HTTP Basic Authentication

Первым, что при обращении к защищенному ресурсу сервер выдаст пользователю, не имеющему доступа, будет ошибка 401 Unauthorized. При этом ответ также содержит информацию о типе аутентификации (в нашем случае – Basic), который он может принимать, и контекст, в рамках которого эта аутентификация действует (Realm). Пользователь вводит логин и пароль, они упаковываются в Base64 и отправляются на сервер для проверки. Здесь существуют различные опасности. Самая распространенная — угроза man-in-the-middle attack, или атаки посредника, в ходе которой при использовании незащищенного соединения учетные данные могут перехватить злоумышленники в момент передачи от клиента к серверу или обратно.

### HTTP Digest Authentication

Следующим этапом развития технологии стала чуть более сложная система HTTP digest authentication, которая исключает передачу учетных данных в открытом виде — здесь для проверки используется MD5-хеш с некоторыми примесями, что позволяет избежать подбора логина и пароля. Конечно, этот алгоритм выглядит более надежным, но и он подвержен целому ряду не самых сложных атак.

### Forms Authentication

Позднее появился процесс Forms authentication, при котором аутентификация происходит на более высоком уровне модели абстракции. HTTP-сервер при этом не сообщает об ошибке доступа, а просто перенаправляет неаутентифицированного пользователя на другую страницу. Обычно на этой странице отображаются поля для ввода логина и пароля, после заполнения которых формируется POST-запрос с данными и через защищенный канал направляется на сервер. Серверная сторона в свою очередь возвращает пользователю токен или идентификатор сессии, который сохраняется в Cookies и в дальнейшем используется для доступа к защищенному ресурсу.

### Token Authentication

Следующее поколение способов аутентификации представляет Token Based Authentication, который обычно применяется при построении систем Single sign-on (SSO). При его использовании запрашиваемый сервис делегирует функцию проверки достоверности сведений о пользователе другому сервису. Т. е. провайдер услуг доверяет выдачу необходимых для доступа токенов собственно токен-провайдеру (Identity provider). Это то, что мы видим, например, входя в приложения через аккаунты в социальных сетях. Вне IT самой простой аналогией этого процесса можно назвать использование общегражданского паспорта. Официальный документ как раз является выданным вам токеном — все государственные службы по умолчанию доверяет отделу полиции, который его вручил, и считает паспорт достаточным для вашей аутентификации на протяжении всего срока действии при сохранении его целостности.

### OAuth2 & Open ID Connect

Дальнейшее усовершенствование процесса понадобилось ввиду того, что токен-аутентификация требует присутствия пользователя в момент получения доступа к защищенному ресурсу. Потому что Identity provider при передаче ему управления будет с пользователем взаимодействовать, запрашивая, например, логин и пароль.

В случае сервиса, который от имени пользователя должен через определенные промежутки времени опрашивать некий третий ресурс, — допустим, получать доступ к списку контактов в социальной сети — токен-аутентификация работать уже не будет. Дело в том, что идентификаторы сессии обычно живут очень недолго, чтобы в случае их перехвата злоумышленники получили доступ к сервису лишь на ограниченное время. Но из-за короткого срока действия токена не хватает, например, на ночной процесс.

В 2006 году в ходе работы над реализацией протокола Open ID для Twitter обнаружилась потребность в новом открытом протоколе авторизации. В 2007 инженеры Google и AOL начали совместную работу над ним, а в 2009 Twitter предложил своим пользователям решение, делегировавшее сторонним сервисам доступ к аккаунтам и основанное на протоколе OAuth. Три года спустя была опубликована новая версия — OAuth 2, упростившая разработку клиентских приложений и получившая целый ряд новых возможностей, среди которых оказалось и обновление токена без участия пользователя. Многие сервисы начали использовать этот протокол еще до его официального утверждения.

В данный момент на слуху следующие протоколы:

1. OpenID — для проверки учетных данных пользователя (identification & authentication).
2. OAuth — про то, чтобы получать доступ к чему-то.
3. OpenID Connect — и про и то, и про другое одновременно.

Все три протокола позволяют пользователю не разглашать свои секретные логин и пароль недоверенным приложениям.

OpenID Сonnect нужен, чтобы получить у пользователя его учетные данные и проверить их. OAuth 2.0 нужен, чтобы получать токены доступа и с ними обращаться к ресурсам.

## ТЕРМИНОЛОГИЯ OAUTH2 & OPENID CONNECT

* OpenID Connect Provider (OP)
* Client
* User
* Scope
  + Identity scopes – openid, profile, email
  + Resource scopes – various API
* Authentication/Token Request
* Identity Token
* Access Token
* Refresh Token

Сервис выдачи токенов

Open ID Connect Provider — важнейший объект всей конструкции централизованного сервиса аутентификации, он также может называться Security Token Service, Identity Provider authorization server и т. д. Различные источники называют его по-разному, но по смыслу это сервис, который выдает токены клиентам.

Основные функции:

* Аутентифицировать пользователей, используя внутреннее хранилище пользователей или внешний источник (например, Active Directory).
* Управлять клиентами (хранить) и аутентифицировать их.
* Предоставлять управление сессией и возможность реализации Single sing-on.
* Выдавать identity-токены и access-токены клиентам.
* Проверять ранее выданные токены.

Клиент

Client — устройство или программа (браузер, приложение), которым требуется либо токен для аутентификации пользователя, либо токен для доступа к какому-то ресурсу (подразумевается, что данный ресурс «знаком» с тем конкретным «[Security Token Service](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/" \l "token)» у которого клиент запрашивает токен для доступа).

Пользователь

User — собственно конечный пользователь — человек.

Область (scope)

Scope — идентификатор ресурса, к которому клиент хочет получить доступ. Список scope посылается в адрес [сервиса выдачи токенов](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/#token) в составе [запроса на аутентификацию](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/#authrequest).

По умолчанию все клиенты имеют возможность запрашивать любые области, но это можно (и нужно) ограничивать в конфигурации [сервиса выдачи токенов](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/#authrequest).

Scopes бывают двух видов:

1. Identity scopes — это запрос информации о пользователе. Его имя, профиль, пол, фотография, адрес электронной почты и т. д.
2. Resource scopes — имена внешних ресурсо (Web APIs), к которым клиент хочет получить доступ.

Запрос на аутентификацию

Authentication/Token Request — процесс запроса аутентификации.

В зависимости от того какие [области (scopes)](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/#scope) запрошены, [сервис выдачи токенов](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/#token) вернет:

1. Только Identity Token, если запрошены только Identity scopes.
2. Identity Token и Access Token, если запрошены также и Resources scopes.
3. Access Token и Refresh Token, если запрошeн Offline Access.

Токен личности

Identity Token — подтверждение аутентификации. Этот токен содержит минимальный набор информации о пользователе.

Токен доступа

Access Token — информация, что конкретному пользователю разрешается делать. Клиент запрашивает Access Token и затем использует его для доступа к ресурсам (Web APIs). Access Token содержит информацию о клиенте и пользователе, если она присутствует. Важно понимать, что есть такие типы авторизации, при которых пользователь в процессе непосредственно не участвует (подробнее в раздел «[типы авторизации](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/" \l "authTypes)»)

Токен обновления

Refresh Token — токен, по которому STS вернет новый Access Token. В зависимости от режима, работа Refresh Token может быть многоразовым и одноразовым. В случае с одноразовым токеном, при запросе нового Access Token будет также сформирован готовый Refresh Token, который следует использовать при повторном обновлении. Очевидно, что одноразовые токены более безопасны.

## ПРОЦЕСС АУТЕНТИФИКАЦИИ

При обращении пользователя к клиенту, тот перенаправляет пользователя на Open ID Connect Provider, который запрашивает у пользователя логин и пароль. В случае успешного прохождения проверки параметров аутентификации он возвращает назад [identity token](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/#identitytoken) и [access token](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/#accesstoken), с которыми пользователь может обращаться к защищенному ресурсу.

## СТРУКТУРА ТОКЕНА

### Формат

В реализации OAuth2 используется так называемый jwt-токен, который состоит из трех частей. Допустим, при обращении к Identity provider вы отправляете логин/пароль и в ответ получаете токен. Он будет включать в себя: Header (заголовок), Payload (контент) и Signature (подпись). На сайте [jwt.io](http://jwt.io/) его можно декодировать и посмотреть содержимое формате JSON. На этом сайте вы также найдете описание правил формирования jwt-токенов.

В том, что токены в процессе обмена передаются незашифрованными, ничего страшного нет. Мы изначально исходим из предположения, что коммуникация происходит по защищенному HTTPS-каналу, и повторное шифрование токена было бы избыточным. Единственное, в чем нам нужно убедиться – то, что токен не был подменен или сфальсифицирован на клиентской стороне, для этого достаточно иметь подпись и проверять ее на сервере. Кроме того, токен не содержит никакой критически важной информации.

Кроме identity tokens, есть еще и аccess tokens, которые содержат информацию о выданных пользователю клеймах. Срок действия access token достаточно короткий, потому что его хищение может обеспечить несанкционированный доступ к ресурсу. Т. е. злоумышленник, если ему удастся заполучить токен этого типа, доступ получит на очень непродолжительное время. Для получения нового access token используется refresh token, который обычно не фигурирует в незащищенных средах, в частности в режиме доступа из браузера он вообще не используется. Какие именно токены будут возвращены клиенту в процессе аутентификации, разбирается в разделе «[типы авторизации](https://dataart.ua/news/autentifikatsiya-i-avtorizatsiya-v-mikroservisny-h-prilozheniyah/#authTypes)».

### Основные поля

Кратно остановимся на том, какие есть стандартные полях в токене и зачем они нужны:

* iss — адрес или имя удостоверяющего центра.
* sub — идентификатор пользователя. Уникальный в рамках удостоверяющего центра, как минимум.
* aud — имя клиента для которого токен выпущен.
* exp — срок действия токена.
* nbf — время, начиная с которого может быть использован (не раньше чем).
* iat — время выдачи токена.
* jti — уникальный идентификатор токен (нужен, чтобы нельзя был «выпустить» токен второй раз).