gRPC: основы, безопасность и авторизация

Николаев Михаил, БИБ232

Что такое gRPC?

- RPC*-фреймворк с открытым исходным кодом, разработанный Google
- Использует протокол HTTP/2
- Позволяет определять сервисы и методы с помощью Protocol Buffers (protobuf)
- Основные преимущества:
  - Высокая производительность
  - Эффективная сериализация данных
  - Поддерджка большого количества языков

*RPC – remote procedure call

Обзор безопасности в gRPC

- Безопасность в gRPC базируется на:
  - Шифровании соединения (TLS*)
  - Механизмах аутентификации (JWT, OAuth2 и т. д.)
  - mTLS для взаимной аутентификации

TLS – Transport Layer Security; mTLS – mutual TLS

Преимущества HTTP/2

HTTP/2

- Использует бинарные фреймы для передачи данных
- Заголовки сжимаются, что существенно уменьшает объём передаваемых данных
- Позволяет серверу отправлять данные клиенту без явного запроса
- Можно одновременно передавать несколько запросов и ответов в рамках одного TCP-соединения

HTTP/1.1

- Основан на текстовом формате, что влечёт дополнительные накладные расходы на парсинг
- Заголовки передаются в несжатом текстовом виде
- Сервер может отвечать только на запрос клиента и не имеет механизма «проталкивания» ресурсов

Проблемы, связанные с HTTP/2

- Downgrade-атаки и попытки переключиться на HTTP/1.1
  - При подключении по HTTPS клиент и сервер согласовывают версию протокола, используя механизм ALPN*
  - Злоумышленник может попытаться перехватить трафик и заставить стороны перейти на HTTP/1.1, где защита слабее
  - Итог: теряются преимущества HTTP/2 (мультиплексирование, более надёжное шифрование), что может привести к уязвимостям и снижению производительности
- Возможные уязвимости на уровне TLS/HTTP/2, влияющие на конфиденциальность
- DoS-атаки: чрезмерное использование ресурсов сервера из-за большого числа потоков или запросов

ALPN- Application-Layer Protocol Negotiation

Механизмы защиты от MitM-атак

- TLS: Шифрование трафика для предотвращения пассивных атак и подслушивания, валидация сертификата сервера для защиты от ложных сервисов
- mTLS: Дополнительно проверяет сертификат клиента на сервере, исключает неавторизованных клиентов
- Дополнительные меры проверки сертификатов:
  - Регулярная проверка на отзыв сертификатов для своевременного обнаружения компрометации
  - Убедиться, что сертификат сервера (и клиента при mTLS) выпущен и подписан известным корневым CA* или промежуточным CA

CA – Certificate Authority

OAuth2 в gRPC

- Использование JWT при аутентификации
- Передача Bearer-токена в метаданных gRPC
- Валидность токена и проверка на сервере
- Преимущества:
- Централизованная система аутентификации
- Хорошо интегрируется со многими сервисами (Google, Auth0, Keycloak и т. д.)

mTLS как метод авторизации

- Взаимная аутентификация:
  - Клиент проверяет сервер
  - Сервер проверяет клиента
- Применяется в высокобезопасных окружениях (internal API, микросервисы)
- Управление сертификатами (PKI-инфраструктура):
  - Сложность настройки
  - Проблемы ротации сертификатов
  - Высокая степень доверия

Сравнение OAuth2 и mTLS

OAuth2

- Простая реализация для клиентов
- Подходит для публичных API
- Гибкие механизмы контроля прав

mTLS

- Высокий уровень доверия и безопасности
- Сложная инфраструктура
- Подходит для внутренних микросервисов и корпоративных систем

Лучшие практики настройки безопасности gRPC

- Использовать последний протокол TLS (1.2 или 1.3)
- Обновлять сертификаты и ключи по расписанию
- Грамотно обрабатывать ошибки аутентификации/авторизации
- Постоянно тестировать безопасность

Выводы и рекомендации

- gRPC — высокопроизводительный RPC-фреймворк, критически важно соблюдать меры безопасности
- HTTP/2 даёт новые возможности, но требует аккуратной настройки и защиты от downgrade-атак
- Защита от MITM — обязательное шифрование TLS; при высокой степени доверия — mTLS
- Авторизация: OAuth2 (JWT) и/или mTLS, в зависимости от сценария
- Регулярные проверки и обновления (сертификаты, токены) — залог безопасной системы