**Сookie** - небольшой фрагмент данных, отправленный веб сервером и хранимый в браузере пользователя, это данные, которые приходят с сервера и хранятся на клиенте.

кэш - файлы и картинки, шрифты и т.д.

cookie вспомогательная информация, небольшой объем данных

Например, при авторизации:

1. клиент отправляет запрос на авторизацию с логином и паролем
2. сервер проверяет соответствие и валидность данных
3. сервер в респонс хедер отправляет cookie, которые содержать токен
4. браузер сохраняет данные куки.

при последующих отправках запроса браузер данные токена в cookie отправляет в реквест хедере запроса к серверу

сервер сверяет токен из cookie с токенов сервера, который был сгенерирован и отправлен в процессе авторизации

Cookie хранит:

* токен для управления сеансом
* персонализация (язык, таймзона и т.д.)
* трекинг (для аналитики: фиксирует действия на странице например)

Header set-cookie создает сервер в запросе БЕЗ COOKIE; сервер кладет cookie в этот header для сохранения браузером.

какие атрибуты хранятся в cookie?

* домен, суб домен
* путь (path=/ - такой вид означает, что все возможные пути)
* срок действия максимальный срок жизни
* secure безопасность (используется с https)
* httpOnly содержимое передается исключительно между браузером клиента и сервером во время HTTP-запросов, через консоль вручную нельзя создать через document.cookie, нет доступа к ним через JS
* SameSite функция безопасности для HTTP-cookie, которая определяет, как файлы cookie отправляются вместе с межсайтовыми запросами

Тип данных - строка

Размер: 4KB

https://habr.com/ru/articles/511318/

**XSS Attack Cross-Site Scripting**

внедрение на страницу JS кода, который не предусмотрен разработчиками

1) добавить JavaScript-код в поле ввода, текст из которого сохраняется и в дальнейшем отображается на странице для всех пользователей.

Это может быть поле для ввода информации о себе на странице профиля социальной сети или комментарии на форуме. Когда другие пользователи зайдут на эту же страницу, вместе с текстом они загрузят и JavaScript-код злоумышленника.

Именно в момент загрузки этот код отработает.

Такой текст должен быть специально обработан прежде, чем отдать его браузеру. Такая обработка называется “экранированием”.

В процессе экранирования текста в этом тексте все спец. символы заменяются их “аналогами”, и браузер уже знает наверняка, что это просто текст.

Уязвимое место, требующее проверки: места на сайте, в которых у обычного пользователя есть возможность повлиять на контент.

ВАЖНО! Даже если мы провели валидацию поля на вводимые символы, то сам URL должен также быть обработан, если мы в URL выносим данные, которые ввел пользователь.

2) проверка уязвимости, в cookie для информации по авторизации/платежам/другая личная информация должен обязательно стоять атрибут httpOnly, т.к. cookie без данного атрибута можно забрать через из браузера и использовать

Local storage

Хранилище данных (ключ-значение) без срока давности.

Разница

cookie - с сервера приходят в хэдере

local storage - устанавливает js код браузера

Тип данных - строка

Размер: <5-10MB

session storage - хранилище данных (ключ - значение), которые хранятся только в рамках текущей вкладки

(независимо от того, что это один сайт и один домен, session storage будет разный на разных вкладках)

если вкладка закрывается, то session storage пропадает

Тип данных - строка

Размер: <5MB

Одно из мест в приложении, которое должно быть проверено:

при разлогине пользователя cookie, local storage и session storage должны чиститься

отступление на http:

http - какая версия, которая используется, решает браузер, но сервер может влиять на версию, если версия ему не подходит, то запрос сервер может не принять

устанавливается коннект TLS--> сервер и браузер определяют версию протокола

**Cache** - временная память для повторных запросов.

Промежуточное хранилище данных с быстрым доступом,

содержащая информацию, которая может быть зарошена с наибольшей вероятностью.

данные, которые чаще всего используются и некритичные по актуальности.

кэш - файлы и картинки, шрифты и т.д.

server cache, есть ограничение на хранение

работает с помощью cache service (например Redis)

browser cache

преимущества cache - быстрота

in-memory cache - хранится в оперативной памяти сервера (минусы: небольшой объем, перезагрузка сервака чистит кэш)

хранение: от сервера приходят данные Expires в Cache -Control

ETag, If-None-Match

Last-Modified, If-Modified-Since

версионирование файлов

cache invalidation - актуализация cache

cache API (почитать самому)

Clear- Site- Data Header - сервер присылает браузеру для очистки, но не все браузеры умеют чистить cache [limited support]