Технологии разработки Интернет приложений: Lab4

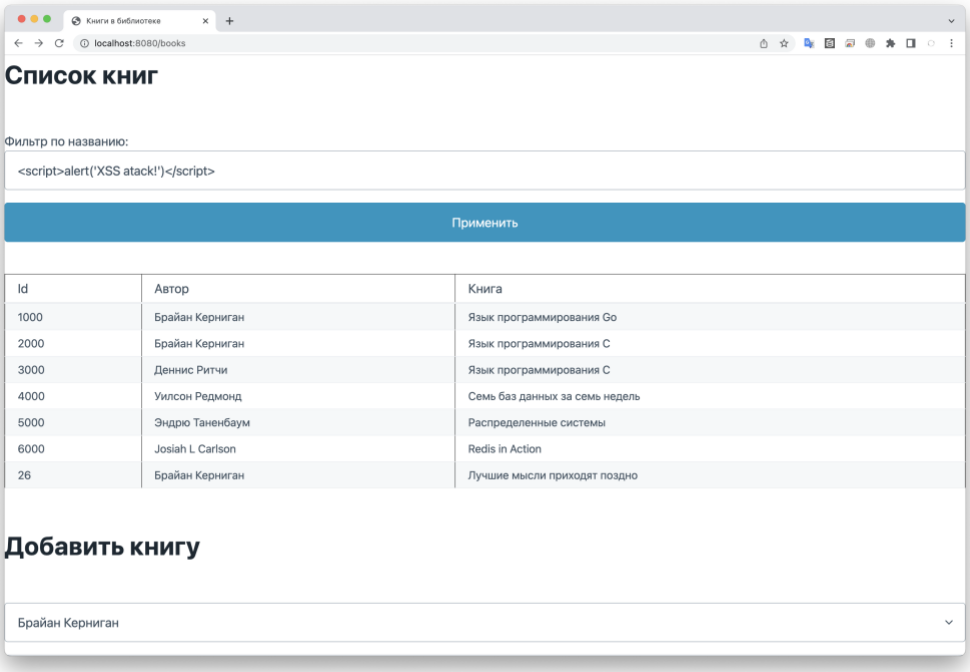
Адилов Марсель
РИ-491223

- ТРИП: Lab4
    - Уязвимости
        - Reflected XSS в поиске книг
        - Persisted (Stored) XSS
        - Cookie injection / Session hijacking
        - Defacing
    - Исправляем уязвимости
        - Reflected XSS в поиске книг
        - Persisted (Stored) XSS / Defacing
        - Cookie injection / Session hijacking

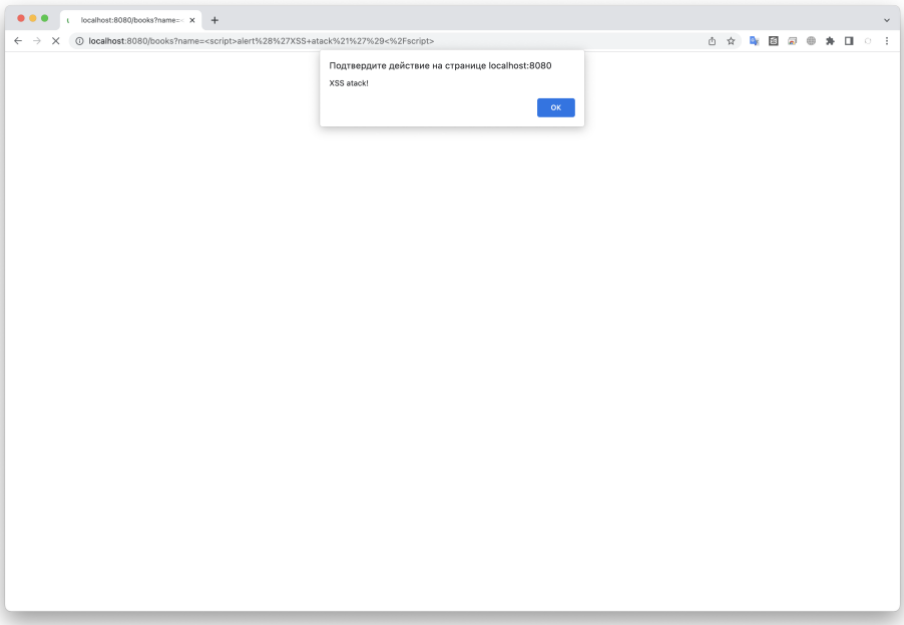
Уязвимости

Reflected XSS в поиске книг

После входа под логином Jhon проверяем поиск книг. Вводим в строку поиска скрипт по вызову баннера `<script>alert('XSS atack!')</script>`.



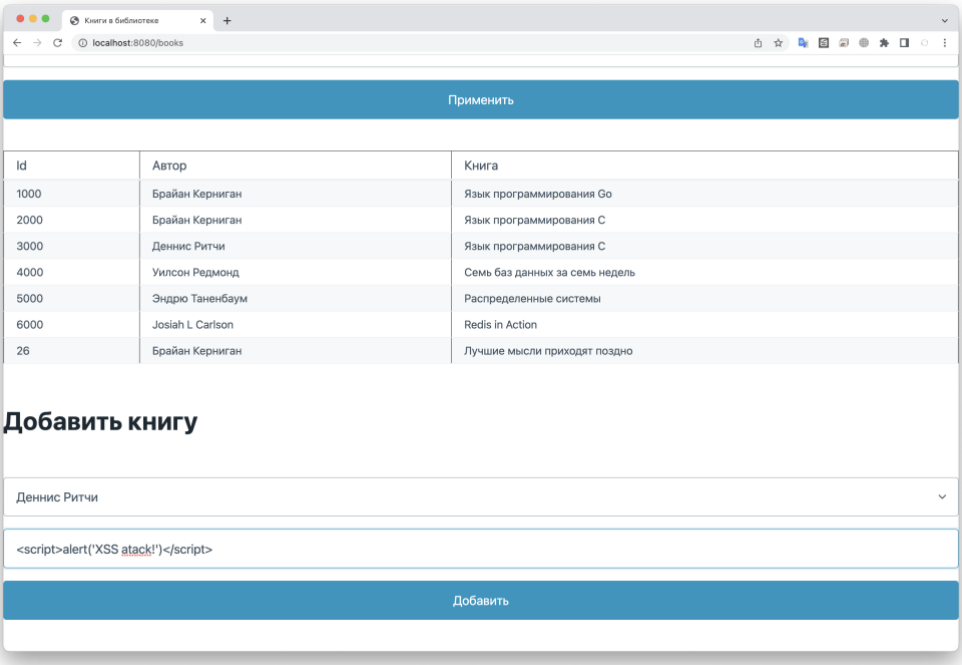
После подтверждения получаем ответ от сервера и видим баннер `XSS atack!`, значит скрипт выполнился успешно.



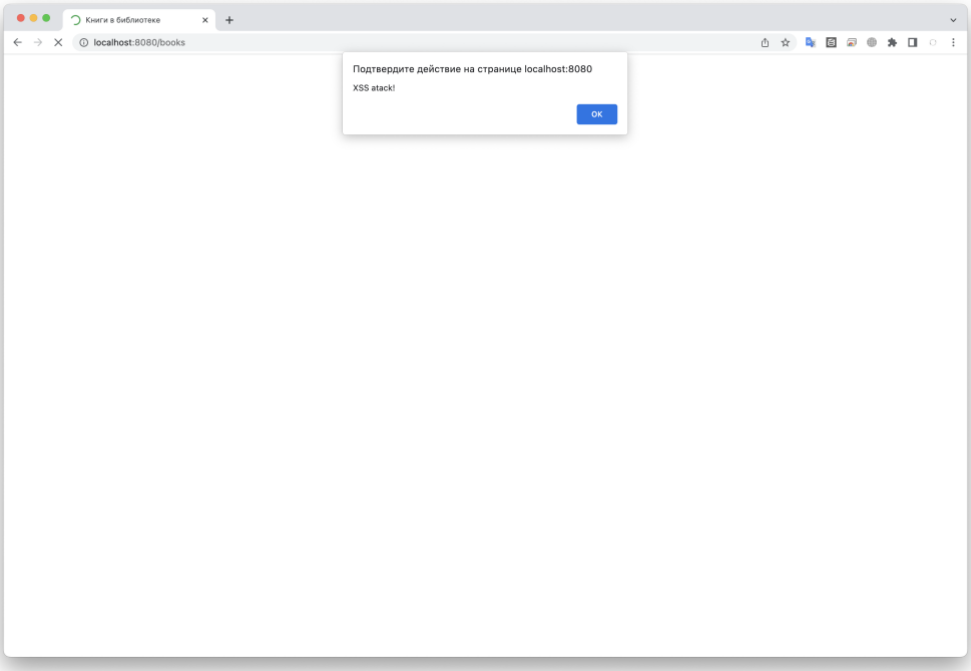
Теперь можно скопировать ссылку `http://localhost:8080/books?name=%3Cscript%3Ealert%28%27XSS+atack%21%27%29%3C%2Fscript%3E`, укоротить её для скрытия видимого слова `script` и отправить жертве.

Persisted (Stored) XSS

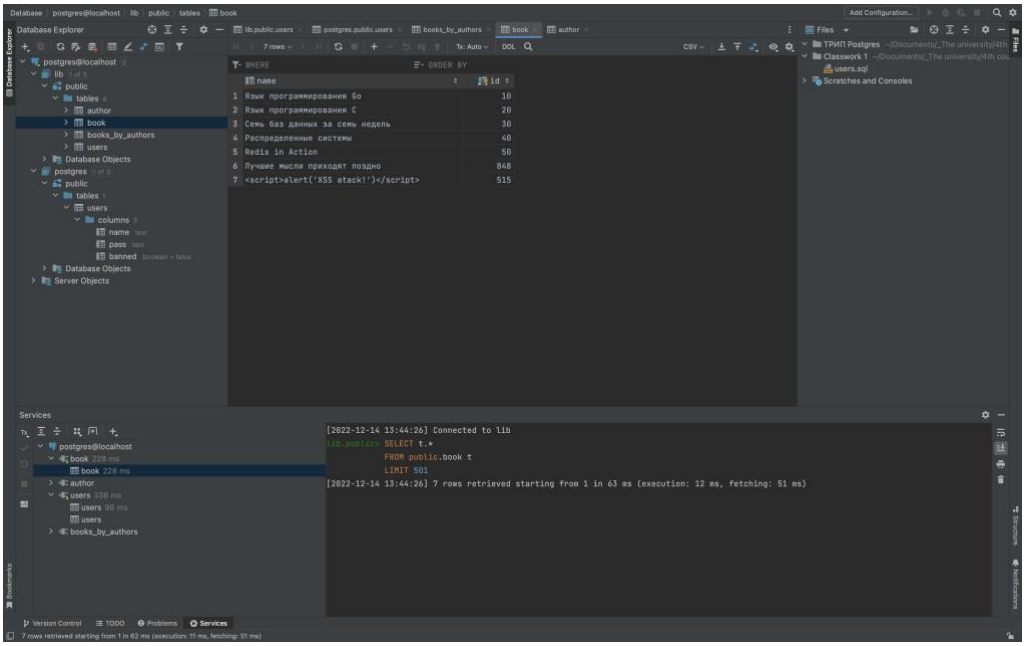
При создании книги вводим в поле скрипт `<script>alert('XSS atack!')</script>`.



Теперь при каждой загрузке страницы со списком книг у пользователей выполняется скрипт и появляется соответствующий баннер.



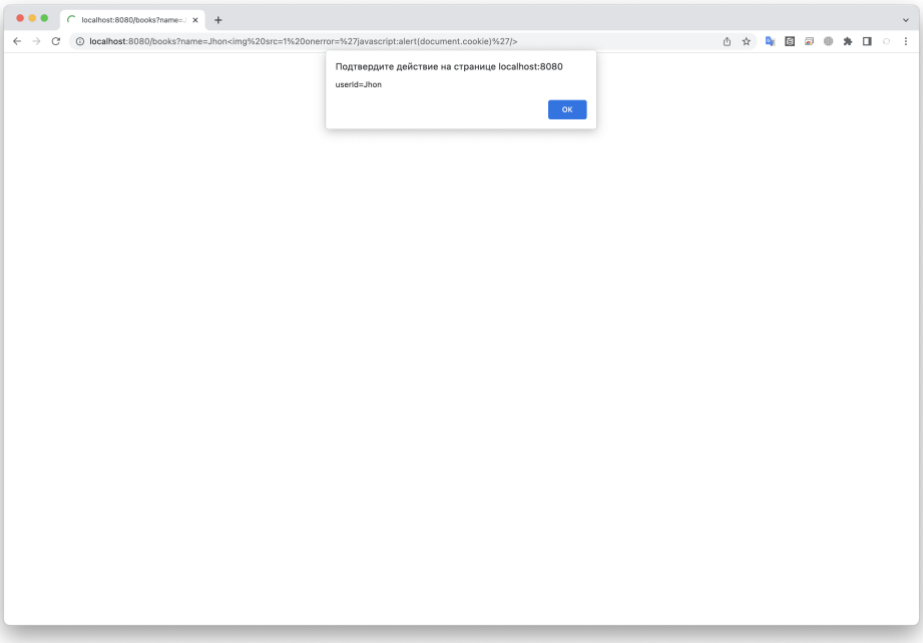
Проверяем в базе данных и видим, что скрипт действительно хранится в таблице книг.



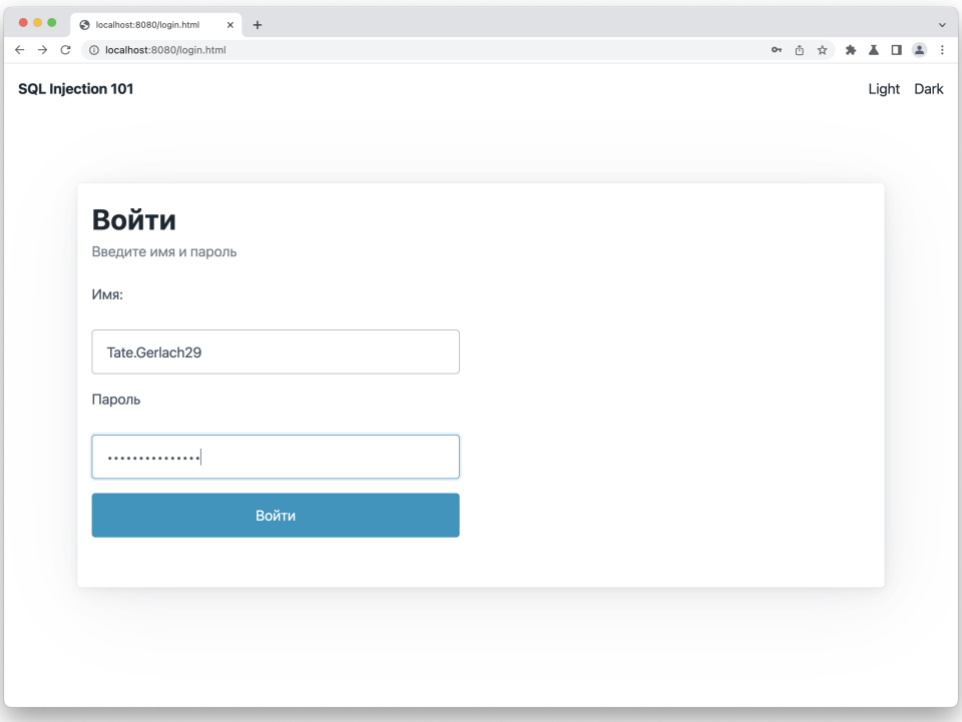
Cookie injection / Session hijacking

После входа на сайт под именем пользователя Jhon вводим в адресную строку `http://localhost:8080/books?name=Jhon<img src=1`

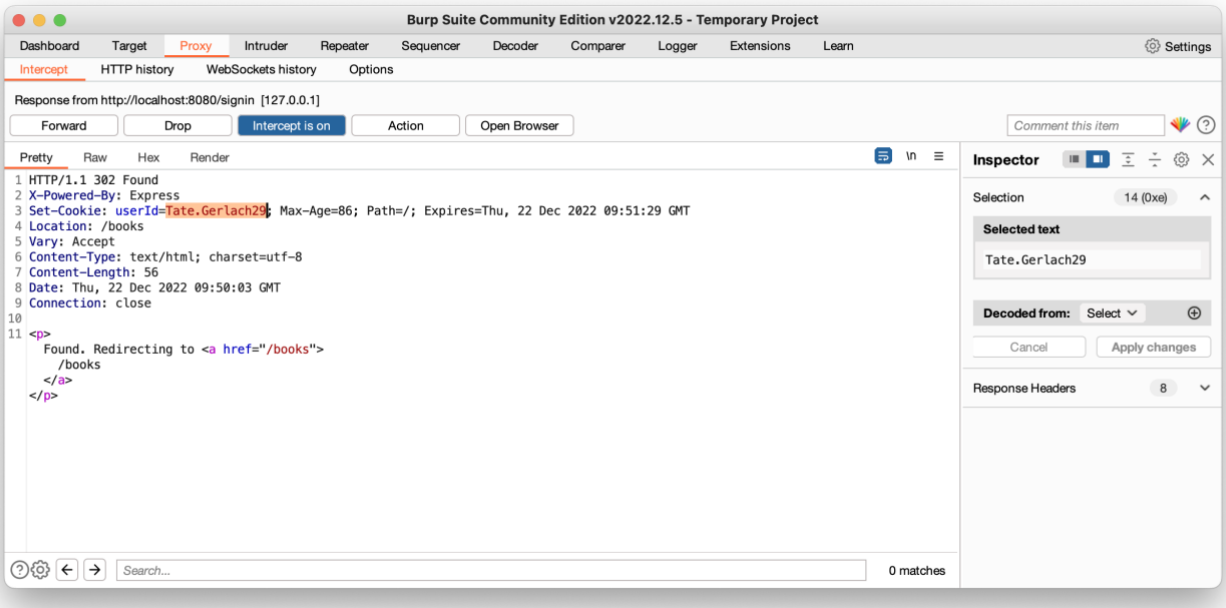
`onerror='javascript:alert(document.cookie)'/>` и получаем на выходе идентификатор Cookie `userID = Jhon`. Используем его для Cookie injection.



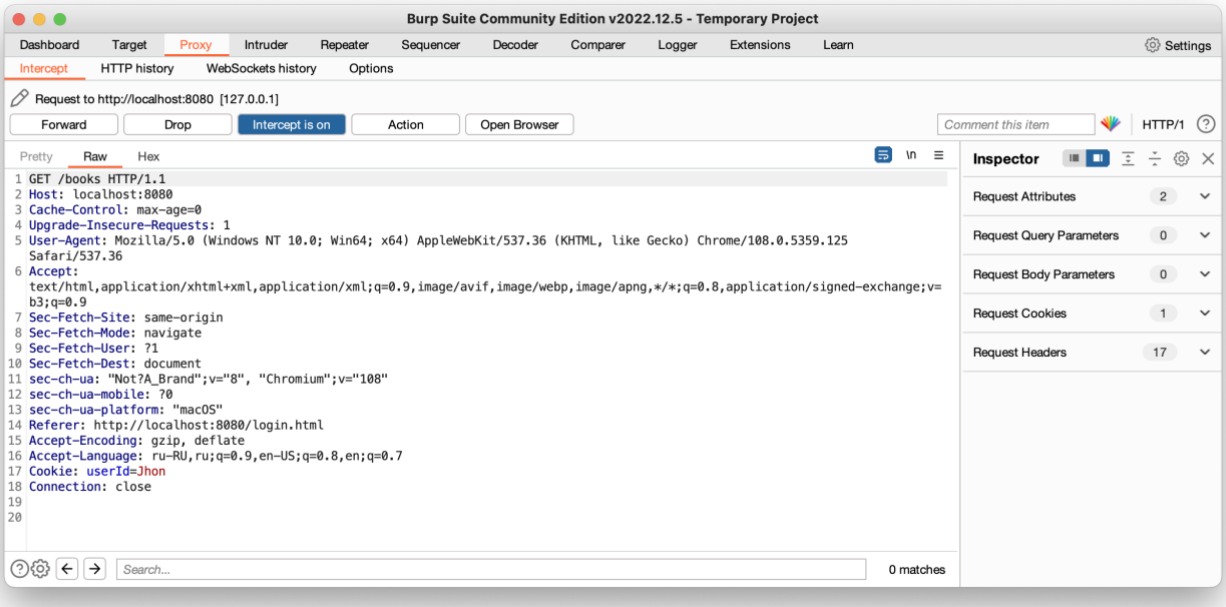
Воспользуемся программой Burp Suite и включим в нем Proxy Intercept. После заходим на страницу входа с другого браузер и вводим данные другого пользователя для успешного входа.



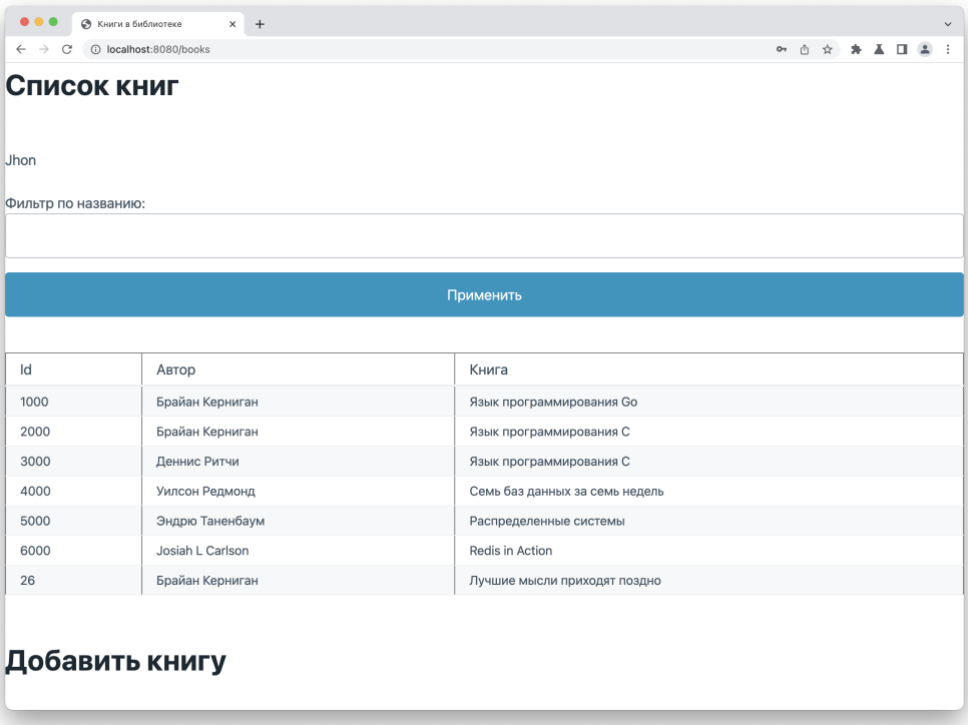
После нажатия кнопки "Войти" в программе Burp Suite мы меняем пойманный ответ в строке `Set-Cookie` с `userID=Tate.Gerlach29` на `userID=Jhon`.



Далее мы посылаем запрос на сервер, и уже видно, что cookie `usedID=Jhon`.

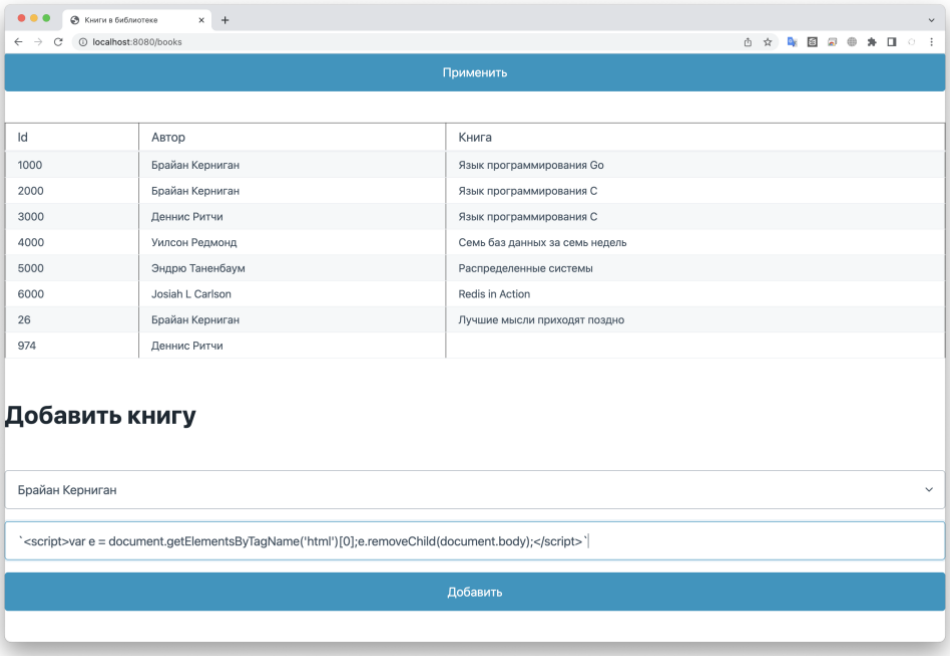


По итогу мы входим на сайт под акаунтом Jhon.

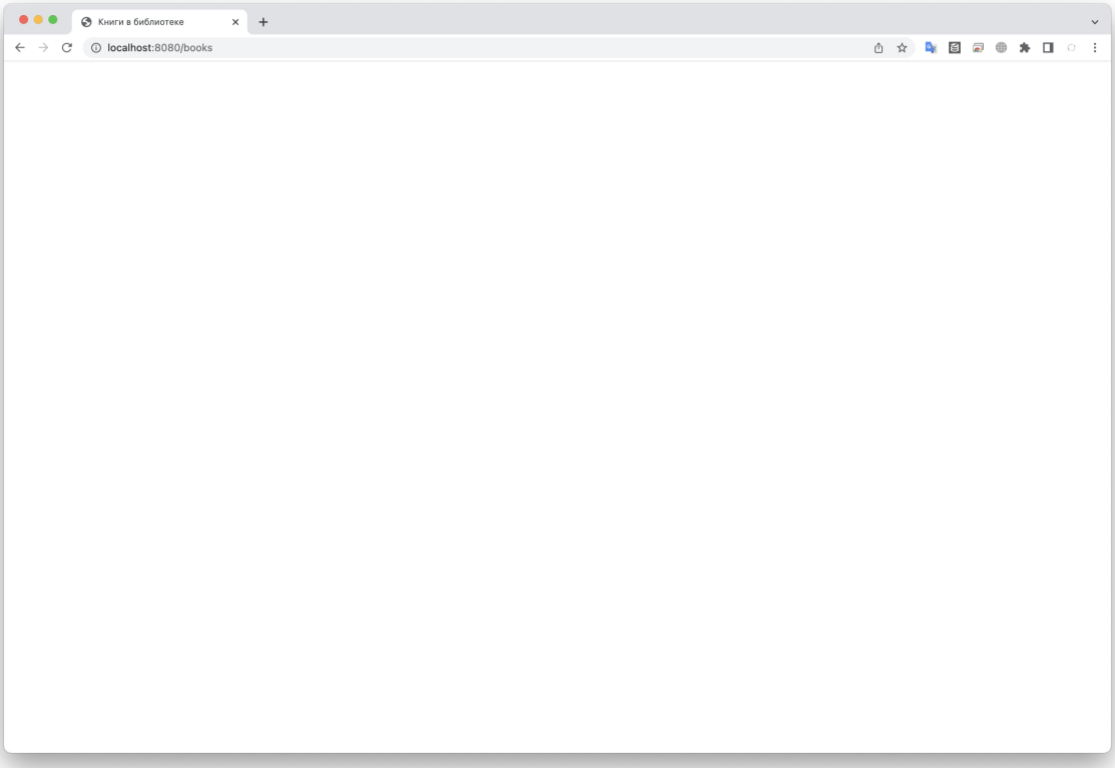


Defacing

В поле добавления книги вводим скрипт по удалению содержимого сайта `<script>var e = document.getElementsByTagName('html')[0];e.removeChild(document.body);</script>`.



После нажатия кнопки "Добавить" сайт перезагружается и ничего не показывает, так как при каждой загрузке страницы выполняется введенный нами скрипт, который попал в таблицу книг.

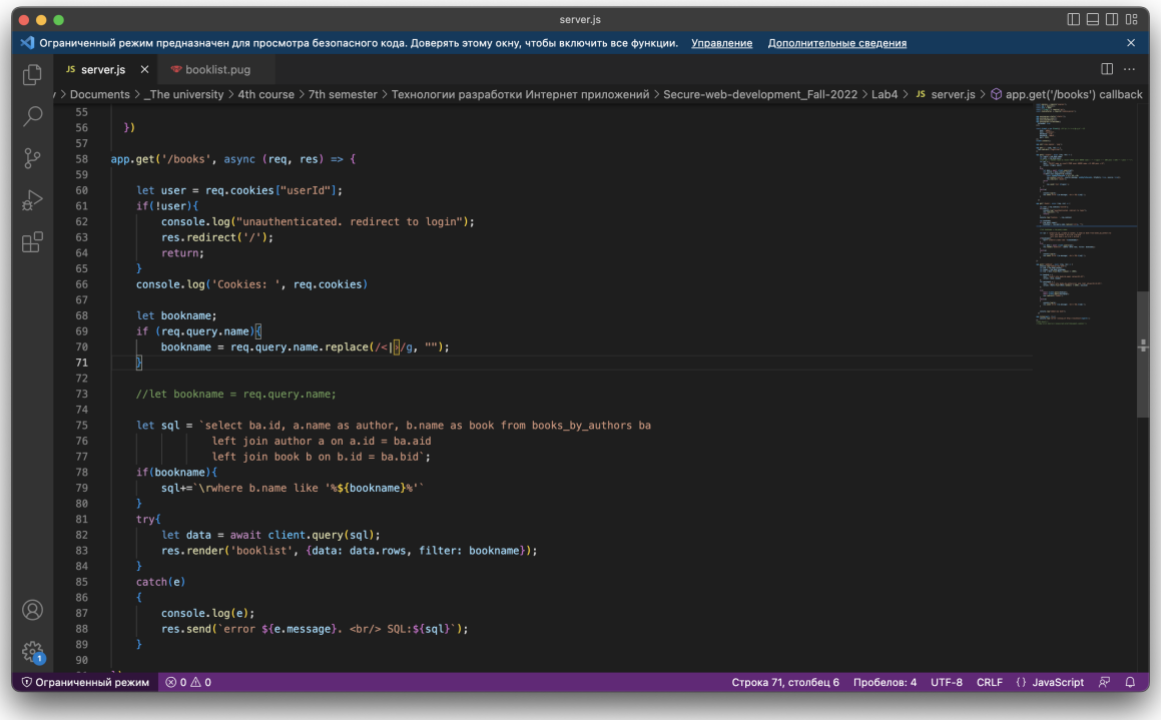

Книги в библиотеке
localhost:8080/books

Исправляем уязвимости

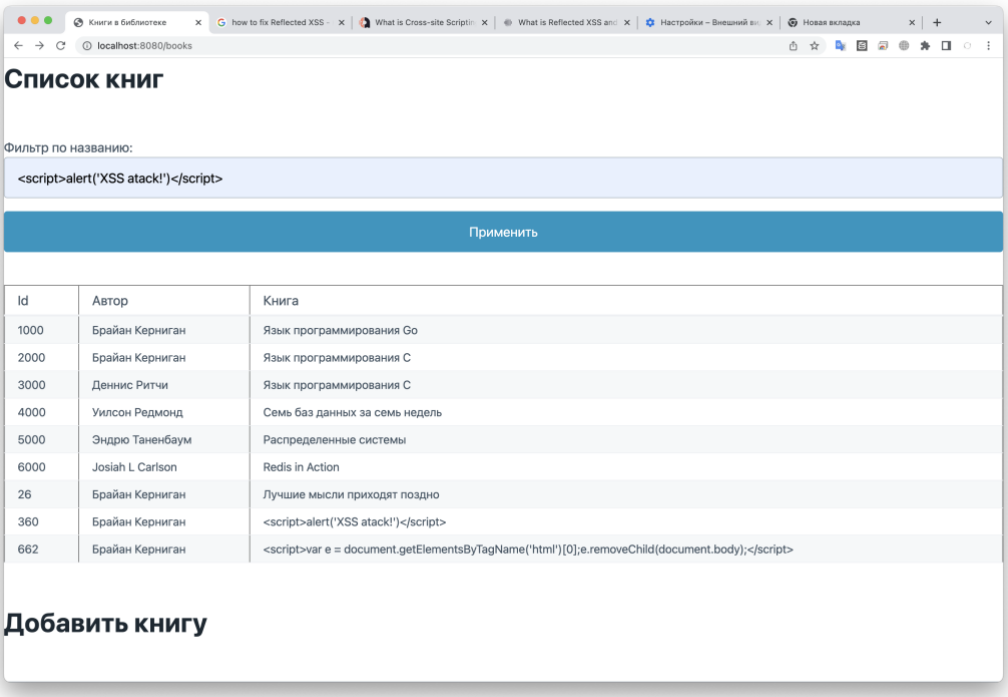
Reflected XSS в поиске книг

В server.js будем делать замену символов:

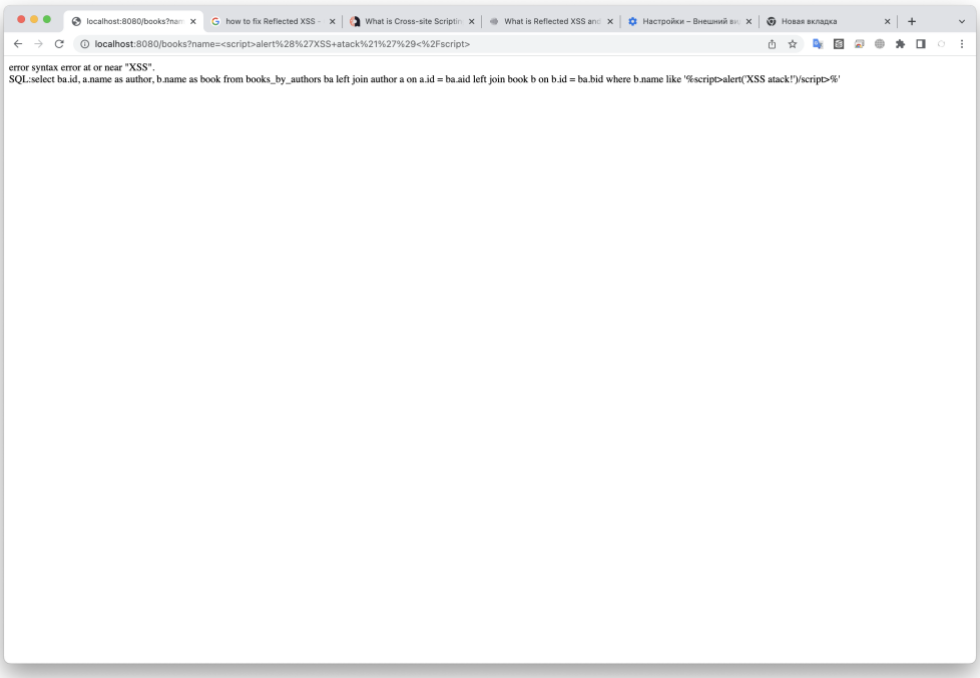
```
let bookname;
    if (req.query.name){
        bookname = req.query.name.replace(/<|›/g, "");
    }
```



Теперь проверим. В поле поиска книги введем скрипт по вызову баннера `<script>alert('XSS atack!')</script>`.

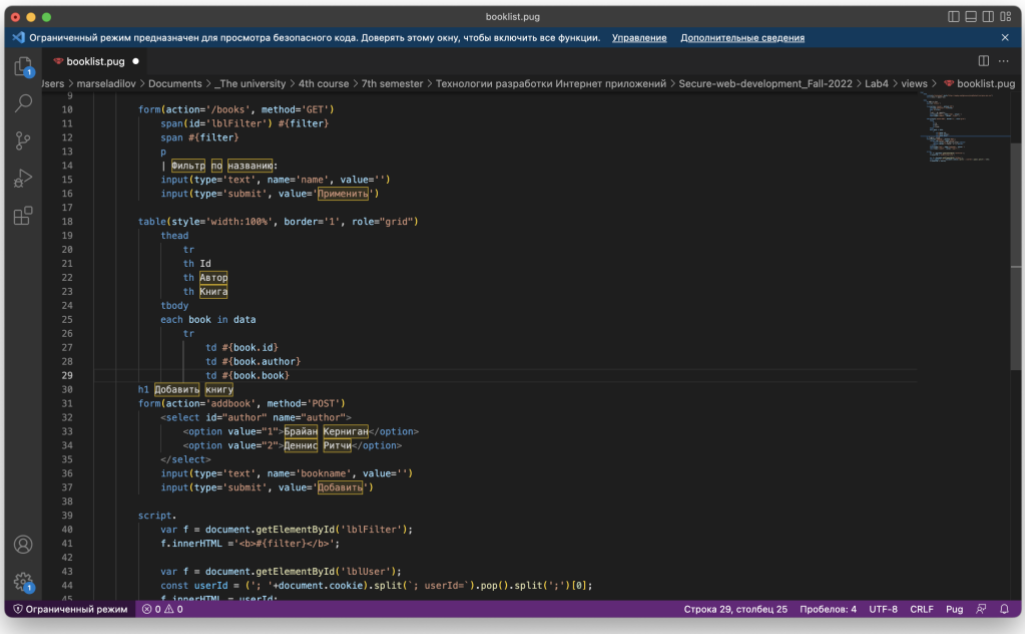


После выполнения никакого баннера не было, следовательно уязвимость закрыта.

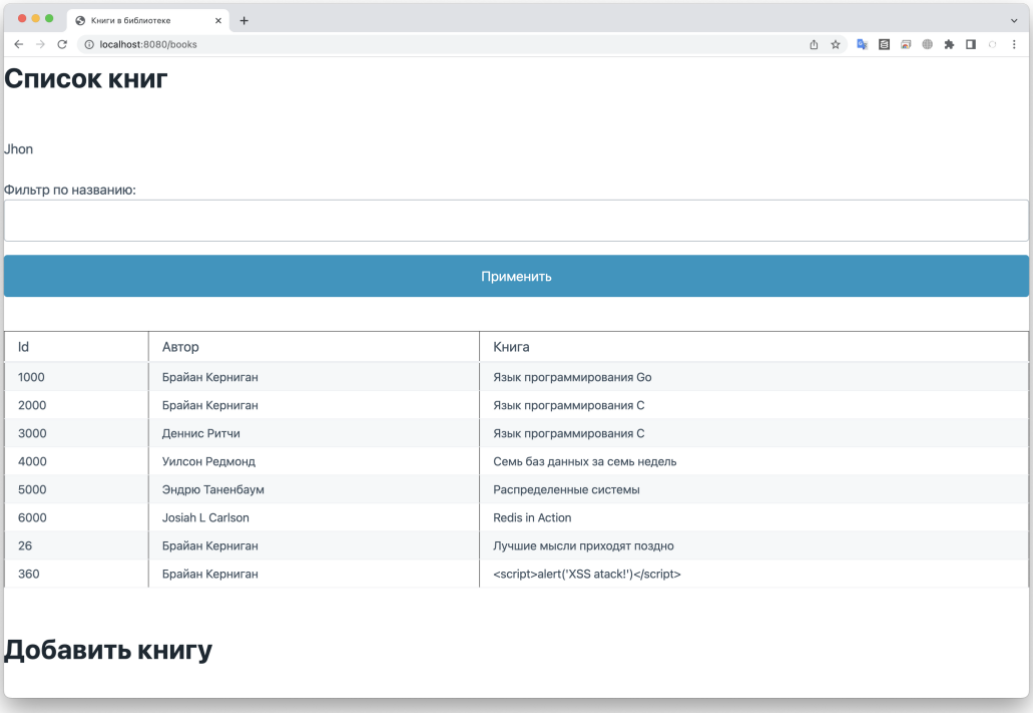


Persisted (Stored) XSS / Defacing

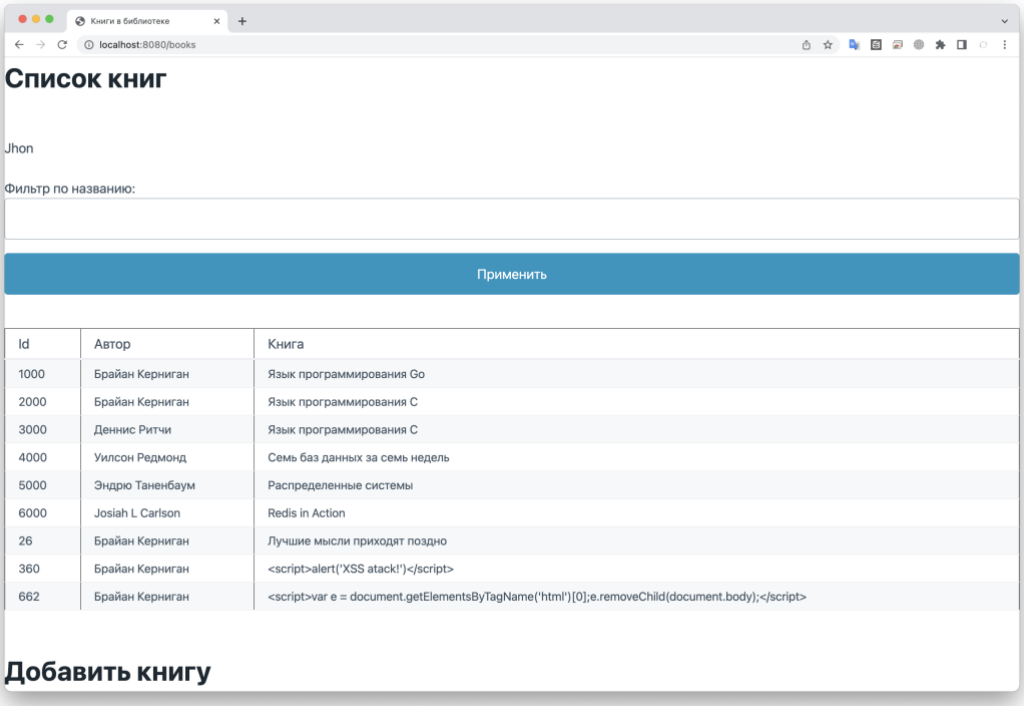
В секции файла booklist.pug, отвечающей за отображение списка книг, для отображения названия книг `!{book.book}` стоит восклицательный знак. Поэтому скрипт срабатывает, ведь код обрабатывается. Для предотвращения поставим решетку `#{book.book}`.



Проверим, впишем в поле добавления книги скрипт по вызову баннера `<script>alert('XSS atack!')</script>`. После добавления страница перезагружается, а баннер не высвечивается. Уязвимость закрыта.

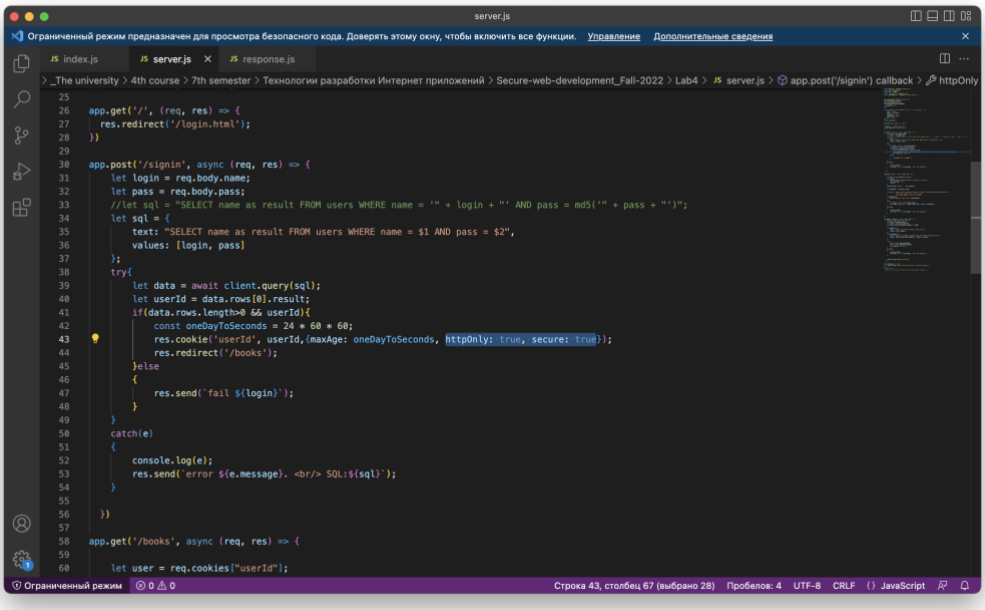


Это также решает уязвимость для Defacing. В поле добавления книги вводим скрипт по удалению содержимого сайта `<script>var e = document.getElementsByTagName('html')[0];e.removeChild(document.body);</script>`. После перезагрузки страницы видим, что все содержимое осталось на месте.



Cookie injection / Session hijacking

В файле `server.js` добавим атрибуты HttpOnly и Secure для Cookie. `res.cookie('userId', userId,{maxAge: oneDayToSeconds, httpOnly: true, secure: true});`.



Теперь проверим, сработало ли. Зайдем на сайт в качестве пользователя Jhon и вставим в адресную строку следующее: `http://localhost:8080/books?name=Jhon<img src=1 onerror='javascript:alert(document.cookie)'/>`. По выполнению нам не высвечивается

идентификатор userID из Cookie, поэтому нам нечего будет использовать для Cookie injection. Уязвимость закрыта.