Я расскажу о применении различных способов аутентификации для веб-приложений, включая аутентификацию по паролю, по сертификатам, по одноразовым паролям, по ключам доступа и по токенам. Коснусь технологии единого входа (Single Sign-On), рассмотрю различные стандарты и протоколы аутентификации.  
  
Перед тем, как перейти к техническим деталям, давайте немного освежим терминологию.

* **Идентификация** — это заявление о том, кем вы являетесь. В зависимости от ситуации, это может быть имя, адрес электронной почты, номер учетной записи, итд.
* **Аутентификация** — предоставление доказательств, что вы на самом деле есть тот, кем идентифицировались (от слова “authentic” — истинный, подлинный).
* **Авторизация** — проверка, что вам разрешен доступ к запрашиваемому ресурсу.

Например, при попытке попасть в закрытый клуб вас *идентифицируют* (спросят ваше имя и фамилию), *аутентифицируют* (попросят показать паспорт и сверят фотографию) и *авторизуют*(проверят, что фамилия находится в списке гостей), прежде чем пустят внутрь.  
  
Аналогично эти термины применяются в компьютерных системах, где традиционно под *идентификацией* понимают получение вашей учетной записи (identity) по username или email; под *аутентификацией* — проверку, что вы знаете пароль от этой учетной записи, а под *авторизацией* — проверку вашей роли в системе и решение о предоставлении доступа к запрошенной странице или ресурсу.  
  
Однако в современных системах существуют и более сложные схемы аутентификации и авторизации, о которых я расскажу далее. Но начнем с простого и понятного.

**Аутентификация по многоразовому паролю**

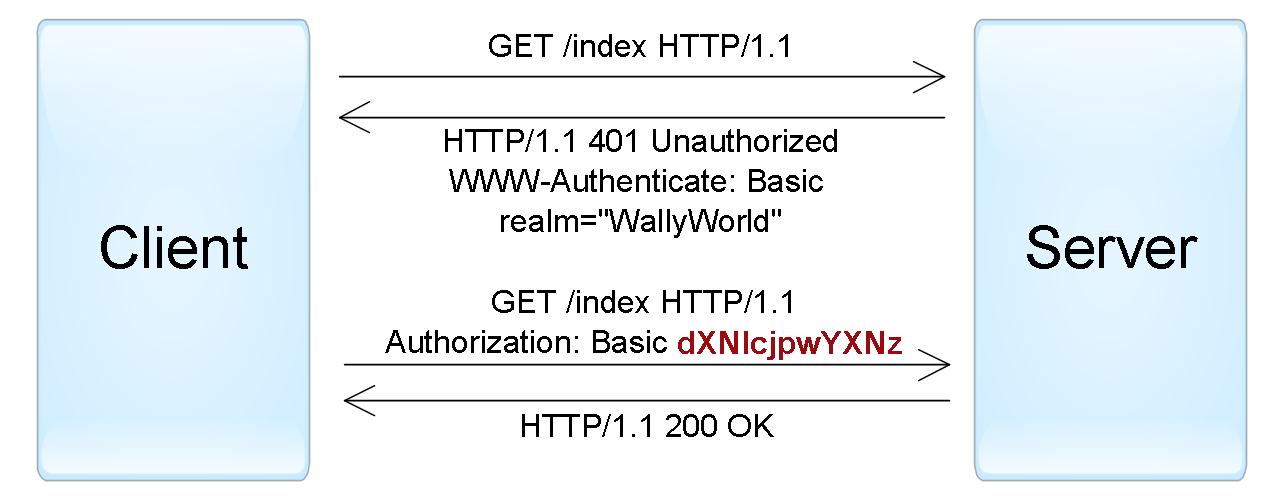
Этот метод основывается на том, что пользователь должен предоставить username и password для успешной идентификации и аутентификации в системе. Пара username/password задается пользователем при его регистрации в системе, при этом в качестве username может выступать адрес электронной почты пользователя.  
  
Применительно к веб-приложениям, существует несколько стандартных протоколов для аутентификации по паролю, которые мы рассмотрим ниже.

**HTTP authentication**

Этот протокол, описанный в стандартах HTTP 1.0/1.1, существует очень давно и до сих пор активно применяется в корпоративной среде. Применительно к веб-сайтам работает следующим образом:

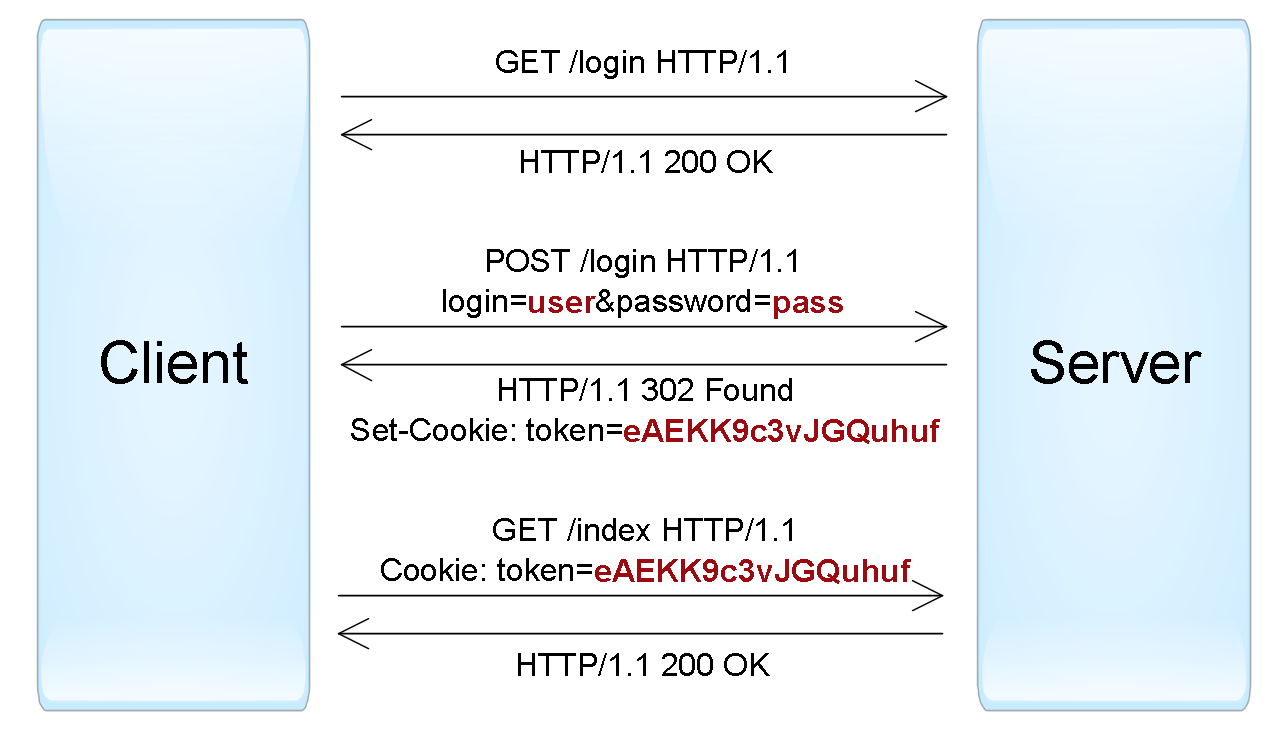
1. Сервер, при обращении неавторизованного клиента к защищенному ресурсу, отсылает HTTP статус “401 Unauthorized” и добавляет заголовок “WWW-Authenticate” с указанием схемы и параметров аутентификации.
2. Браузер, при получении такого ответа, автоматически показывает диалог ввода username и password. Пользователь вводит детали своей учетной записи.
3. Во всех последующих запросах к этому веб-сайту браузер автоматически добавляет HTTP заголовок “Authorization”, в котором передаются данные пользователя для аутентификации сервером.
4. Сервер аутентифицирует пользователя по данным из этого заголовка. Решение о предоставлении доступа (авторизация) производится отдельно на основании роли пользователя, ACL или других данных учетной записи.

Весь процесс стандартизирован и хорошо поддерживается всеми браузерами и веб-серверами. Существует несколько схем аутентификации, отличающихся по уровню безопасности:

1. **Basic** — наиболее простая схема, при которой username и password пользователя передаются в заголовке Authorization в незашифрованном виде (base64-encoded). Однако при использовании HTTPS (HTTP over SSL) протокола, является относительно безопасной.  
     
   *Пример HTTP аутентификации с использованием Basic схемы.*
2. **Digest** — challenge-response-схема, при которой сервер посылает уникальное значение nonce, а браузер передает MD5 хэш пароля пользователя, вычисленный с использованием указанного nonce. Более безопасная альтернативв Basic схемы при незащищенных соединениях, но подвержена man-in-the-middle attacks (с заменой схемы на basic). Кроме того, использование этой схемы не позволяет применить современные хэш-функции для хранения паролей пользователей на сервере.
3. **NTLM** (известная как Windows authentication) — также основана на challenge-response подходе, при котором пароль не передается в чистом виде. Эта схема не является стандартом HTTP, но поддерживается большинством браузеров и веб-серверов. Преимущественно используется для аутентификации пользователей Windows Active Directory в веб-приложениях. Уязвима к pass-the-hash-атакам.
4. **Negotiate** — еще одна схема из семейства Windows authentication, которая позволяет клиенту выбрать между NTLM и Kerberos аутентификацией. Kerberos — более безопасный протокол, основанный на принципе Single Sign-On. Однако он может функционировать, только если и клиент, и сервер находятся в зоне intranet и являются частью домена Windows.

Стоит отметить, что при использовании HTTP-аутентификации у пользователя нет стандартной возможности выйти из веб-приложения, кроме как закрыть все окна браузера.

**Forms authentication**

Для этого протокола нет определенного стандарта, поэтому все его реализации специфичны для конкретных систем, а точнее, для модулей аутентификации фреймворков разработки.  
  
Работает это по следующему принципу: в веб-приложение включается HTML-форма, в которую пользователь должен ввести свои username/password и отправить их на сервер через HTTP POST для аутентификации. В случае успеха веб-приложение создает session token, который обычно помещается в browser cookies. При последующих веб-запросах *session token* автоматически передается на сервер и позволяет приложению получить информацию о текущем пользователе для авторизации запроса.  
  
  
*Пример forms authentication.*  
  
Приложение может создать session token двумя способами:

1. Как идентификатор аутентифицированной сессии пользователя, которая хранится в памяти сервера или в базе данных. Сессия должна содержать всю необходимую информацию о пользователе для возможности авторизации его запросов.
2. Как зашифрованный и/или подписанный объект, содержащий данные о пользователе, а также период действия. Этот подход позволяет реализовать stateless-архитектуру сервера, однако требует механизма обновления сессионного токена по истечении срока действия. Несколько стандартных форматов таких токенов рассматриваются в секции «Аутентификация по токенам».

Необходимо понимать, что перехват session token зачастую дает аналогичный уровень доступа, что и знание username/password. Поэтому все коммуникации между клиентом и сервером в случае forms authentication должны производиться только по защищенному соединению HTTPS.

**Другие протоколы аутентификации по паролю**

Два протокола, описанных выше, успешно используются для аутентификации пользователей на веб-сайтах. Но при разработке клиент-серверных приложений с использованием веб-сервисов (например, iOS или Android), наряду с HTTP аутентификацией, часто применяются нестандартные протоколы, в которых данные для аутентификации передаются в других частях запроса.  
  
Существует всего несколько мест, где можно передать username и password в HTTP запросах:

1. **URL query** — считается небезопасным вариантом, т. к. строки URL могут запоминаться браузерами, прокси и веб-серверами.
2. **Request body** — безопасный вариант, но он применим только для запросов, содержащих тело сообщения (такие как POST, PUT, PATCH).
3. **HTTP header** —оптимальный вариант, при этом могут использоваться и стандартный заголовок Authorization (например, с Basic-схемой), и другие произвольные заголовки.