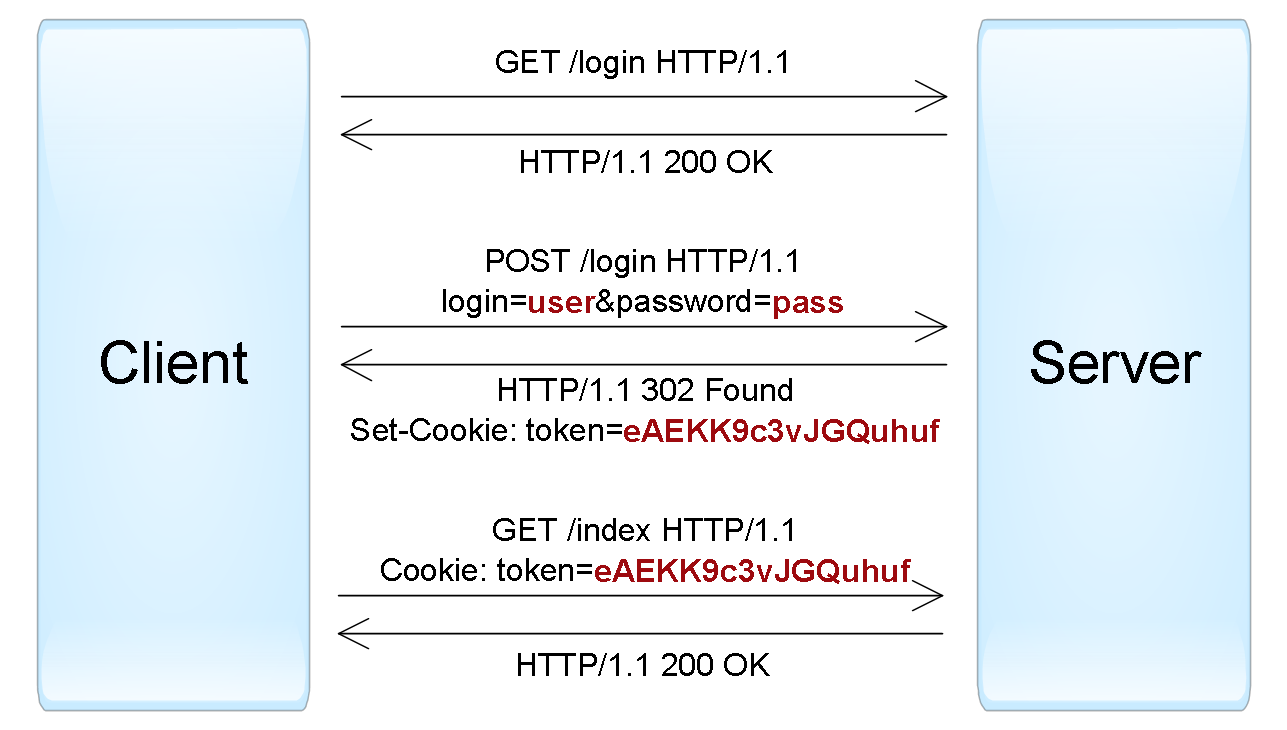
1. **Опишите схему FORMS-аутентификации с использованием сессий.**

Для этого протокола нет определенного стандарта, поэтому все его реализации специфичны для конкретных систем, а точнее, для модулей аутентификации фреймворков разработки.

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.



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».

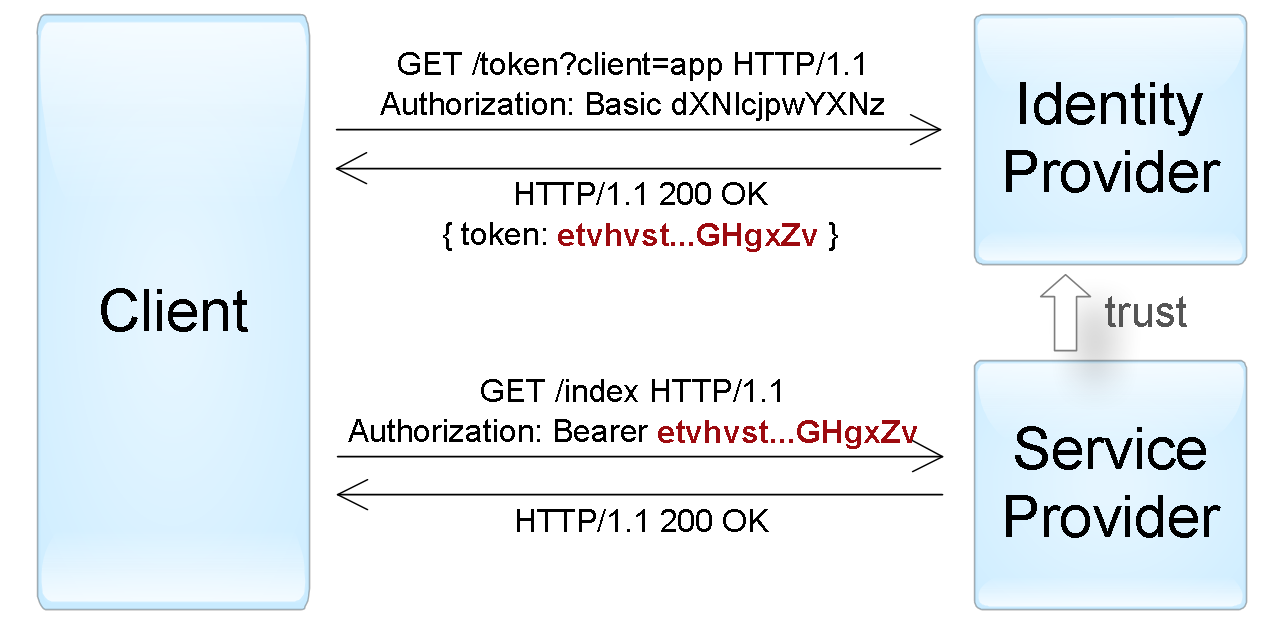
Необходимо понимать, что перехват session token зачастую дает аналогичный уровень доступа, что и знание username/password. Поэтому все коммуникации между клиентом и сервером в случае forms authentication должны производиться только по защищенному соединению HTTPS.

1. **Опишите схему FORMS-аутентификации с использованием токенов.**

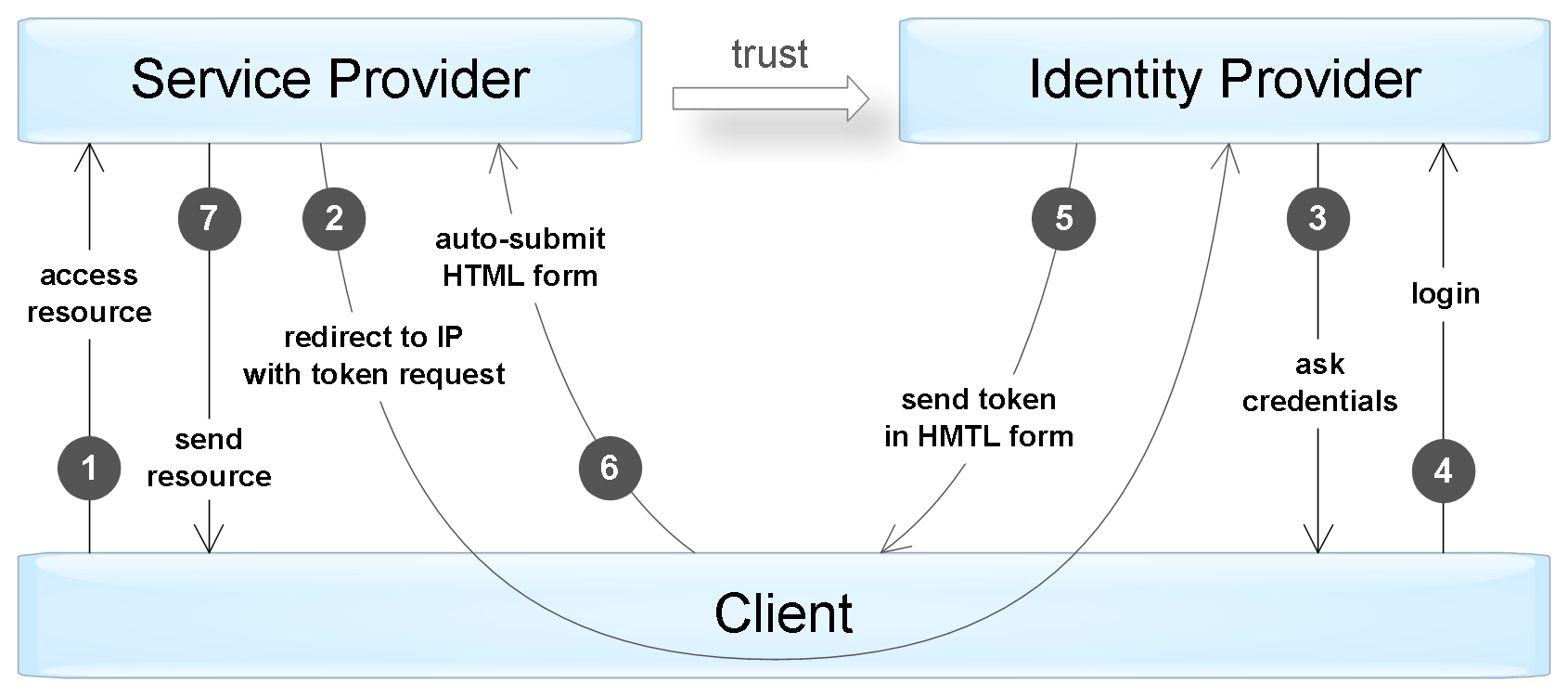
Такой способ аутентификации чаще всего применяется при построении распределенных систем *Single Sign-On* (SSO), где одно приложение (*service provider* или *relying party*) делегирует функцию аутентификации пользователей другому приложению (*identity provider* или *authentication service*). Типичный пример этого способа — вход в приложение через учетную запись в социальных сетях. Здесь социальные сети являются сервисами аутентификации, а приложение доверяет функцию аутентификации пользователей социальным сетям.

Реализация этого способа заключается в том, что identity provider (IP) предоставляет достоверные сведения о пользователе в виде токена, а service provider (SP) приложение использует этот токен для идентификации, аутентификации и авторизации пользователя.  
На общем уровне, весь процесс выглядит следующим образом:

1. Клиент аутентифицируется в identity provider одним из способов, специфичным для него (пароль, ключ доступа, сертификат, Kerberos, итд.).
2. Клиент просит identity provider предоставить ему токен для конкретного SP-приложения. Identity provider генерирует токен и отправляет его клиенту.
3. Клиент аутентифицируется в SP-приложении при помощи этого токена.



Процесс, описанный выше, отражает механизм аутентификации активного клиента, т. е. такого, который может выполнять запрограммированную последовательность действий (например, iOS/Android приложения). Браузер же — пассивный клиент в том смысле, что он только может отображать страницы, запрошенные пользователем. В этом случае аутентификация достигается посредством автоматического перенаправления браузера между веб-приложениями identity provider и service provider.



Существует несколько стандартов, в точности определяющих протокол взаимодействия между клиентами (активными и пассивными) и IP/SP-приложениями и формат поддерживаемых токенов. Среди наиболее популярных стандартов — OAuth, OpenID Connect, SAML, и WS-Federation.

Сам токен обычно представляет собой структуру данных, которая содержит информацию, кто сгенерировал токен, кто может быть получателем токена, срок действия, набор сведений о самом пользователе (claims). Кроме того, токен дополнительно подписывается для предотвращения несанкционированных изменений и гарантий подлинности.

При аутентификации с помощью токена SP-приложение должно выполнить следующие проверки:

1. Токен был выдан доверенным identity provider приложением (проверка поля *issuer*).
2. Токен предназначается текущему SP-приложению (проверка поля *audience*).
3. Срок действия токена еще не истек (проверка поля *expiration date*).
4. Токен подлинный и не был изменен (проверка подписи).

В случае успешной проверки SP-приложение выполняет авторизацию запроса на основании данных о пользователе, содержащихся в токене.

**Форматы токенов**

Существует несколько распространенных форматов токенов для веб-приложений:

**Simple Web Token (SWT)** — наиболее простой формат, представляющий собой набор произвольных пар имя/значение в формате кодирования HTML form. Стандарт определяет несколько зарезервированных имен: Issuer, Audience, ExpiresOn и HMACSHA256. Токен подписывается с помощью симметричного ключа, таким образом оба IP- и SP-приложения должны иметь этот ключ для возможности создания/проверки токена.

**JSON Web Token (JWT)** — содержит три блока, разделенных точками: заголовок, набор полей (claims) и подпись. Первые два блока представлены в JSON-формате и дополнительно закодированы в формат base64. Набор полей содержит произвольные пары имя/значения, притом стандарт JWT определяет несколько зарезервированных имен (iss, aud, exp и другие). Подпись может генерироваться при помощи и симметричных алгоритмов шифрования, и асимметричных. Кроме того, существует отдельный стандарт, отписывающий формат зашифрованного JWT-токена.

**Security Assertion Markup Language (SAML)**— определяет токены (SAML assertions) в XML-формате, включающем информацию об эмитенте, о субъекте, необходимые условия для проверки токена, набор дополнительных утверждений (statements) о пользователе. Подпись SAML-токенов осуществляется при помощи ассиметричной криптографии. Кроме того, в отличие от предыдущих форматов, SAML-токены содержат механизм для подтверждения владения токеном, что позволяет предотвратить перехват токенов через man-in-the-middle-атаки при использовании незащищенных соединений.