1. Перечислите все виды http-аутентификации, какие из них описаны в RFC?
2. Парольная аутентификация - используется для проверки сочетания имени пользователя и пароля.
3. Аутентификация OAuth - позволяет пользователям использовать свои учетные записи на сторонних сайтах для аутентификации на вашем сайте.
4. JWT-аутентификация - основана на токенах JSON Web, которые представляют собой зашифрованный JSON-объект, содержащий информацию о пользователе и его правах.
5. Аутентификация с использованием сертификатов - используется для аутентификации клиентов с помощью сертификатов SSL.
6. Аутентификация на основе протокола Kerberos - используется для централизованного управления учетными записями и аутентификации.

Некоторые из этих видов аутентификации могут быть описаны в RFC (Request for Comments), которые являются официальными документами, описывающими стандарты Интернета. Например, аутентификация OAuth описана в RFC 6749, а аутентификация на основе протокола Kerberos - в RFC 4120.

1. Опишите схему BASIC-аутентификации.

* **BASIC-аутентификация (Basic Access Authentication)** - это простой метод аутентификации, который использует передачу имени пользователя и пароля в виде незашифрованного текста в заголовке Authorization запроса HTTP.

Схема BASIC-аутентификации выглядит следующим образом:

1. Клиент отправляет запрос HTTP на сервер.
2. Сервер отправляет ответ с кодом 401 (Unauthorized) и в заголовке WWW-Authenticate указывает, что для доступа к ресурсу требуется аутентификация с помощью схемы BASIC.
3. Клиент повторно отправляет запрос с заголовком Authorization, содержащим имя пользователя и пароль, разделенные двоеточием и закодированные в формате Base64.
4. Сервер проверяет имя пользователя и пароль, и если они совпадают с записью в базе данных, отправляет запрошенные данные клиенту.

Схема BASIC-аутентификации является простой и распространенной, но не очень безопасной из-за незашифрованного передачи имени пользователя и пароля.

1. Опишите схему DIGEST-аутентификации.

* **DIGEST-аутентификация (Digest Access Authentication)** - это метод аутентификации, который использует хеширование пароля и предотвращает передачу пароля в открытом виде.

Схема DIGEST-аутентификации выглядит следующим образом:

1. Клиент отправляет запрос HTTP на сервер.
2. Сервер отправляет ответ с кодом 401 (Unauthorized) и в заголовке WWW-Authenticate указывает, что для доступа к ресурсу требуется аутентификация с помощью схемы DIGEST.
3. Клиент повторно отправляет запрос с заголовком Authorization, содержащим информацию о пользователе, ранее предоставленную сервером.
4. Сервер отправляет клиенту случайный набор данных (nonce), который клиент должен захешировать вместе с логином, паролем и другой информацией.
5. Клиент создает хеш, используя метод хеширования и передает его вместе с логином, случайным числом (nonce), временем и другой информацией в заголовке Authorization.
6. Сервер использует алгоритм хеширования, чтобы сравнить хеш, созданный клиентом, с хешем, созданным сервером. Если они совпадают, сервер разрешает доступ клиенту.

DIGEST-аутентификация является более безопасной, чем BASIC-аутентификация, так как предотвращает передачу пароля в открытом виде. Однако, схема DIGEST также может быть скомпрометирована при использовании слабых алгоритмов хеширования и медленных алгоритмов шифрования.

1. Почему в чистом виде http-аутентификация не является надежной?

* **Нешифрованный текст** - информация, передаваемая в HTTP-аутентификации, не защищена и передается в открытом виде. Это означает, что любой, кто может перехватывать сетевой трафик (например, злоумышленник, который использует программное обеспечение перехвата сетевого трафика), может прочитать логин и пароль.
* **Хранение паролей** - многие пользователи используют один и тот же пароль для нескольких сайтов и сервисов. Если пароль для HTTP-аутентификации становится известным, он может быть использован для получения доступа к другим сервисам.
* **Слабая защита** - HTTP-аутентификация использует только базовые методы аутентификации, такие как Basic и Digest, которые могут быть легко обойдены или взломаны злоумышленниками.

**Идентификация** – заявление пользователя о себе.

**Аутентификация** – процедура проверки подлинности идентификации юзера.

**Авторизация** – процедура проверки прав аутентифицированного пользователя.

**Аутентификация –** процедура, направленная на подтверждение идентификатора пользователя (логин и пароль, как правило). На основании какого-то секрета.

**Авторизация** – после аутентификации; проверка есть ли права на выполнение тех или иных действий.

Исп-ся **1 код возврата** (401 (и 200ый само собой))

401 код – Unauthorized – ваш запрос является не авторизированным.   
407 код – для прокси серверов.

**2 заголовка**, кот.мы будем использовать:

\*authorization

\*www-autherticate

Все открыто, злоумышленники могут работать вместо клиента, надо шифровать.

Схема аутентификации (клиент и сервер)

**Способы аутентификации:**

1. BASIC

Схема

Ключевое слово **basic** от сервера к клиенту – указывает на вид аутентификации

Realm – То, что знает клиент (твоя аутентификация действует в рамках этого сектора), т.е. определяет область действия аутентификации.

Если на стороне клиента – браузер, то 401 код заставляет браузер зажечь окошечко, которое попросит вас ввести логин и пароль.

Base64 строка – сцепленные через «:» имя и пароль в заголовке authorization.

С получает, проверяет валидность, если все хорошо – отвечает 200ым кодом, если нет – сначала все идет, т.к.выдается код 401.

7617 протокол

1. Дайджест - DIGEST

Схема

7616 протокол

Описание схемы:

Идет запрос

Дальше идет ответ 401, в котором указываем заголовки WW-Authenticate и указывает DIGEST. Особенность в том, что указывается не имя и пароль, а хэш имени и пароля.

Как захешировать имя и пароль?

\*добавить домен и т.д.

У С есть БД, где существует хеш и остается проверить валидный хеш или нет.

Является большей защитой, чем basic.

С т.зр. криптографии – также беззащитен как basic.

Исп-ся реже, т.к. не защищает (ур-нь защиты почти как у basic), но больше мороки => исп-ют в основном basic.