47. Авторизация

Цель:

Познакомиться со следующими особенностями авторизации:

- виды авторизаций
- OAuth 2
- JWT tokens

План занятия:

- Что такое авторизация и зачем она нужна
    - виды
- OAuth 2
- JWT tokens

Конспект:

Авторизация: предоставление пользователю прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

Аутентификация: процедура проверки легальности пользователя или данных, например, проверки соответствия введенного пользователем пароля к учётной записи паролю в базе данных.

Авторизация производит контроль доступа к различным ресурсам системы в процессе работы легальных пользователей после успешного прохождения ими аутентификации.

Виды аутентификаций/авторизаций:

1. Сессии
2. Токены
3. Беспарольный вход
4. Единая точка входа (Single Sign On, SSO)
5. Аутентификация в соцсетях

1. Сессии:

1. Пользователь вводит в браузере своё имя и пароль, после чего клиентское приложение отправляет запрос на сервер.
2. Сервер проверяет пользователя, аутентифицирует его, затем в ответе клиентскому приложению шлет уникальный пользовательский токен. При этом этот токен сохраняется в память или базу данных.
3. Клиентское приложение сохраняет токены в куках и отправляет их при каждом последующем запросе.
4. Сервер получает каждый запрос, требующий аутентификации, с помощью токена проверяет пользователя и возвращает запрошенные данные.

5. Когда пользователь выходит, клиентское приложение удаляет его токен, поэтому все последующие запросы от этого клиента становятся неавторизованными.

Недостатки сессий:

- При каждой аутентификации пользователя сервер должен создавать у себя запись. При большом количестве пользователей, может быть высокая нагрузка на сервер.
- Работа с сессиями может вызывать проблемы при масштабировании, так как все сессии хранятся в памяти. Если нам необходимо многократно реплицировать сервер, то на все новые серверы придётся реплицировать и все пользовательские сессии. Вариантом решения данной проблемы может быть создание выделенного сервера для управления сессиями, но и это не может считаться простым решением проблемы.

2. Токены:

1. Пользователь вводит имя и пароль, после чего клиентское приложение отправляет запрос на сервер.
2. Сервер проверяет их и возвращает токены (JWT), который может содержать метаданные вроде user_id, роль, разрешения и т. д.
3. Токен хранится на клиентской стороне, чаще всего в локальном хранилище, но может лежать и в хранилище сессий или кук.
4. Последующие запросы к серверу обычно содержат этот токен в качестве дополнительного заголовка авторизации в виде *Bearer {JWT}*.
5. Сервер расшифровывает JWT, если токен верный, сервер обрабатывает запрос.
6. Когда пользователь выходит из системы, токен на клиентской стороне удаляется.

Преимущества:

- Нет необходимости хранить записи токенов на сервере.
- JWT может хранить в себе метаданные любого типа

3. Беспарольная аутентификация

Беспарольная аутентификация — это способ входа и аутентификации пользователей без ввода паролей.

Вместо ввода почты/имени и пароля пользователи вводят только свою почту. Приложение отправляет на этот адрес одноразовую ссылку, пользователь по ней кликает и автоматически входит в клиентское приложение авторизованным пользователем.

Недостаток:

- если кто-то получит доступ к почте пользователя, он получит доступ к сайтам пользователя

Преимущество:

- нет проблем с частой проблемой у пользователя, что он забыл пароль и его надо восстанавливать
- разработчикам не нужно разрабатывать функционал “сбросить/создать/изменить пароль”

4. Единая точка входа (Single Sign On, SSO)

Логика единой точки входа: пользователь входит один раз и получает доступ ко всем системам без необходимости входить в каждую из них.

1. Пользователь входит в один из сервисов Google.

2. Пользователь получает сгенерированную в Google Accounts куку .
3. Пользователь идёт в другой продукт Google.
4. Пользователь снова перенаправляется в Google Accounts.
5. Google Accounts видит, что пользователю уже присвоена кука, и перенаправляет пользователя в запрошенный продукт.

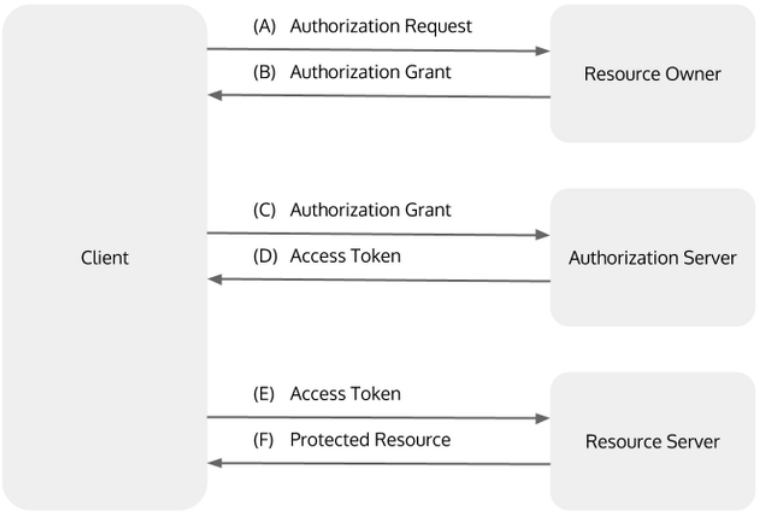
5. Аутентификация в соцсетях (Social sign-in)

Аутентификацией в соцсетях — это разновидность единой точки входа с упрощением процесса регистрации/входа пользователя в разных приложениях.

Есть возможность аутентифицировать пользователей по их аккаунтам в соцсетях. Тогда пользователям не придётся проходить аутентификацию отдельно в другом приложении.

OAuth 2

Общая схема OAuth 2.0 :



1. Клиент запрашивает авторизацию у владельца ресурса на доступ к серверу ресурсов.
2. Клиент получает грант авторизации.
3. Клиент запрашивает токен доступа путем аутентификации с помощью сервера авторизации и предоставление гранта авторизации.
4. Сервер авторизации аутентифицирует клиента, проверяя грант авторизации и, если он действителен, выдает токен доступа (access token) и рефреш токен (refresh token).
5. Клиент запрашивает защищенный ресурс у провайдера и аутентифицируется, представляя токен доступа.
6. Провайдер проверяет токен доступа и, если он действителен, предоставляет запрашиваемый ресурс приложению.

Преимущества и недостатки OAuth 2.0

Преимущества:

- Обращение к ресурсам происходит по HTTP/HTTPS с указанием токена в заголовках. Это позволяет использовать OAuth практически на любых платформах: мобильных и десктоп приложениях, сайтах, и даже в плагинах для браузеров.
- Возможность авторизации пользователя.
- Популярность - большинство компаний используют его в своих API.
- Простота реализации
- Большое количество документации, статей и ресурсов
- Наличие готовых решений, которые можно изменять под свои нужды

Из минусов:

- Нет единого установленного формата, вследствие чего на каждый сервис нужно иметь отдельную реализацию.
- При аутентификации иногда приходится делать дополнительные запросы для получения даже минимальной информации о пользователе. Решается использованием jwt токена, но далеко не все сервисы его поддерживают.
- При краже токена у злоумышленника на какое-то время появляется доступ к защищенным данным.

Токены JWT

Токены доступа (JWT) — это токены, с помощью которых можно получить доступ к защищенным ресурсам. Это короткоживущие, но многоразовые токены. В них может содержаться дополнительная информация, например, время жизни токена. Все зависит от желания разработчика.

Рефреш токен (RT) — эти токены служат только для получение нового токена доступа. Они долгоживущие, но одноразовые.

Схема использования у токенов следующая:

- Пользователь логинится в приложении, передавая логин и пароль на сервер. Сервер возвращает два токена и время их жизни. При этом пароль и логин не сохраняются на клиентском приложении(на устройстве)
- Приложение сохраняет токены и использует access token для последующих запросов
- Когда время жизни access token подходит к концу или уже истекло, приложение использует refresh token, чтобы обновить оба токена и продолжить использовать новый access token

Полезные ссылки:

1. https://habr.com/ru/post/340146/
2. https://habr.com/ru/post/466929/
3. https://habr.com/ru/company/Voximplant/blog/323160/