

Лекция №5: Авторизация, безопасность

Web-программирование / ПГНИУ

Основные понятия

- **Идентификация:**
 - процедура распознавания пользователя
- **Аутентификация:**
 - процедура проверки подлинности пользователя
- **Авторизация:**
 - предоставление прав на выполнение операции или процедура проверки прав пользователя на выполнение операции

Основные способы аутентификации

- По паролю
- По сертификату
- По ключам доступа
- По токенам (рассмотрим во второй половине курса)
- Web Authentication (WebAuthN)

Авторизация по паролю


- Где передать пароль?
 - URL (query)
 - Тело запроса
 - Заголовки запроса
- Основные подходы
 - HTTP Authentication
 - HTML Forms

HTTP Authentication

1. При запросе к защищённому ресурсу сервер отправляет `401 Unauthorized` с заголовком `WWW-Authenticate`
2. Браузер предоставляет пользователю форму авторизации
3. В последующих запросах браузер будет прикреплять данные авторизации к запросу:
 - Basic: `Authorization: Basic <base64 of login+pass>`
 - Digest: `Authorization: Digest ... hash with secret ...`



Authentication Required

 http://localhost is requesting your username and password. The site says: "Please Login"

User Name:

Password:

Cookies

- Небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя
- Общие данные клиента и сервера, хранящиеся на клиенте
- Отправляются на сервер с HTTP заголовком запроса
- Устанавливаются на клиенте с HTTP заголовком ответа
- `Set-Cookie: <name>=<value>[;<параметры>]`
- `Cookie: <name>=<value>{ ;<name>=<value> }`
- `document.cookie // <name>=<value>{ ;<name>=<value> }`

Основные параметры Cookie

- `Expires`, `Max-Age` – время жизни
- `Secure` - только HTTPS
- `HttpOnly` – недоступны из JavaScript
- Область видимости: `Domain`, `Path`, `SameSite`

Область видимости Cookie

- `Cookie` устанавливается с `Domain` сервера, который её устанавливает
- Дополнительно можно указать `Path`
- При запросе на одном домене отправляются `Cookie`, попадающие в область видимости, а с другого домена зависит от `SameSite`
 - `none` - всегда отправляются
 - `Lax` - отправляются только при "навигации высокого уровня, которая использует "безопасные" HTTP методы": GET, HEAD, OPTIONS и TRACE в переходах и формах
 - `Strict` - не отправляются

Form Authentication with Cookie

- В Cookie можно сохранить данные пользователя после аутентификации?
- Да, но надо сделать это так, чтобы их нельзя было подделать. Подпись, хеширование, шифрование.

Сессия

- Сессия – повторяющееся мероприятие
- Сессия – сеанс - ограниченный во времени период какой-либо деятельности, процесса
- HTTP не имеет сессий в самом протоколе. Каждый запрос-ответ независимы.
- Совокупность связанных между собой запросов, ограниченных каким-то периодом времени, образует сессию
- Сессия должна быть достоверной

Client-side session

- Вся информация текущей сессии хранится на клиенте и передаётся на сервер через Cookie
- Для подлинности подписывается или шифруется
- Нельзя инвалидировать

Server-side session

- Вся информация текущей сессии хранится на сервере:
 - БД
 - Файл
 - Память приложения
- На клиент с cookie передаётся только идентификатор этой сессии
- Можно инвалидировать
- Требуется хранилище на сервере, проблема с масштабированием

Form Authentication

1. Пользователь заполняет форму и отправляет
2. На сервер приходят данные, проверяются, и в сессии сохраняется идентификатор пользователя
3. С каждым запросом приходят Cookie с сессией, в которой есть идентификатор пользователя

Основные угрозы безопасности веб-приложения

- Безопасность серверной части: исполнение данных от пользователя, SQL-инъекции, доступ к файлам и т.д.
- Межсайтовый скриптинг (Cross-Site Scripting, XSS) - внедрение кода на сайт
 - Прямой вывод данных, полученных от пользователя (включая cookie, url и т.д.)
 - Ненадёжные источники
- Межсайтовая подделка запроса (Cross-Site Request Forgery, CSRF, XSRF)

CSRF

1. На сайте злоумышленника создаётся запрос (например, форма) на сайт, где жертва прошла аутентификацию
2. При отправке запроса отправляются Cookie с данными аутентификации жертвы
3. Выполняется действие от имени жертвы

Защита от CSRF

- `Cookie` с `SameSite`
- `CSRF-token`:
 - На страницу добавляется секретный токен
 - Вместе с запросами (формой) отправляется токен, который подтверждает, что запрос выполнен с подлинной страницы

Web Authentication API (WebAuthN)

- Новый стандарт аутентификации в веб-приложениях на основе асимметричного шифрования, поддерживаемый браузерами
- Позволяет через JS регистрировать и авторизовывая пользователя с помощью подписи, которую осуществляет Web-браузер
- В качестве подписи омгут использоваться как внешние (физические) электронные ключи, так и различные платформозависимые

Авторизация

- При большом количестве разных видов операций, субъектов, объектов, правил может быть очень нетривиальной задачей определить, есть ли у пользователя права на выполнение операции. Существует несколько подходов, таких как:
- **ACL** (Access Control List) - список "кто, что, с чем" может делать
- **RBAC** (Role Based Access Control) - управление доступом на основе ролей
- **ABAC** (Attribute-Based Access Control) - разграничение доступа на основе атрибутов

Ссылки

- <https://developer.mozilla.org/ru/docs/Web/HTTP/Куки>
- <https://developer.mozilla.org/ru/docs/Web/HTTP/Авторизация>
- [https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Веб Безопасность](https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Веб_Безопасность)
- <https://webauthn.io>
- Обзор способов и протоколов аутентификации в веб-приложениях: [Habr](#)
- Подходы к контролю доступа: RBAC vs. ABAC: [Habr](#)
- Никто (почти) не знает, что такое авторизация: [Habr](#)

In the next episode

AJAX, API

К теме аутентификации и безопасности мы вернёмся во второй половине курса