Теория 10 (5)

Digest-авторизация

Метод аутентификации, используемый в протоколе HTTP, который предоставляет механизм идентификации сообщений и защиту от повторного использования перехваченных данных авторизации.

В отличие от базовой аутентификации, Digest-авторизация не передает пароль в открытом виде по сети. Вместо этого он использует хэш-функцию для создания “отпечатка” пароля, который затем передается на сервер. Это делает Digest-авторизацию более безопасной, чем базовую аутентификацию, особенно при использовании незащищенных сетей.

CRAM-MD5

CRAM-MD5 (Challenge-Response Authentication Mechanism MD5) - это протокол аутентификации, который не передает пароли в открытом виде. Вместо этого он использует хэш-функцию MD5 для передачи хэша пароля и уникальной строки, известной как “nonce”.

Протокол работает следующим образом:

1. Сервер отправляет уникальную строку (nonce) клиенту.
2. Клиент генерирует хэш MD5 от пароля и nonce.
3. Клиент отправляет этот хэш обратно на сервер.
4. Сервер сравнивает полученный хэш с тем, который он ожидал получить. Если они совпадают, аутентификация проходит успешно.

Этот метод предоставляет некоторую защиту от перехвата паролей, так как злоумышленник, перехвативший хэш, не сможет использовать его для повторной аутентификации без знания исходного пароля.

MD5 - криптографическая хэш-функция, которая преобразует данные любого размера в уникальный отпечаток фиксированного размера (128 бит или 32 шестнадцатеричных символа).
Отпечаток всегда одинаков для одних и тех же входных данных.

Алгоритм:

1. **Ввод данных**: MD5 принимает входные данные любого размера.
2. **Подготовка данных**: Входные данные делятся на блоки по 512 бит. Если данные не делятся нацело, они дополняются до следующего 512-битного блока.
3. **Инициализация буфера**: MD5 использует 4-словный буфер (A, B, C, D), который инициализируется конкретными значениями.
4. **Обработка**: Каждый 512-битный блок обрабатывается в 4 “раунда”, которые состоят из 16 операций. Операции включают в себя битовые операции и модульное сложение.

5. **Вывод**: После обработки всех блоков буфер преобразуется в 128-битный хеш, который и является выходными данными.

OAuth 2.0

OAuth 2.0 - это открытый стандарт для авторизации, который позволяет третьим сторонам получать ограниченный доступ к HTTP-сервису. Это обычно используется для предоставления пользователям возможности входа в систему на веб-сайтах с использованием учетных данных от других сервисов, таких как Google или Facebook.

Вот как работает OAuth 2.0:

1. Пользователь перенаправляется на страницу входа в систему провайдера.
2. После успешного входа в систему провайдер перенаправляет пользователя обратно на веб-сайт с кодом авторизации.
3. Веб-сайт обменивает код авторизации на токен доступа.
4. Токен доступа затем используется для доступа к защищенным ресурсам от имени пользователя.

Существует также параметр `scope`, который указывает уровень доступа, который предоставляется приложению или пользователю.

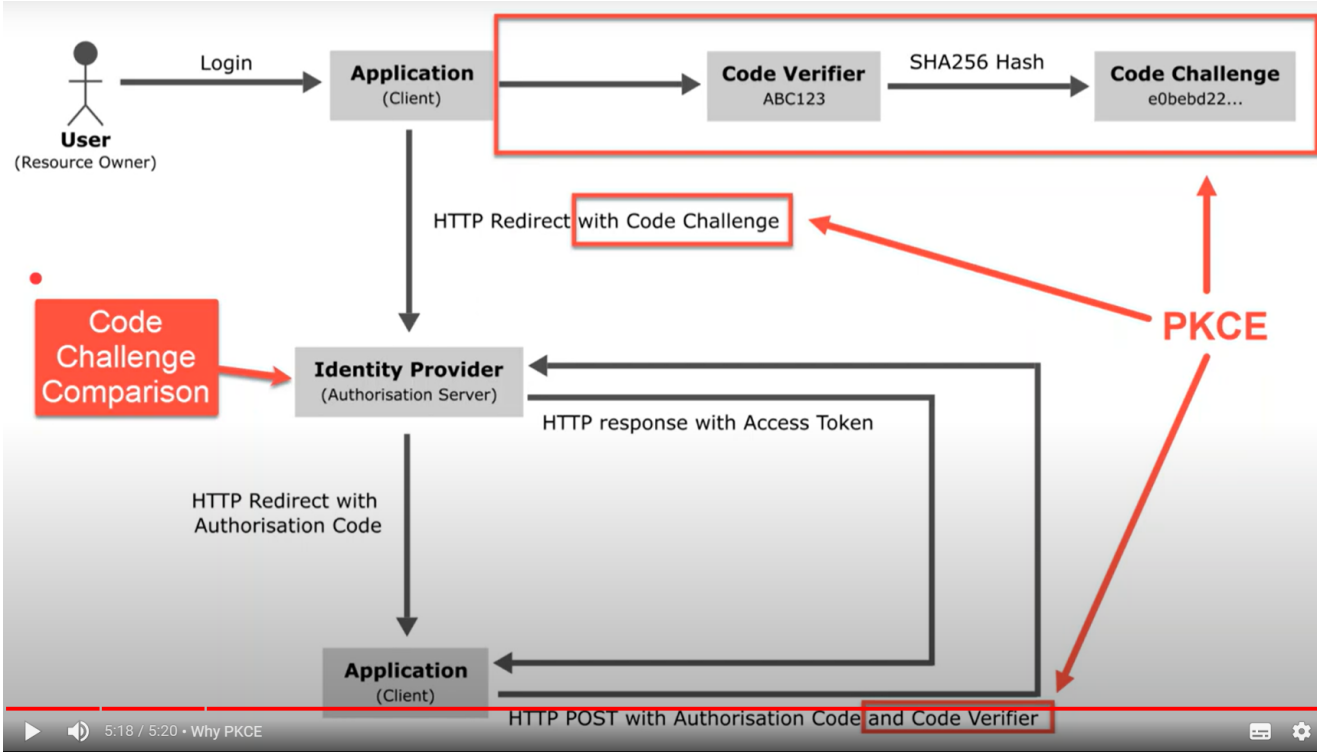
Authorization Code Flow + PKCE

Authorization Code Flow + PKCE (Proof Key for Code Exchange) - это расширение стандартного потока авторизации OAuth, предназначенное для безопасной аутентификации одностраничных приложений или нативных приложений.

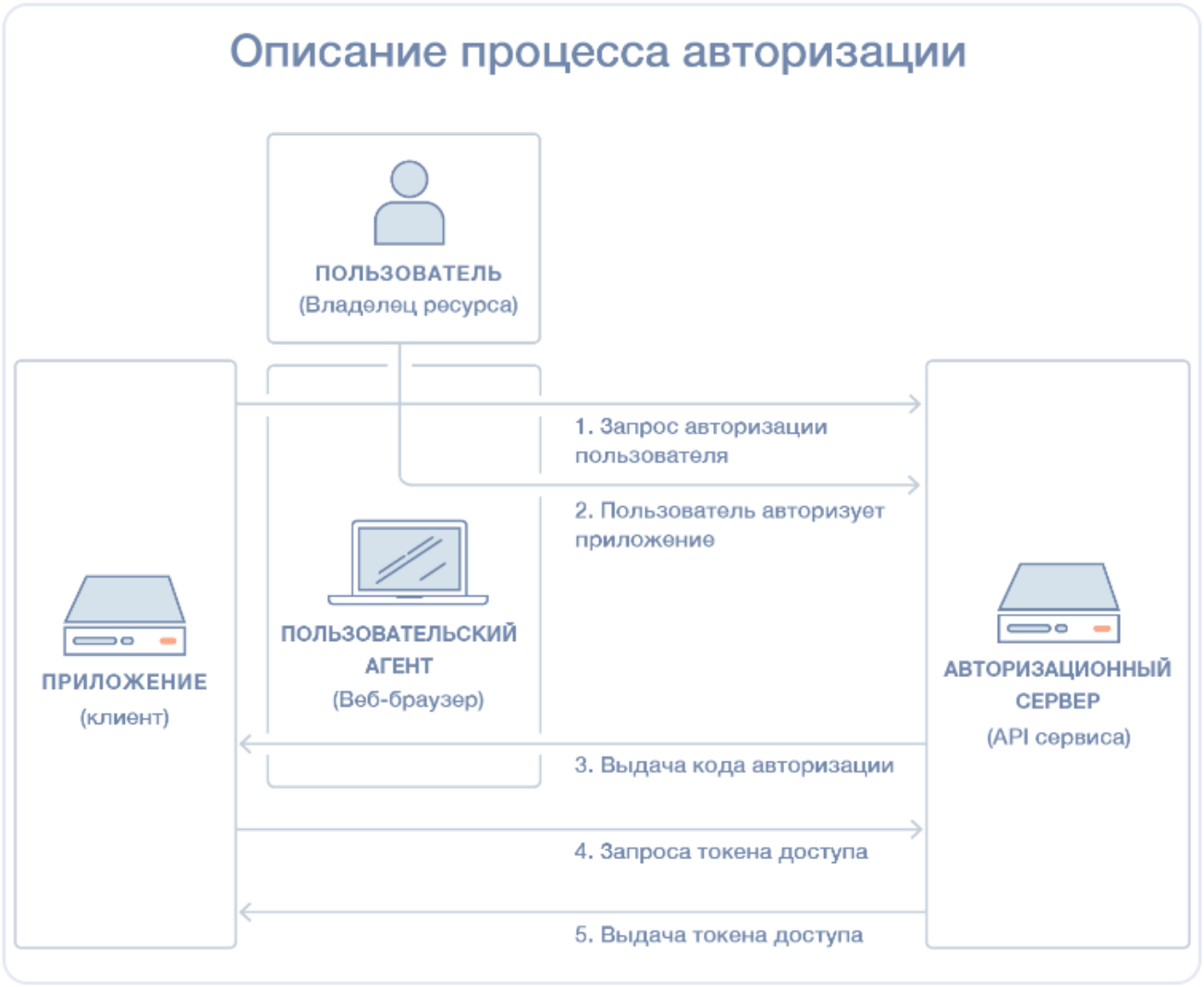
Вот как работает этот поток:

1. **Создание кода испытания**: Ваше приложение создает случайную строку, которую мы называем “код испытания”. Этот код испытания преобразуется в хэш.
2. **Авторизация пользователя**: Приложение перенаправляет пользователя на страницу входа в систему провайдера авторизации (например, Google или Facebook) с этим хэшем кода испытания.
3. **Получение кода авторизации**: После того, как пользователь успешно вошел в систему, провайдер авторизации перенаправляет пользователя обратно в ваше приложение с так называемым “кодом авторизации”.
4. **Обмен кода авторизации на токен доступа**: Ваше приложение отправляет код авторизации и оригинальный код испытания обратно провайдеру авторизации. Если хэш кода испытания совпадает с хэшем, который был отправлен во втором шаге, провайдер авторизации возвращает вашему приложению “токен доступа”.
5. **Использование токена доступа**: Теперь ваше приложение может использовать этот токен доступа для доступа к защищенным ресурсам от имени пользователя.

PKCE предотвращает атаку, в которой злоумышленник может перехватить код авторизации и обменять его на токен доступа. Если злоумышленник попытается обменять перехваченный код авторизации, он не сможет предоставить оригинальный код испытания, и обмен будет отклонен.



https://youtu.be/3OMENOsq2VA



Client Credentials Flow

Client Credentials Flow - это поток OAuth 2.0, который позволяет приложениям обмениваться своими учетными данными (например, идентификатором клиента и секретом клиента) на токен доступа. Этот поток обычно используется для сервер-серверной аутентификации, где приложение нуждается в доступе к сервису, но не от имени конкретного пользователя.

Вот как работает этот поток:

1. **Аутентификация приложения**: Приложение отправляет запрос на сервер авторизации с его учетными данными (идентификатором клиента и секретом(уникальный ключ) клиента).
2. **Получение токена доступа**: Если учетные данные приложения верны, сервер авторизации возвращает токен доступа.
3. **Использование токена доступа**: Приложение может использовать этот токен доступа для доступа к защищенным ресурсам на сервере ресурсов.

Этот поток не включает в себя шаги, связанные с вводом или получением учетных данных пользователя, поэтому он не подходит для сценариев, где требуется аутентификация или авторизация пользователя. Он обычно используется для автоматических задач или задач на стороне сервера, где требуется доступ к сервису, но не от имени пользователя.

Implict Flow

Implicit Flow - это упрощенный поток OAuth 2.0, который был ранее рекомендован для нативных приложений и JavaScript-приложений, где токен доступа возвращается немедленно без дополнительного шага обмена кодом авторизации.

Вот как работает Implicit Flow:

1. **Авторизация пользователя**: Приложение перенаправляет пользователя на страницу входа в систему провайдера авторизации (например, Google или Facebook).
2. **Получение токена доступа**: После успешного входа в систему, провайдер авторизации перенаправляет пользователя обратно в приложение с токеном доступа в URL.
3. **Использование токена доступа**: Приложение извлекает токен доступа из URL и использует его для доступа к защищенным ресурсам от имени пользователя.

Device Code Flow

Device Code Flow - это поток аутентификации, который позволяет пользователям использовать другое устройство, такое как компьютер или мобильный телефон, для интерактивного входа в систему. Этот поток особенно полезен для устройств или операционных систем, которые не предоставляют веб-браузер.

Вот как работает Device Code Flow:

Запрос кода устройства: Приложение отправляет запрос на сервер авторизации для получения кода устройства и кода пользователя.
Авторизация пользователя: Приложение просит пользователя перейти на указанный веб-сайт на другом устройстве и ввести код пользователя.
Получение токена доступа: Приложение регулярно отправляет запросы на сервер авторизации, чтобы проверить, ввел ли пользователь код. Как только код введен, сервер авторизации возвращает токен доступа.
Этот поток обеспечивает безопасность, поскольку токен доступа никогда не передается напрямую, а код, который передается, не может быть использован без соответствующего кода устройства

JWT (JSON Web Token)

JSON Web Token (JWT) - это открытый стандарт (RFC 7519), который определяет компактный и самодостаточный способ безопасной передачи информации между сторонами в виде объекта JSON. Эта информация может быть проверена и доверена, поскольку она цифрово подписана.

JWT обычно состоит из трех частей, разделенных точками, которые являются:

- Заголовок
- Полезная нагрузка
- Подпись
  Таким образом, JWT обычно выглядит следующим образом: xxxxx.yyyyy.zzzzz.

Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения подлинности аккаунта.

Фингер принт

Фингерпринт или отпечаток компьютера (браузера) - это информация, собранная об удалённом устройстве для дальнейшей идентификации. Это уникальный идентификатор конфигураций веб-браузера и операционной системы, который формируется на основе собранных данных различными технологиями отслеживания.

Фингерпринт может быть использован для различных целей:

1. Предотвращение мошенничества и кражи личности: Фингерпринт может помочь определить и предотвратить мошенничество, такое как мошенничество с кредитными картами.
2. Персонализация рекламы: Анализируя действия пользователя и устройства в Интернете, веб-сайты могут использовать полученную информацию для создания и показа персонализированной рекламы.
3. Внутренняя аналитика: Действия пользователя (просмотренные страницы, клики, движения мышкой и т. п.) позволяют разработчикам получать негласную обратную связь, видеть, на что пользователь обращает внимание в первую очередь и корректировать свои онлайн-платформы для достижения максимального отклика

Refresh Token и Access Token

- **Refresh Token** - это специальный токен, который используется для получения дополнительных токенов доступа без необходимости повторной аутентификации пользователя.
- **Access Token** - это токен, который используется для доступа к защищенным ресурсам.

Если эти токены перехвачены, злоумышленник может получить доступ к защищенным ресурсам или даже выдать себя за пользователя. Поэтому очень важно использовать безопасные методы передачи этих токенов, такие как HTTPS.