**Федеральное государственное автономное образовательное учреждение**

**высшего образования   
«Национальный исследовательский университет   
«Высшая школа экономики»**

**Московский институт электроники и математики им. А.Н.Тихонова**

Направление подготовки

**«10.03.01 Информационная безопасность»**

Образовательная программа **«Информационная безопасность»**

**О Т Ч Е Т**

**По семинару «Разработка защищенных приложений»**

**Студент:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Николаев М.А. |  |  |
|  | Фамилия И.О. |  | Подпись |

**Преподаватель:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ведущий программист МИЭМ НИУ ВШЭ |  | Башун В.В. |  |  |
| должность и место работы |  | Фамилия И.О. |  | подпись |

**Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва, 2025**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 19](#_Toc156911908)

[1. Теоретические основы планирования маркетинга на предприятии 20](#_Toc156911909)

[1.1. Особенности организации маркетинга на предприятии 20](#_Toc156911910)

[1.2. Основные концепции управления маркетинговой деятельностью на предприятии 20](#_Toc156911911)

[1.2.1. Концепция совершенствования производства 20](#_Toc156911912)

[1.2.2. Концепция совершенствования товара 20](#_Toc156911913)

[1.2.3. Концепция маркетинга и концепция социально-этичного маркетинга 20](#_Toc156911914)

[1.3. Особенности формирования маркетингового плана на предприятии (на примере ООО) 20](#_Toc156911915)

[2. Исследование маркетинговой деятельности и особенностей составления маркетингового плана на примере ООО 21](#_Toc156911916)

[2.1. Организационно-экономическая характеристика маркетинговой деятельности на предприятии 21](#_Toc156911917)

[2.1.1. Анализ рыночных возможностей 21](#_Toc156911918)

[2.1.2. Отбор целевых рынков 21](#_Toc156911919)

[2.1.3. Разработка комплекса маркетинга 21](#_Toc156911920)

[2.1.3.1. Анализ маркетинговой деятельности предприятия 21](#_Toc156911921)

[2.2. Процесс формирование основных разделов маркетингового плана на примере ООО 21](#_Toc156911922)

[3. Рекомендации по совершенствованию маркетингового планирования на предприятии 22](#_Toc156911923)

[3.1. Определение основных направлений маркетинговой политики 22](#_Toc156911924)

[3.2. Формирование плана рекламной кампании на предприятии 22](#_Toc156911925)

[ЗАКЛЮЧЕНИЕ 23](#_Toc156911926)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc156911927)

[**Список компьютеров и программного обеспечения, доступных** 25](#_Toc156911928)

[**в сети на момент аудита** 25](#_Toc156911929)

# ВВЕДЕНИЕ

В условиях стремительного развития распределённых систем и микросервисной архитектуры возрастает потребность в быстрых, надёжных и масштабируемых способах взаимодействия между сервисами. Одним из таких решений стал **gRPC** — современный фреймворк удалённого вызова процедур, разработанный Google.

gRPC построен на основе протокола HTTP/2 и использует компактный бинарный формат сериализации Protocol Buffers, что делает его значительно более производительным по сравнению с традиционными REST API. Однако высокая производительность и гибкость требуют соответствующего внимания к вопросам **безопасности**: передаваемые данные должны быть защищены от перехвата, подмены и несанкционированного доступа.

В рамках этого доклада мы рассмотрим:  
– что такое gRPC и как он работает;  
– особенности протокола HTTP/2 и связанные с ним риски (в том числе downgrade-атаки);  
– как обеспечивается **шифрование и защита от MITM-атак** с помощью TLS и mTLS;  
– какие механизмы авторизации, включая **OAuth2**, применяются в gRPC для контроля доступа.

Цель доклада — дать общее представление о технологии gRPC и показать, как грамотно настроить её использование в безопасных распределённых системах.

# 1. История gRPC и его создатели

С появлением микросервисной архитектуры в Google в начале 2010-х годов перед инженерами встала непростая задача: как обеспечить надёжный, масштабируемый и при этом лёгкий в использовании механизм обмена сообщениями между сотнями разрозненных компонентов. Внутри компании уже успешно применялись Protocol Buffers для эффективного хранения и передачи структурированных данных, но для межсервисного взаимодействия требовалось нечто большее, чем просто сериализация. Требовалось решение, способное обрабатывать длительные потоки запросов, поддерживать двунаправленные потоки данных и при этом оставаться независимым от платформы и языка программирования.

Идея gRPC зародилась как логичное продолжение этой внутренней эволюции: сочетание преимуществ Protocol Buffers с возможностями современного HTTP/2. HTTP/2 в тот момент лишь начинал входить в обиход, предлагая мультиплексирование, встроенное сжатие заголовков и приоритеты потоков. Благодаря этому инженеры Google получили инструмент, позволяющий не только быстро передавать бинарные сообщения, но и эффективно управлять соединениями, минимизируя задержки и расход TCP-сессий. Разработка велась в тени закрытых репозиториев, где команды экспериментировали с видом API, моделями потоковой передачи и механизмом авторизации.

Момент истины наступил в мае 2015 года, когда Google принял решение поделиться своим детищем с сообществом и опубликовал на GitHub проект grpc/grpc под лицензией Apache 2.0. Этот шаг стал одновременно признанием зрелости технологии и приглашением к открытому сотрудничеству: исходники оказались доступными всем желающим, а подробная документация и набор примеров дали разработчикам прочный фундамент для старта. Уже к концу года в репозитории появились стабильные реализации на C++ и Go — тех языках, которые были центром разработки микросервисов внутри Google. Следом подтянулись Java и Python, а чуть позже к ним присоединились C#, Ruby и другие популярные экосистемы.

В центре проекта оказалась небольшая, но сплочённая команда инженеров Google Cloud: Вадим Яворский, Эрик Сиггерс и Сэм Янг взяли на себя роль архитекторов и координаторов, формируя дорожную карту и следя за тем, чтобы gRPC оставался не просто «обёрткой» над HTTP/2, а полноценной платформой с богатым набором вспомогательных библиотек. Их усилия создавали мосты между миром Google и всем остальным сообществом: они активно участвовали в конференциях, писали статьи и выступали с докладами, объясняя, почему гетерогенные микросервисы выигрывают от жёстко типизированного контракта, определённого в .proto-файлах.

К 2020 году экосистема вокруг gRPC значительно разрослась: проект получил статус «Graduated» в Cloud Native Computing Foundation, а на практике его использовали десятки крупных компаний — от финансовых стартапов до телеком-гигантов. Сообщество открытых разработчиков ежедневно вносило вклад в кодовую базу, добавляя новые плагины, расширения для сетевой безопасности и интеграции с популярными фреймворками. Именно эта открытость, подкреплённая инженерной культурой Google, позволила gRPC вырасти из внутреннего прототипа в один из стандартов де-факто для коммуникации между распределёнными системами.

# 2. Как работает gRPC

2.1 Описание сервисов и сообщений

• Определение интерфейса в .proto-файлах (Protocol Buffers v3).

• Пример:

syntax = "proto3";

service Greeter {

rpc SayHello (HelloRequest) returns (HelloReply);

}

message HelloRequest { string name = 1; }

message HelloReply { string message = 1; }

2.2 Генерация клиентских и серверных «заглушек»

• Утилита protoc создаёт код-обёртки (stubs) для выбранного языка.

• Разработчик реализует лишь бизнес-логику, остальное берет на себя gRPC-рантайм.

2.3 Модели вызова

• Unary (1→1).

• Server streaming (1→N).

• Client streaming (N→1).

• Bidirectional streaming (N↔N).

2.4 Жизненный цикл запроса

1) Клиент вызывает локальный метод.

2) Сериализация данных в protobuf → бинарный буфер.

3) Отправка через HTTP/2-канал.

4) Приём на сервере, десериализация, исполнение, обратный путь.

# 3. Особенности протокола HTTP/2 и связанные риски

3.1 Ключевые преимущества HTTP/2

• Бинарный формат фреймов вместо текста.

• Мультиплексирование потоков в одном TCP-соединении.

• Сжатие заголовков HPACK.

• Server push (редко используется в gRPC).

3.2 Downgrade-атаки

• При установке TLS через ALPN клиент и сервер договариваются о «h2».

• Злоумышленник или некорректный прокси может сбросить ALPN-переговоры, вернув нас к HTTP/1.1.

• Контрмеры: HSTS и строгае́ SSL-политики.

3.3 DoS-риски

• Создание тысяч параллельных потоков (MaxConcurrentStreams).

• Переполнение буферов некорректными фреймами.

• Решения: лимиты на потоки, rate-limiting, circuit-breaker.

3.4 Проблемы совместимости

• Старые прокси и балансировщики могут не поддерживать HTTP/2.

• Неожиданный fallback на HTTP/1.1 нарушает производительность и безопасность.

# 4. Шифрование и защита от MITM-атак: TLS и mTLS

4.1 Принципы TLS

• Handshake: ClientHello → ServerHello (+ALPN=h2) → обмен ключами (ECDHE) → проверка сертификата.

• Сертификаты X.509, подписанные доверенным CA.

• TLS 1.3 сокращает количество раундов и исключает слабые шифры.

4.2 Принцип mTLS

• Взаимная проверка сертификатов: клиент предъявляет свой, сервер — свой.

• Высокий уровень доверия, но требует PKI-инфраструктуры (CA, ротация ключей).

# 5. Механизмы авторизации

5.1 OAuth2 и JWT

• Клиент получает access-токен у Identity Provider.

• Токен передаётся в метаданных gRPC

• Сервер проверяет подпись (JWKs), срок действия, поле scope.

5.2 Другие методы

• API-ключи в Metadata.

• Custom-interceptors: проверка ролей/прав.

• SPIFFE/SPIRE для доверенных сервис-to-сервис сценариев.

# Заключение

gRPC стал одним из ключевых инструментов для высокопроизводительного обмена данными в микросервисных архитектурах.

Его преимущества — двусторонний стриминг, дешёвые операции сериализации и эффективное использование сетевых ресурсов — реализуются на основе HTTP/2 и Protocol Buffers.

Однако для защиты каналов необходимо применять TLS (желательно версии 1.3) и, при высоких требованиях к безопасности, mTLS.

Для контроля доступа внешних клиентов отлично подходит OAuth2/JWT, а сервис-to-сервис коммуникацию лучше обезопасить через mTLS или SPIFFE/SPIRE.

Только комплексный подход к шифрованию, аутентификации и авторизации позволяет построить надёжную и масштабируемую распределённую систему.

Список литературы

1. Official gRPC Documentation — https://grpc.io

2. RFC 7540 — Hypertext Transfer Protocol Version 2 (HTTP/2)

3. RFC 8446 — The Transport Layer Security (TLS) Protocol Version 1.3

4. Google Cloud Whitepaper: “Building Secure gRPC Microservices”

5. Ivan Ristić. Bulletproof TLS and PKI. 2023

6. OWASP Cheat Sheet — Transport Layer Protection

7. CNCF SPIFFE/SPIRE Specifications

8. Istio Security Insights Blog Series, 2024