**Авторизация**

**Авториза́ция (англ. authorization «разрешение; уполномочивание») —** предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

Авторизацию не следует путать с аутентификацией.

**Аутентификация –** это процедура проверки подлинности, например:

- проверка подлинности пользователя путём сравнения введённого им пароля (для указанного логина) с паролем, сохранённым в базе данных пользовательских логинов;

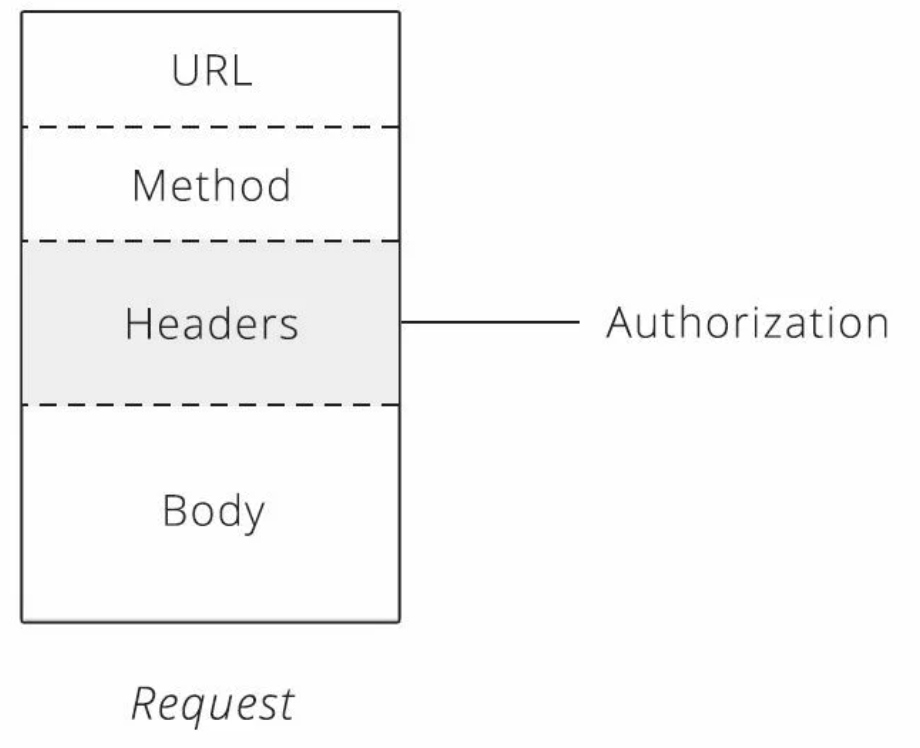
- подтверждение подлинности электронного письма путём проверки цифровой подписи письма по открытому ключу отправителя.

Существуют различные протоколы, предназначенные для защиты пользователей, которые избавляют их от необходимости повторного ввода паролей или учетных данных. Рассмотрим наиболее популярные из этих протоколов.

**Base Autentification**

Вход в систему с использованием имени пользователя и пароля — один из примеров аутентификации. Когда вы аутентифицируетесь на сервере, вы подтверждаете свою личность серверу, сообщая ему информацию, которую знаете только вы (логин и пароль). Эти данные отправляются в заголовке запроса Authorization в виде base64 кода:

Authorization: Basic base64\_encode(login:password)



Когда сервер получает запрос, он просматривает заголовок Авторизации и сравнивает его с сохраненными учетными данными. Если имя пользователя и пароль совпадают с одним из пользователей в списке сервера, сервер выполняет запрос клиента от лица этот пользователя. Если совпадений нет, сервер возвращает специальный код состояния (401), чтобы клиент знал, что аутентификация не удалась и запрос отклонен.

Главным достоинством такой авторизации является простота реализации.

Уязвимостей гораздо больше:

- Передача открытого пароля (пароль и имя пользователя передаются нешифрованными, секретная информация оснащена весьма броской «меткой» – текстом «Authorization», которую легко найти в общем потоке данных);

- Возможность подбора пароля (BA не предоставляет никаких средств, ограничивающих количество неудачных попыток авторизоваться);

- Невозможность «разлогиниться» (разработчик веб-ресурса никак не может заставить браузер забыть пароль и имя пользователя).

**Open Authorization, OAuth**

OAuth – это протокол авторизации открытого стандарта, который можно добавить в приложения, чтобы позволить пользователям получать безопасный доступ к их платформе.

OAuth — протокол для авторизованного доступа к стороннему API. OAuth позволяет скрипту потребителя получить ограниченный API-доступ к данным стороннего поставщика, если пользователь дает добро. Т.е. это средство для доступа к API.

Пример: представим, что мы пишем социальную сеть. В ней имеется форма импорта контактов из адресной книги электронного почтового ящика. Что нужно для доступа к контактам почты? Конечно, логин и пароль от ящика. Но если мы попросим ввести их на нашем сайте, пользователь заподозрит неладное. Поэтому нам хочется, чтобы пароль вводился только на сайте электронной почты, и после этого доступ к контактам через API электронной почты предоставлялся нашей социальной сети.

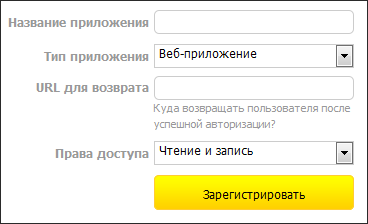
Форма состоит из единственной кнопки — «Импортировать контакты». После нажатия на нее пользователя временно редиректят на страницу электронной почты, где он вводит свой логин и пароль. Далее пользователя возвращают обратно на наш сайт, где скрипт уже получает возможность скачать контакты через внутренний API электронного ящика.

Задача OAuth — сделать так, чтобы пользователь имел возможность работать на сервисе соцсети с защищенными данными электронной почты, вводя пароль к этим данным исключительно на странице электронной почты и оставаясь при этом на сайте соцсети.

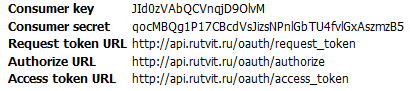
Когда клиент использует OAuth, сервер выдает маркер доступа третьей стороне, этот маркер используется для доступа к защищенному ресурсу, а источник проверяет этот маркер. Здесь еще стоит обратить внимание на то, что личность владельца маркера не проверяется.

Общий принцип работы

Откуда поставщики ресурсов (электронный почтовый ящик) узнают о приложениях (соцсеть), которые хотят получить данные пользователя? Поставщики предоставляют специальную форму регистрации приложений.



После регистрации приложения вам выдается 5 параметров, которые требуются для работы с OAuth.

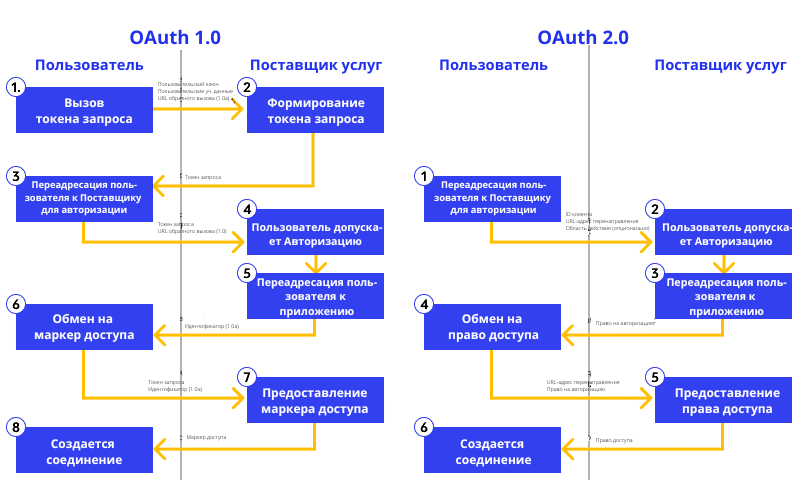


Один из принципов OAuth гласит, что никакие секретные ключи не должны передаваться в запросах открытыми. Поэтому протокол оперирует понятием Token. Токен очень похож на пару логин + пароль: логин — открытая информация, а пароль — известен только передающей и принимающей стороне. В терминах OAuth аналог логина называется Key (Consumer key), а аналог пароля — Secret (Consumer secret). Ситуация, когда Secret известен только отправителю и получателю, но более никому, называется Shared Secret.

Далее, в общем виде, взаимодействие строится таким образом:

1. Приложение получает Request Token.
2. Пользователь перенаправляется на сайт Поставщика и авторизует там Request Token.
3. Приложение обменивает Request Token на Access Token.
4. Приложение делает авторизованные запросы к API сервиса.

На данный момент на смену OAuth пришла OAuth 2.0. OAuth 1.0 был разработан с учетом криптографических требований, а также он не поддерживал более трех потоков и даже масштабирование. OAuth 2.0 разработан так, что он поддерживает целых шесть потоков, которые в свою очередь поддерживают различные приложения и требования. Этот протокол аутентификации позволяет использовать подписанные учетные идентификационные данные через HTTPS. Токены OAuth не нужно шифровать при отправке из одного места в другое, поскольку они шифруются непосредственно при передаче.



**OAuth 2.0**

OAuth 2.0 — протокол авторизации, позволяющий выдать одному сервису (приложению) права на доступ к ресурсам пользователя на другом сервисе. Протокол избавляет от необходимости доверять приложению логин и пароль, а также позволяет выдавать ограниченный набор прав, а не все сразу.

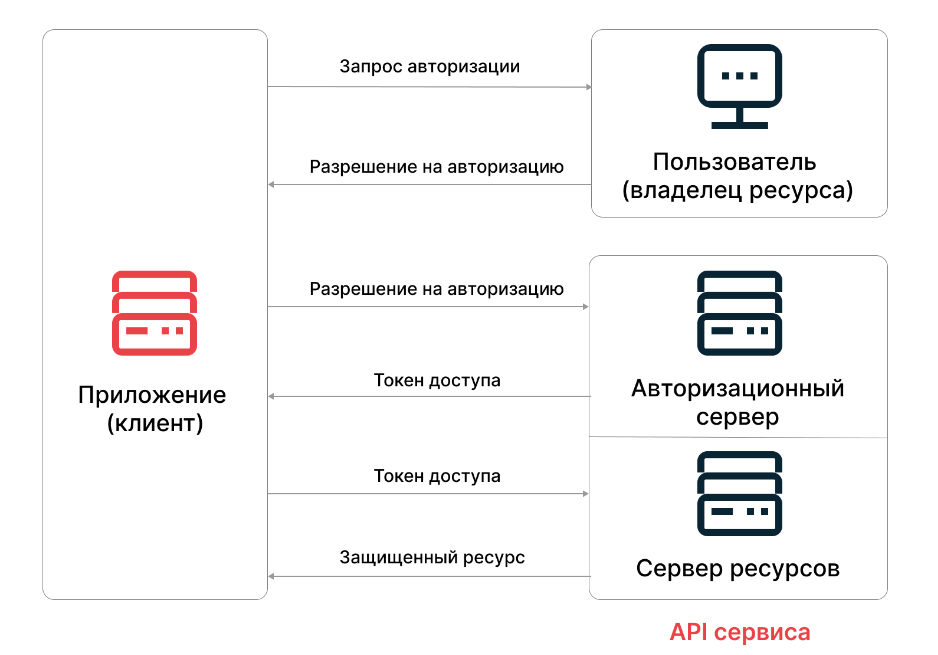
Типы ролей здесь остаются прежними:

- владелец (пользователь): авторизует клиентское приложение на доступ к данным своего аккаунта;

- сервер ресурсов/API: здесь располагаются данные пользовательских аккаунтов, а также бизнес-логика авторизации, отвечающая за выдачу новых OAuth-токенов и проверку их подлинности при обращении клиентских приложений к ресурсам;

- клиентское приложение: сервис, которому пользователь делегирует права доступа к своим данным на сервере ресурсов. Пользователь должен авторизовать приложение, а со стороны сервера API оно должно получить подтверждение в виде ключа (токена) доступа.

Принцип действия протокола



1. Приложение запрашивает у пользователя разрешение на доступ к серверу ресурсов.
2. После получения разрешения приложение сообщает об этом авторизационному серверу, а также предоставляет ему сведения о себе.
3. Сервер авторизации проверяет подлинность разрешения и предоставленных сведений о приложении. В случае успешной проверки генерируется токен доступа.
4. Далее приложение обращается к серверу ресурсов, предоставляя токен в качестве подтверждения пройденной авторизации.
5. Сервер ресурсов проверяет действительность токена и выдает приложению доступ к запрашиваемому ресурсу.

Преимущества и недостатки OAuth 2.0

Плюсы

* использование SSL для защиты данных,
* ограничение прав доступа стороннего приложения к пользовательским ресурсам областями видимости,
* обилие библиотек для реализации использования протокола во всех языках общего назначения.

Минусы

* различия в подходах к реализации у разных сервисов, порождающие необходимость написания отдельного кода под каждый новый сервис;
* если реализована интеграция с семейством приложений, работающих в одной экосистеме (например, Google), существует риск для всех интеграций при компрометации учетных данных либо сбое на стороне сервера API;
* OAuth 2.0 является новым и динамично развивающимся протоколом, в связи с чем возможны некоторые логические противоречия в его спецификации;
* основанная на SSL безопасность протокола может стать проблемой в высоконагруженных проектах.

Различия протоколов OpenID и OAuth 2

Основное различие между этими двумя протоколами состоит в цели использования. Так, OpenID служит для аутентификации, то есть подтверждения личности пользователя в клиентском сервисе. Например, вы можете авторизоваться в любом из сервисов Google под своим аккаунтом и работать с ними от своего имени, со своими данными. OAuth же представляет собой протокол авторизации, то есть выдачи клиентскому сервису прав на выполнение действий с ресурсами пользователя (как правило, на чтение данных, но иногда и на изменение) от его имени.

Для верификации пользователя OpenID использует ID учетной записи у провайдера, а OAuth — авторизационные ключи (токены) с предопределенным сроком действия и правами доступа.

**OpenID**

OpenID — открытый стандарт децентрализованной системы аутентификации, предоставляющей пользователю возможность создать единую учётную запись для аутентификации на множестве не связанных друг с другом интернет-ресурсов, используя услуги третьих лиц.

OpenID Connect является надстройкой над OAuth 2.0. C помощью OAuth 2.0 мы только авторизовали пользователя т.е. о пользователе мы знаем только access token, с помощью которого можем получать определенные данные из почтового ящика. Но access token ничего не говорит о личности пользователя и с помощью него мы не можем предоставить доступ к закрытым данным пользователя на нашем сайте соцсети. OpenID Connect — добавляет сведения о логине и профиле пользователя (identity). Это как раз и позволяет реализовать его аутентификацию.

OpenID Connect также добавляет возможность динамической регистрации и обнаружения сервисов “service discovery”. Это дает возможность строить системы SSO (Single Sign-On), в которых используется один логин для многих не связанных между собой сервисов.

OpenID Connect (OIDC) расширяет OAuth 2.0 следующими основными возможностями:

Authorization Server, помимо access token и refresh token, возвращает “identity token” (ID Token). Он содержится в том же JWT. Из ID Token можно извлечь следующую информацию: имя пользователя, время входа в учетную запись, срок окончания действия ID Token. ID Token можно передавать между участниками.

OIDC предоставляет дополнительное API, которые позволяет запрашивать информацию о пользователе и его текущих сессиях.