль, где владельцы ресурсов самостоятельно управляют доступом.
- Mandatory Access Control (MAC) – мандатный контроль, в котором права определяются на основе классификационных меток безопасности.
- Role-Based Access Control (RBAC) – ролевое управление доступом, где права назначаются на основе ролей пользователей.
- Attribute-Based Access Control (ABAC) – атрибутное управление доступом, учитывающее множество факторов, включая свойства субъекта, объекта и контекста запроса.

В настоящий момент в микросервисной архитектуре наиболее популярными являются RBAC и ABAC.

### *A. Role-based access control*

RBAC — это модель управления доступом, при которой пользователи получают определённые права доступа на основе их ролей в системе. Роли представляют собой совокупность разрешений, необходимых для выполнения определённых функций внутри организации.

Разрешения ролей могут наследоваться через иерархию ролей, что упрощает управление доступом. Данная роль может быть назначена как одному человеку, так и группе пользователей.

Ролевое разграничение доступа обеспечивает гибкость в управлении правами пользователей, позволяя адаптировать и изменять их в соответствии с бизнес-требованиями. Этот метод особенно полезен в организациях, где пользователи имеют чётко определённые роли и обязанности.

Роли в RBAC могут быть организованы в иерархические структуры, что позволяет пользователям унаследовать права доступа от родительских ролей.

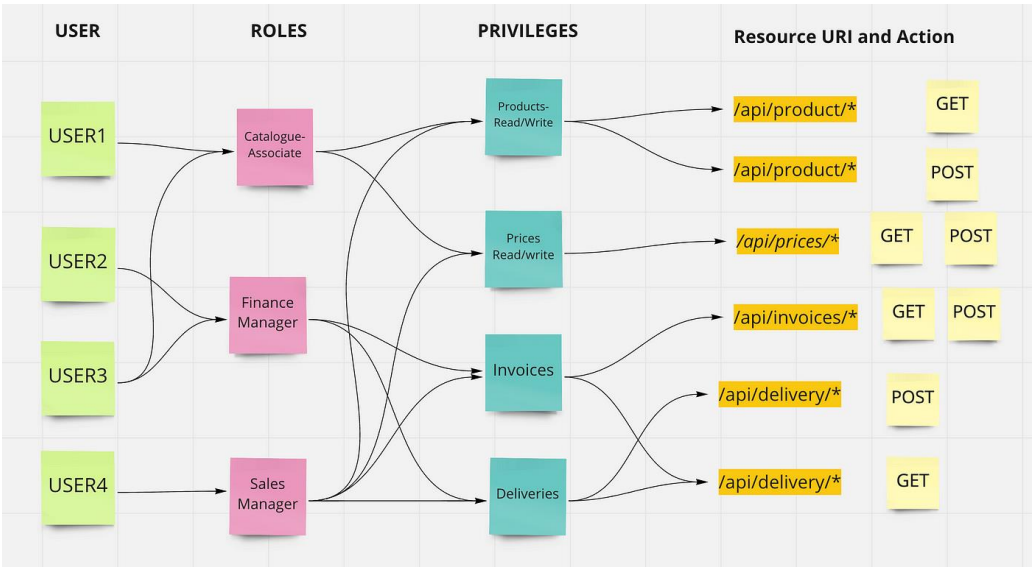


**Рисунок 1. Модель RBAC**

Несмотря на широкое распространение и удобство, модель управления доступом на основе ролей (RBAC) имеет ряд недостатков:

- RBAC плохо подходит для динамически изменяющихся сред;
- Сложность управления при увеличении числа ролей;
- В распределённых системах и микросервисах необходимо учитывать больше параметров, включая изменяющийся контекст запроса.

*B. Attribute-based access control*

Разграничение доступа на основе атрибутов (ABAC) - модель управления доступом, при которой запросы субъекта на выполнение операций над объектами

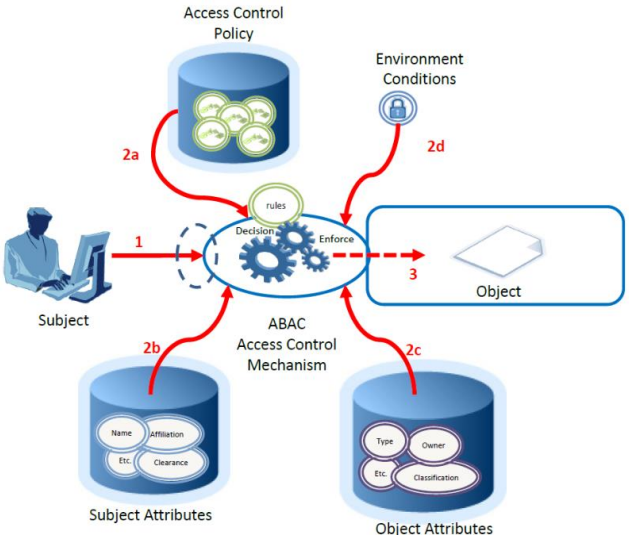


**Рисунок 2. Модель ABAC**

разрешаются или отклоняются на основе следующих признаков [4]:

- Атрибутов субъекта;
- Атрибутов объекта;
- Условий среды;
- Набора политик, указанных с точки зрения этих атрибутов и условий.

Атрибутами являются характеристики пользователей, ресурсов, действий и окружающей среды. Примером атрибута объекта может быть то, какое действие запрашивается над ним (чтение, запись).

Модель ABAC функционирует следующим образом:

1. Субъект запрашивает доступ к объекту;
2. Механизм управления доступом оценивает
    a. Имеющиеся политики;
    b. Атрибуты субъекта;
    c. Атрибуты объекта;
    d. Условия окружающей среды.
3. На основании полученных данных механизм управления доступом принимает решение о предоставления/отклонении доступа.
4. Субъект получает/не получает доступ к запрашиваемому ресурсу.

Используя гибкое описание правил, ABAC предлагает ряд преимуществ:

- Бизнес-правила не привязаны к конкретным ролям, что облегчает их изменение;
- Отсутствие иерархических правил упрощает систему, сохраняя при этом возможность повторного использования правил;
- Нет ограничений на сложность бизнес-правил, что позволяет описывать их более точно, без необходимости создания новых ролей;
- Просто задавать правила с динамически изменяемыми атрибутами, что упрощает определение их без изменений в программном коде.

*C. Описание политик в ABAC*

Важной частью ABAC являются политики доступа. Они формируют правила, на основе которых принимаются решения о разрешении или отклонении запросов. Политики определяются с точки зрения атрибутов субъекта, объекта, действия и условий среды.

Разберем на примере конкретной политики: "*Только сотрудники из отдела кадров могут просматривать*

*персональные данные сотрудников, и только с корпоративных устройств*":

- **Субъект**: должен быть сотрудником отдела кадров;
- **Объект**: файл с персональными данными;
- **Действие**: чтение;
- **Условия среды**: доступ разрешён только с корпоративных устройств и в будни.

Это пример политик на естественном языке, они определяют, как управляется доступ к информации и кто, при каких обстоятельствах, может получить доступ к какой информации.

Чтобы система управления доступом могла автоматически применять правила, их необходимо перевести в формальный язык политик, который может быть интерпретирован программными средствами.

Два популярных языка для описания политик в ABAC:

- XACML (eXtensible Access Control Markup Language) [5] предоставляет возможности для описания политик на основе XML.
- Rego (Open Policy Agent, OPA) – альтернативный, декларативный, более лёгкий и гибкий язык политик, язык, ориентированный на облачные и микросервисные среды.

С помощью этих языков можно описать политики, заданные на естественном языке.

### *D. Компоненты управления доступом в ABAC*

Разграничение доступа на основе атрибутов (ABAC) требует централизованного механизма принятия решений. Для этого система управления доступом включает несколько ключевых компонентов:

- PEP (Policy Enforcement Point) – точка применения политики;
- PDP (Policy Decision Point) – точка принятия решений;
- PAP (Policy Administration Point) – точка администрирования политик;
- PIP (Policy Information Point) – точка получения информации.

PEP – это компонент, который перехватывает запросы на доступ и применяет решение, принятое системой. Он может находиться, например, в API-шлюзе, в системе управления базами данных или в любом другом сервисе, контролирующем доступ к ресурсам.

PDP – это центральный компонент, который анализирует политики доступа и принимает решения о возможности доступа. Он сопоставляет поступивший запрос с заданными политиками, определяя, разрешить или запретить запрошенное действие.

PAP – это модуль управления политиками. Он отвечает за создание, хранение и обновление правил доступа.

PIP – это источник данных, который предоставляет PDP дополнительные атрибуты для принятия решений.

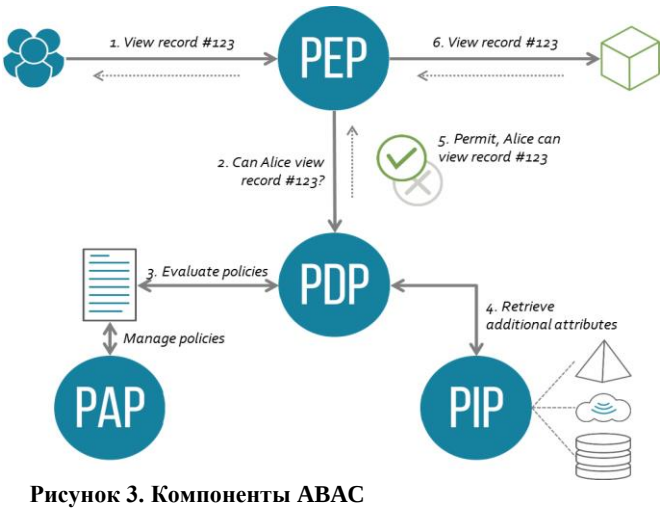


**Рисунок 3. Компоненты ABAC**

Пример полного цикла обработки запроса:

1. Пользователь отправляет запрос → (PEP)
2. PEP передаёт запрос PDP → (PDP)
3. PDP запрашивает политики у PAP → (PAP)
4. PDP запрашивает атрибуты у PIP → (PIP)
5. PDP принимает решение (разрешить/запретить)
6. PEP применяет решение и даёт или блокирует доступ

Эти элементы взаимодействуют между собой, обеспечивая гибкую и динамическую авторизацию пользователей.

## V. MUTUAL TLS

Взаимная TLS-аутентификация (mTLS) — это вариант традиционного TLS (Transport Layer Security), при котором оба участника соединения (клиент и сервер) аутентифицируют друг друга с использованием цифровых сертификатов. В отличие от стандартного TLS, где только сервер предъявляет сертификат, в mTLS клиент также предоставляет свой сертификат, подтверждая свою подлинность.

Этот механизм повышает безопасность сетевого взаимодействия, так как исключает возможность подключения неавторизованных клиентов.

### *A. Основные принципы mTLS*

Основными принципами mutual TLS являются:

- Взаимная аутентификация. Обе стороны (и клиент, и сервер) используют TLS-сертификаты для идентификации и аутентификации. Это предотвращает атаки типа «человек посередине» (MitM), так как оба узла должны быть доверенными (обычный TLS защищает клиента от поддельных серверов, но не защищает сервер от поддельных клиентов).
- Использование сертификатов X.509v3. Они содержат открытый ключ, подпись удостоверяющего центра (CA), срок действия и другую информацию. Эти сертификаты выдаются доверенным центром сертификации.
- Управление доверенными сертификатами. Сервер проверяет сертификат клиента, проверяя его подпись и атрибуты, указанные в

сертификате, используя открытый ключ доверенного CA.

- Шифрование трафика. mTLS гарантирует конфиденциальность передаваемых данных, так как весь трафик шифруется. Это критично для защиты микросервисных взаимодействий, API-запросов и машинного взаимодействия.

### *B. Применение mTLS в микросервисной архитектуре*

В микросервисной архитектуре взаимодействие между сервисами часто происходит по открытым сетям. Использование mTLS даёт несколько преимуществ:

- Безопасность межсервисного взаимодействия – mTLS гарантирует, что только авторизованные сервисы могут взаимодействовать друг с другом.
- Отсутствие необходимости в паролях или токенах – идентификация выполняется с помощью сертификатов, что снижает риск компрометации учетных данных.
- Защита API и удалённых подключений – сервисы могут проверять подлинность клиентов и исключать доступ незарегистрированных устройств.

Однако использование mTLS имеет и некоторые недостатки, которые необходимо учитывать при его внедрении:

- Сложность управления сертификатами – необходимо централизованно выпускать, обновлять и отзывать сертификаты, что требует инфраструктуры PKI (Public Key Infrastructure).
- Дополнительная нагрузка на систему – mTLS требует дополнительной вычислительной мощности для установления соединений и проверки сертификатов.
- Сложность отладки и мониторинга – при возникновении проблем с аутентификацией диагностика может быть затруднена из-за шифрования данных.
- Повышенные требования к DevOps и безопасности – необходимо обеспечивать безопасное хранение закрытых ключей.

### *C. TLS-сертификаты*

TLS-сертификат – это цифровой документ, который подтверждает подлинность сервера (или клиента) и используется для установления защищённого соединения поверх TLS. Он содержит:

- Открытый ключ.
- Информацию о владельце (домен, организация и т. д.).
- Цифровую подпись удостоверяющего центра (CA), подтверждающую подлинность сертификата.
- Срок действия сертификата.
- Дополнительные расширения, которые могут содержать не только политики использования, но и дополнительные атрибуты.

TLS-сертификаты могут использоваться для следующих целей:

- Взаимная аутентификация (mTLS) – клиент и сервер проверяют сертификаты друг друга.
- Управление доступом – информация из сертификатов (например, организация, отдел, роль) может использоваться в моделях управления доступом, таких как ABAC.

TLS 1.3 (RFC 8446) позволяет более эффективно возобновлять установленную сессию, используя 0-RTT при сокращенном Рукопожатии PFS (Perfect Forward Secrecy), что дает возможность использовать сертификаты в качестве основы для построения гибких и безопасных механизмов авторизации на основе атрибутов (ABAC) без снижения общей производительности.

## VI. Совместное использование ABAC и mutual TLS

Микросервисы обмениваются данными в распределённой среде, где необходимо учитывать не только роль пользователя, но и дополнительные характеристики запроса: время, место, тип операции, контекст вызова. ABAC предоставляет такую возможность за счёт оценки набора атрибутов, что делает его особенно ценным в гетерогенных и динамичных системах. Однако, для эффективной работы ABAC требуется достоверный источник этих атрибутов.

Протокол mTLS позволяет установить защищённое соединение и выполнить взаимную аутентификацию. Эти сертификаты могут содержать атрибуты, представляющие идентификационные и организационные данные субъекта, что делает их удобным и надёжным источником информации для механизмов ABAC.

Таким образом, необходимость интеграции ABAC и mTLS возникла естественным путём — как результат стремления объединить:

- безопасный способ установления связи между сервисами (mTLS),
- гибкий, контекстно-зависимый механизм принятия решений о доступе (ABAC).

Подход позволяет не только удостовериться в подлинности участника соединения, но и использовать предоставленную им информацию для строгого контроля доступа.

### *A. Основная идея*

Ключевая идея предлагаемой модели заключается в использовании TLS-сертификатов не только как инструмента аутентификации в процессе mTLS, но и как надёжного источника атрибутов для системы управления доступом на основе атрибутов (ABAC).

Традиционно для авторизации использовались токены, такие как JWT, а сертификаты X.509 использовались исключительно для подтверждения подлинности клиента и сервера при установлении TLS-соединения. Однако JWT-токены уязвимы в случае перехвата и требуют дополнительных механизмов валидации. В предлагаемом подходе вся информация, необходимая для авторизации, встраивается в X.509-сертификаты и проверяется криптографически при каждом соединении. Это позволяет:

- отказаться от хранения и транспортировки токенов,

- повысить защищенность (всё заверено подписью CA),
- интегрировать внешние системы, такие как OpenID, записывая полученные данные в выпускаемый сертификат.

Например, если клиентский сертификат содержит поле подразделение и в нем указан департамент HR (Human Resource), политика может разрешить доступ к сервису обработки персональных данных. Если же в поле организации указано внешнее юридическое лицо, доступ может быть ограничен.

Такая система является наиболее гибкой для организаций, использующих собственные CA – они могут добавлять в сертификаты пользовательские поля, на основании которых ABAC может принимать решение. Она сочетает гибкость токенов и безопасность TLS. Кроме того, эта архитектура хорошо вписывается в философию Zero Trust, где доверие строится не на расположении узла в сети, а на фактах, подтверждённых криптографически и проверенных в момент обращения. Таким образом, использование TLS-сертификатов как источника атрибутов превращает процесс аутентификации в полноценную точку принятия решений по доступу в систему.

### *B. Существующие решения*

Реализация mTLS а также их интеграция с ABAC доступна в ряде популярных стеков и фреймворков. Ниже приведён обзор таких решений и обоснование необходимости предложенного подхода.

Spring Security (Java):

- Поддержка mTLS осуществляется через конфигурацию X509AuthenticationFilter, в котором извлекается имя субъекта (DN) из клиентского сертификата.
- Дополнительно можно настроить AuthenticationManager, чтобы преобразовать DN в роль пользователя.
- Однако, в типичных реализациях используется RBAC — атрибуты из сертификатов напрямую в политике доступа не участвуют.
- Поддержка ABAC возможна, но требует ручной интеграции с внешними PDP, логика взаимодействия с которыми не встроена в Spring Security.

Node.js:

- Нативная поддержка TLS обеспечивается модулем tls (tls.createServer()).
- Для реализации mTLS используется параметр requestCert: true, а проверка клиентского сертификата производится вручную.
- Нет из коробки поддержки ни RBAC, ни ABAC — всё реализуется через пользовательскую логику.
- Сложно обеспечить безопасное хранение ключей и централизованную политику без дополнительных библиотек или прокси.

Istio / Envoy (Service Mesh):

- Поддержка mTLS на уровне всей mesh-сети.
- Политики авторизации задаются через AuthorizationPolicy, но они чаще основаны на субъекте или IP, а не на атрибутах из сертификата.
- Интеграция с OPA и Rego возможна, но требует дополнительных усилий по конфигурации и логике PEP.

**Таблица 1. Сравнение существующих моделей авторизации в различных средах**

| Параметр | Spring | Node.js | Istio /Envoy | Предложенная модель |
|---|---|---|---|---|
| mTLS поддержка | Да | Да | Да | Да |
| Извлечение атрибутов из сертификатов | Частично | Нет | Частично | Да |
| ABAC | Через внешнюю интеграцию | Нет | Через внешнюю интеграцию | Да |
| PDP на Rego /OPA | Настройка вручную | Нет | Частично | Да |
| Безопасная замена JWT | Нет | Нет | Частично | Да |

Таким образом, предлагаемая архитектура сочетает в себе следующие преимущества:

- Централизованный анализ политик на языке Rego
- Использование сертификатов как источника проверяемых атрибутов
- Поддержка гибких условий доступа на уровне среды, субъектов и объектов
- Полная совместимость с Zero Trust

### *C. Архитектура предлагаемого решения*

На рисунке 5 представлена архитектура интеграции моделей ABAC и mTLS для управления доступом в распределённых системах.

Участниками являются те же компоненты, о которых было рассказано в главе 2.

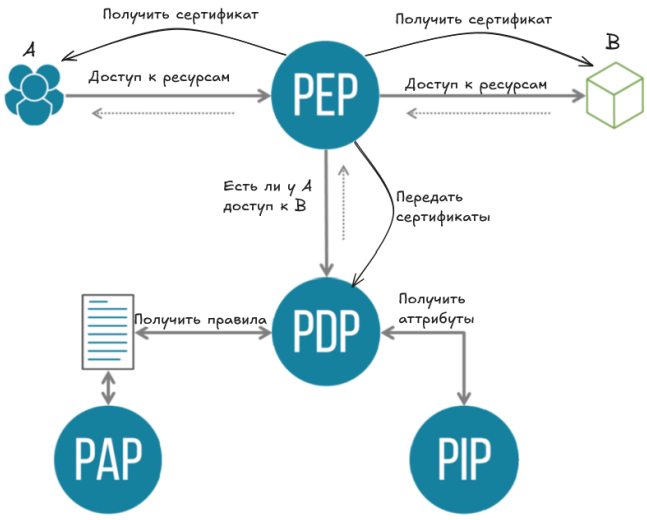


**Рисунок 5. Совместное использование ABAC и mTLS**

Рассмотрим пошаговое выполнение запроса от Алисы (A) к Бобу (B):

1. Выдача сертификатов.
   Участники A и B получают TLS-сертификаты от доверенного центра сертификации. Эти сертификаты могут содержать информацию об организации, подразделении, роли и других параметрах
2. Установление mTLS-соединения.
   При попытке получить доступ к ресурсу, участник A инициирует соединение с B. Используется взаимная TLS-аутентификация, при которой оба участника предъявляют и проверяют сертификаты
3. Анализ запроса в PEP.
   После успешного TLS-рукопожатия, запрос передаётся в компонент PEP, который извлекает сертификаты из соединения и направляет их в PDP
4. Принятие решения PDP.
   PDP получает
   a. Сертификаты и метаинформацию о запросе от PEP
   b. Политики доступа от PAP
   c. Атрибуты через PIP
5. Принятие и возврат решения.
   PDP принимает решение (разрешить/запретить доступ) и передаёт его в PEP, который либо допускает запрос к ресурсу, либо блокирует его.

Интеграция ABAC и mTLS позволяет построить архитектуру авторизации, основанную не только на внешних источниках данных, но и на проверяемых криптографически атрибутах, содержащихся в сертификатах. Ключевыми особенностями и преимуществами являются:

- Достоверность атрибутов – атрибуты содержатся в X.509-сертификатах, которые подписаны доверенным центром сертификации.
- Использование дополнительных сведений о конечных пользователях – атрибуты получаются не только из PIP, но также из данных, полученных из сертификатов.
- Безопасность по умолчанию (Zero Trust Ready) – каждое соединение сопровождается обязательной криптографической проверкой обеих сторон, что соответствует принципам Zero Trust.

### *D. Практическая реализация*

Рассмотрим практическую реализацию модели управления доступом на основе атрибутов (ABAC) с использованием механизма mutual TLS (mTLS) в инфраструктуре Kubernetes. В качестве компонентов архитектуры будут использоваться:

- Ingress-NGINX [6] как Policy Enforcement Point (PEP);
- Open Policy Agent (OPA) [7] как Policy Decision Point (PDP);
- Rego [8] – язык для описания политик в OPA

Эти компоненты выбраны как наиболее востребованные в облачной среде. Требования к работе этих компонент:

- Запрос, поступающий от клиента, сначала попадает на Ingress-nginx, где происходит TLS-рукопожатие. Если используется mTLS, клиент должен предъявить свой сертификат.
- Ingress-nginx должен получить TLS-сертификат сервиса B. Для этого необходимо модифицировать поведение ingress-nginx, чтобы это стало возможным.
- Ingress-nginx выполняет проверку политики доступа, для чего необходимо взаимодействие с OPA, и формирует запрос в OPA, передавая:
  - Сертификаты клиентов,
  - Атрибуты сервиса B,
  - Информацию о запрашиваемом ресурсе, методе и маршруте,
  - Дополнительный контекст (например, IP, время, id клиента и сервера).
- OPA на основе полученной информации и заранее определённых политик (написанных на языке Rego) выносит решение — разрешить или запретить доступ.
- Если allow = true, то Ingress-nginx перенаправляет запрос к целевому сервису. Если нет, то возвращает 403 Forbidden.

С учетом перечисленных требований настройка компонент выглядит следующим образом:

- Для реализации mTLS в Ingress-NGINX используются следующие аннотации

```
nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
nginx.ingress.kubernetes.io/auth-tls-secret: "default/ca-secret"
nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream: "true"
nginx.ingress.kubernetes.io/auth-url: "http://opa-service.opa.svc.cluster.local:8181/v1/data/ingress/authz"
```

- Получение сертификата сервиса (B). Стандартный Ingress-NGINX не предоставляет доступа к TLS-сертификату целевого сервиса. Для решения этой задачи реализуется вспомогательный микросервис cert-fetcher, который по IP-адресу или DNS-имени целевого сервиса выполняет TLS-рукопожатие и возвращает PEM-представление сертификата. Этот сертификат затем передаётся как заголовок X-Server-Cert в запросе к OPA. Также необходимо изменить nginx.tmpl, добавив в него запрос к cert-fetcher.
- В результате взаимодействия OPA получает JSON-запрос следующего вида.

```
{
  "input": {
    "client": "server-a",
    "server": "server-b",
    "client_cert": "<PEM сертификат клиента>",
    "server_cert": "<PEM сертификат сервиса>",
    "method": "GET",
```

```
    "path": "/employee-data",
    "ip": "192.168.1.10"
  }
}
```

- Политики Rego используют встроенную функцию crypto.x509.parse_certificates, которая разбирает сертификаты непосредственно в процессе обработки запроса.

```
package ingress.authz

default allow = false

allow {
    parsed_client := 
crypto.x509.parse_certificates([input.client_ce
rt])
    parsed_server := 
crypto.x509.parse_certificates([input.server_ce
rt])


    client_subject := parsed_client[0].subject
    server_subject := parsed_server[0].subject


    client_subject["OU"] == "HR"
    server_subject["O"] == "Internal Services"
    input.method == "GET"
    input.path == "/employee-data"
    # Также можно использовать атрибуты A и B
через data.clients[input.client]
}
```

Предложенные настройки позволяют реализовать предложенную в статье архитектуру. Таким образом показано, что предложенная интеграция ABAC и mTLS возможна, но требует дополнительной настройки некоторых компонент (на примере ingress-nginx для того, чтобы получать сертификат B).

### *E. Соответствие рекомендациям*

Разрабатываемая модель безопасности с использованием mTLS и ABAC в полной мере соответствует рекомендациям, изложенным в руководстве OWASP Microservices Security Cheat Sheet [9]. Этот документ содержит набор лучших практик по обеспечению безопасности в микросервисной архитектуре, и его положения учитывались при проектировании предлагаемого подхода.

Ключевые рекомендации OWASP и их соответствие нашей модели:

**Таблица 2. Соответствие предложенной модели рекомендациям OWASP**

| **Рекомендации OWASP** | **Реализация в модели mTLS и ABAC** |
|---|---|
| Взаимная аутентификация между сервисами (mTLS) | Полноценная реализация с проверкой сертификатов обеих сторон |
| Отказ от паролей и токенов в межсервисной связи | Все атрибуты встроены в X.509-сертификаты |
| Zero Trust подход к авторизации | Каждый запрос проходит проверку по криптографически проверенным атрибутам |
| Централизованное управление политиками | Использование OPA как PDP, с политиками на языке Rego |
| Ограничение привилегий (Least Privilege) | Атрибуты описывают конкретные права доступа для субъектов |
| Безопасность API Gateway | Ingress-controller выполняет функции PEP с TLS-проверкой |
| Отзыв скомпрометированных сертификатов | Можно использовать CRL/OCSP и short-lived certs |

Такая архитектура безопасности соответствует философии «Defense in Depth», при которой безопасность обеспечивается на нескольких уровнях — от транспортного до уровня бизнес-логики. Кроме того, модель обеспечивает устойчивость к угрозам, перечисленным в OWASP Top 10 [1].

Применение криптографически подписанных сертификатов, строгого контроля атрибутов и централизованных политик снижает вероятность этих уязвимостей.

Предлагаемая архитектура авторизации соответствует положениям, изложенным в документах Национального института стандартов и технологий США (NIST), в частности:

- Security Strategies for Microservices-based Application Systems [10]
- Attribute Based Access Control for Microservices-based Applications using a Service Mesh [11].

Оба документа подчеркивают важность следующих аспектов безопасности в микросервисных средах:

**Таблица 3. Соответствие предложенной модели рекомендациям NIST**

| **Рекомендация NIST** | **Реализация в предлагаемой модели** |
|---|---|
| Использование ABAC с учётом контекста | Поддерживается через PDP/OPA и политики на языке Rego |
| Передача атрибутов через защищённые каналы | Все данные передаются внутри mTLS-сессии |
| Использование взаимной аутентификации | mTLS реализует проверку подлинности и клиента, и сервера |
| Гибкое управление политиками доступа | Реализовано через внешнюю компоненту OPA |
| Минимизация доверия между сервисами (Zero Trust) | Внедрено через криптографически подтверждённые соединения |

В NIST 800-204A акцент сделан на использование атрибутов в сертификатах X.509 как части механизма ABAC, что полностью соответствует предлагаемому решению, где сертификаты служат одновременно для аутентификации и авторизации.

Таким образом, архитектура mTLS + ABAC реализует актуальные принципы и контрольные меры, рекомендованные NIST для защиты микросервисных приложений.

### *F. Ограничения предлагаемого решения*

Несмотря на очевидные преимущества применения mTLS и ABAC в микросервисной архитектуре, следует учитывать ряд ограничений и потенциальных узких мест, связанных с подобным подходом.

Основное ограничение при использовании mTLS — это высокая частота установления TLS-сессий между микросервисами. В отличие от классического TLS, где только клиент проверяет подлинность сервера, в mTLS

обе стороны участвуют во взаимной аутентификации, что требует дополнительных криптографических операций и обмена сертификатами. В условиях высоконагруженной микросервисной среды, где взаимодействия между сервисами могут происходить десятки или сотни раз в секунду, это приводит к значительным затратам на установку и поддержание безопасных соединений.

TLS 1.3 поддерживает *session resumption* на основе PSK. Для снижения нагрузки следует предусмотреть reuse сессий и сессионное кеширование.

Также накладываются дополнительные сложности в обслуживании ABAC. Хотя он предоставляет более гибкую модель управления доступом по сравнению с RBAC, он требует сложной инфраструктуры, что усложняется использованием данных из сертификатов.

Для обеспечения актуальности и надёжности системы необходимо предусмотреть механизм быстрой отмены сертификатов. Возможны следующие подходы:

- Использование CRL (Certificate Revocation List) и OCSP (Online Certificate Status Protocol).
- Генерация короткоживущих сертификатов с автоматической ротацией.
- Интеграция с cert-manager в Kubernetes для динамической замены ключей.

Эти механизмы позволяют гибко реагировать на изменения в атрибутах субъектов, например, при увольнении сотрудника или изменении его полномочий, но также накладывают дополнительные трудности при настройке этих механизмов.

## VII. Заключение

В данной статье рассмотрены принципы использования взаимной TLS-аутентификации (mTLS) и управления доступом на основе атрибутов (ABAC) в микросервисной архитектуре. Было показано, что традиционные механизмы управления доступом, такие как RBAC, часто оказываются недостаточно гибкими в условиях динамических распределённых систем. В свою очередь, ABAC позволяет учитывать широкий набор атрибутов (субъекта, объекта, условий среды), что делает управление доступом более контекстно-зависимым и безопасным.

Одним из ключевых элементов безопасности в микросервисной архитектуре является TLS, обеспечивающий защищённое соединение между сервисами. Однако, mTLS расширяет стандартную TLS-модель, добавляя взаимную аутентификацию, что делает возможным надёжное удостоверение не только серверов, но и клиентов. Это позволяет исключить ненадёжные механизмы аутентификации, такие как пароли и токены, и существенно повышает уровень защиты межсервисных взаимодействий.

Кроме того, TLS-сертификаты могут использоваться не только для установления защищённого соединения, но и для авторизации на основе атрибутов. Информация, содержащаяся в сертификатах (например, организация, отдел, уровень доступа), может служить дополнительным фактором при принятии решений о доступе в рамках ABAC, обеспечивая более строгий и гибкий контроль.

Таким образом, сочетание mTLS и ABAC является перспективным решением для построения защищённых микросервисных сред, обеспечивая надёжную аутентификацию, гибкое управление доступом и высокий уровень защиты данных, но имеющие несколько ограничений, которые необходимо решать.

## Благодарности

Тема статьи, написанной по результатам магистерской диссертации, соответствует направлению “Кибербезопасность” на факультете ВМК МГУ [12]. Как примеры похожих работ см. публикации [13,14].

Как обычно, отмечаем работы В.П. Куприяновского и его многочисленных соавторов, положивших начало цифровой повестке в журнале [15, 16].

## Библиография

[1] The Open Web Application Security Project (OWASP) Top 10, https://owasp.org/Top10/#welcome-to-the-owasp-top-10-2021, 2021.

[2] Fatima A. et al. Towards Attribute-Centric Access Control: an ABAC versus RBAC argument //Security and Communication Networks. – 2016. – Т. 9. – №. 16. – С. 3152-3166.

[3] Yarygina T., Bagge A. H. Overcoming security challenges in microservice architectures //2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). – IEEE, 2018. – С. 11-20.

[4] Hu V. C. et al. Guide to attribute based access control (abac) definition and considerations (draft) //NIST special publication. – 2013. – Т. 800. – №. 162. – С. 1-54.

[5] Anderson A. et al. extensible access control markup language (xacml) version 1.0 //Oasis. – 2003.

[6] Ingress-NGINX [Электронный ресурс]. – Режим доступа: https://kubernetes.github.io/ingress-nginx/, свободный. – Дата обращения: 03.04.2025.

[7] Open Policy Agent [Электронный ресурс]. – Режим доступа: https://www.openpolicyagent.org/, свободный. – Дата обращения: 03.04.2025.

[8] Rego: Policy Language for OPA [Электронный ресурс]. – Режим доступа: https://www.openpolicyagent.org/docs/latest/policy-language/, свободный. – Дата обращения: 03.04.2025.

[9] OWASP. Microservices Security Cheat Sheet [Электронный ресурс]. - Режим доступа: https://cheatsheetseries.owasp.org/cheatsheets/Microservices_Security_Cheat_Sheet.html свободный - Дата обращения: 16.04.2025.

[10] Chandramouli R. Microservices-based application systems //NIST Special Publication. – 2019. – Т. 800. – №. 204. – С. 800-204.

[11] Chandramouli R. et al. Attribute-based access control for microservices-based applications using a service mesh //NIST Special Publication. – 2021. – Т. 800. – С. 41.

[12] Сухомлин, Владимир Александрович. "Создание профиля" Кибербезопасность и искусственный интеллект" для направления подготовки ФИИТ на основе куррикулумного подхода." Современные информационные технологии и ИТ-образование 17.3 (2021): 724-734.

[13] Зимина, К. И., and О. Р. Лапонина. "Механизмы межсервисной аутентификации в приложениях с микросервисной архитектурой." International Journal of Open Information Technologies 11.5 (2023): 146-154.

[14] Ulbi, Timur V., and Olga R. Laponina. "Attribute based access control module for cross-origin web requests." International Journal of Open Information Technologies 11.5 (2023): 128-136.

[15] Развитие транспортно-логистических отраслей Европейского Союза: открытый BIM, Интернет Вещей и кибер-физические системы / В. П. Куприяновский, В. В. Аленьков, А. В. Степаненко [и др.] // International Journal of Open Information Technologies. – 2018. – Т. 6, № 2. – С. 54-100. – EDN YNIRFG.

[16] Цифровая экономика = модели данных + большие данные + архитектура + приложения? / В. П. Куприяновский, Н. А. Уткин, Д. Е. Намиот, П. В. Куприяновский // International Journal of Open Information Technologies. – 2016. – Т. 4, № 5. – С. 1-13. – EDN VWANDZ.

Статья получена 14 мая 2025.
Д.П. Ковтун- МГУ имени М.В. Ломоносова (email: danila.kovtunn@gmail.com)
О.Р. Лапонина - МГУ имени М.В. Ломоносова (email: laponina@oit.cmc.msu.ru)

# Using attribute-based access control and mTLS in microservice architecture

D.P. Kovtun, O.R. Laponina

***Abstract* - This paper explores the possibilities of using mutual TLS authentication (mTLS) together with attribute-based access control (ABAC) in a microservice architecture. The paper discusses the basic principles of mTLS, its role in providing secure authentication between services, and ways to integrate TLS certificates into ABAC for making access decisions.**

**The proposed model involves using TLS certificates to identify and authorize subjects. An architectural approach is considered in which ABAC is implemented as microservices that provide flexibility, scalability, and compatibility with modern distributed systems**

**Integration of mTLS and ABAC will allow building a secure and holistic authentication and authorization model that protects critical data and transactions in microservice systems. The main idea of the proposed model is to use TLS certificates not only as an authentication tool in the mTLS process, but also as a reliable source of attributes for the attribute-based access control (ABAC) system.**

**Traditionally, tokens such as JWT have been used for authorization, and X.509 certificates are used exclusively to confirm the authenticity of the client and server when establishing a TLS connection. However, JWT tokens are vulnerable to interception and require additional validation mechanisms. In the proposed approach, all information required for authorization is embedded in X.509 certificates and verified cryptographically for each connection.**

**A practical implementation of the attribute-based access control (ABAC) model using the mutual TLS (mTLS) mechanism in the Kubernetes infrastructure is performed. Ingress-NGINX is used as a PEP, Open Policy Agent (OPA) as a PDP, Rego as a language for describing policies in OPA are used as architectural components.**

***Keywords* - microservice architecture, mTLS, TLS certificate, ABAC, PEP, PDP, Ingress-NGINX, Open Policy Agent, OPA, Rego.**

## REFERENCES

[1]The Open Web Application Security Project (OWASP) Top 10, https://owasp.org/Top10/#welcome-to-the-owasp-top-10-2021, 2021.

[2]Fatima A. et al. Towards Attribute-Centric Access Control: an ABAC versus RBAC argument //Security and Communication Networks. – 2016. – T. 9. – #. 16. – S. 3152-3166.

[3]Yarygina T., Bagge A. H. Overcoming security challenges in microservice architectures //2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). – IEEE, 2018. – S. 11-20.

[4]Hu V. C. et al. Guide to attribute based access control (abac) definition and considerations (draft) //NIST special publication. – 2013. – T. 800. – #. 162. – S. 1-54.

[5]Anderson A. et al. extensible access control markup language (xacml) version 1.0 //Oasis. – 2003.

[6]Ingress-NGINX [Jelektronnyj resurs]. – Rezhim dostupa: https://kubernetes.github.io/ingress-nginx/, svobodnyj. – Data obrashhenija: 03.04.2025.

[7]Open Policy Agent [Jelektronnyj resurs]. – Rezhim dostupa: https://www.openpolicyagent.org/, svobodnyj. – Data obrashhenija: 03.04.2025.

[8]Rego: Policy Language for OPA [Jelektronnyj resurs]. – Rezhim dostupa: https://www.openpolicyagent.org/docs/latest/policy-language/, svobodnyj. – Data obrashhenija: 03.04.2025.

[9]OWASP. Microservices Security Cheat Sheet [Jelektronnyj resurs]. Rezhim dostupa: https://cheatsheetseries.owasp.org/cheatsheets/Microservices_Security_Cheat_Sheet.html svobodnyj - Data obrashhenija: 16.04.2025.

[10] Chandramouli R. Microservices-based application systems //NIST Special Publication. – 2019. – T. 800. – #. 204. – S. 800-204.

[11] Chandramouli R. et al. Attribute-based access control for microservices-based applications using a service mesh //NIST Special Publication. – 2021. – T. 800. – S. 41.

[12] Suhomlin, Vladimir Aleksandrovich. "Sozdanie profilja" Kiberbezopasnost' i iskusstvennyj intellekt" dlja napravlenija podgotovki FIIT na osnove kurrikulumnogo podhoda." Sovremennye informacionnye tehnologii i IT-obrazovanie 17.3 (2021): 724-734.

[13] Zimina, K. I., and O. R. Laponina. "Mehanizmy mezhservisnoj autentifikacii v prilozhenijah s mikroservisnoj arhitekturoj." International Journal of Open Information Technologies 11.5 (2023): 146-154.

[14] Ulbi, Timur V., and Olga R. Laponina. "Attribute based access control module for cross-origin web requests." International Journal of Open Information Technologies 11.5 (2023): 128-136.

[15] Razvitie transportno-logisticheskih otraslej Evropejskogo Sojuza: otkrytyj BIM, Internet Veshhej i kiber-fizicheskie sistemy / V. P. Kuprijanovskij, V. V. Alen'kov, A. V. Stepanenko [i dr.] // International Journal of Open Information Technologies. – 2018. – T. 6, # 2. – S. 54-100. – EDN YNIRFG.

[16] Cifrovaja jekonomika = modeli dannyh + bol'shie dannye + arhitektura + prilozhenija? / V. P. Kuprijanovskij, N. A. Utkin, D. E. Namiot, P. V. Kuprijanovskij // International Journal of Open Information Technologies. – 2016. – T. 4, # 5. – S. 1-13. – EDN VWANDZ.