МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»  
Институт радиоэлектроники и информационных технологий (ИРИТ-РТФ)

**ОБНАРУЖЕНИЕ И ИСПРАВЛЕНИЕ XSS УЯЗВИМОСТЕЙ  
  
Отчет по лабораторной работе**по дисциплине «Разработка безопасных веб-приложений»

Преподаватель Мирвода С.Г.

Студенты Кетель В.В.

Пестов М.А.

дата подпись

Группа РИ – 571227

Код и наименование направления подготовки (специальности):   
10.05.02 Информационная безопасность телекоммуникационных систем

Екатеринбург 2021

1. **Войти на сайт и увидеть список книг и авторов**

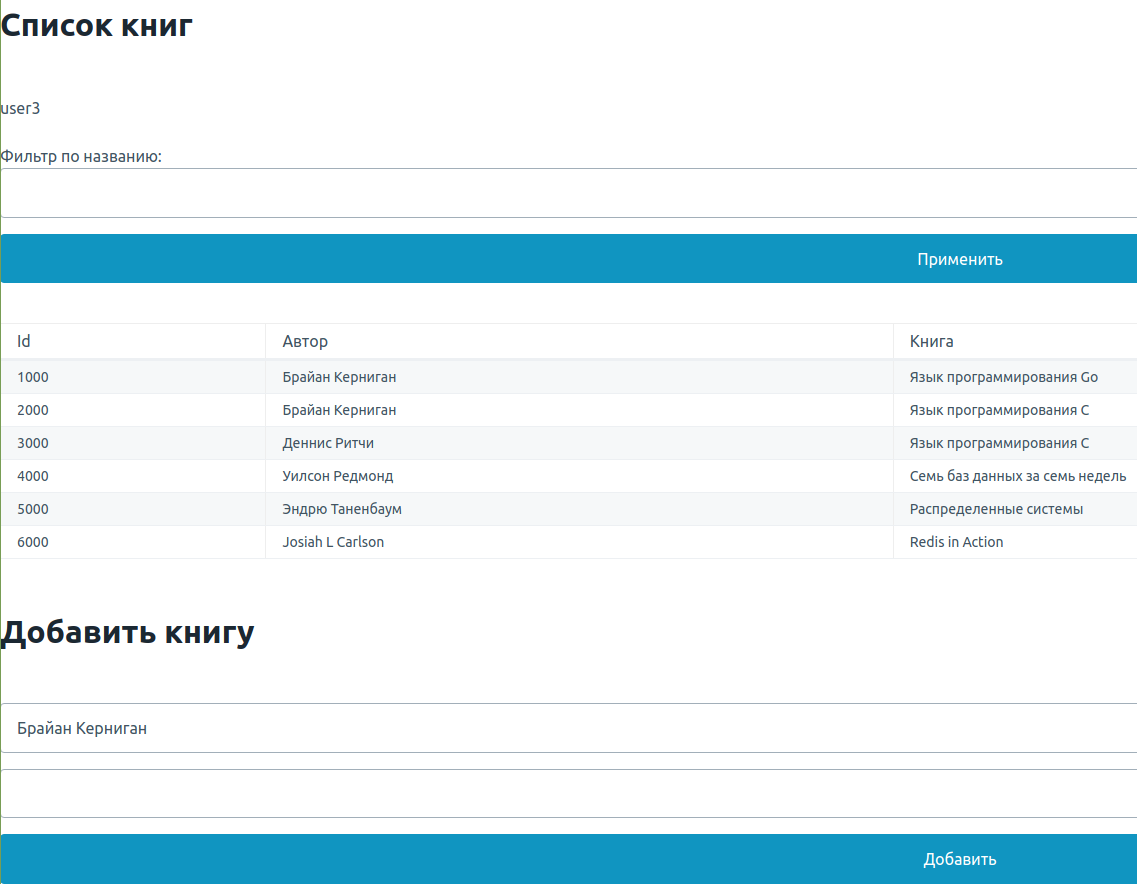


Рисунок 1 – Список книг, доступный после авторизации. Авторизация выполнена под пользователем «user3», пароль «654321». По сравнению с прошлой ЛР добавился раздел «Добавить книгу»

1. **[Исправление уязвимости] Reflected XSS в поиске книг**

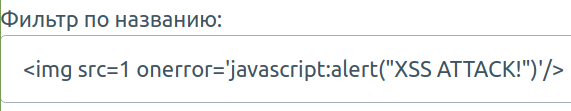


Рисунок 2 – Добавление XSS скрипта в фильтр книг по названию

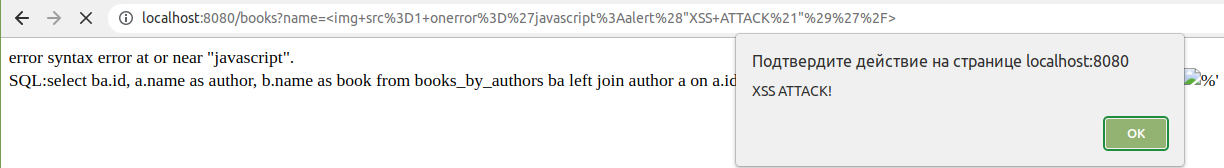


Рисунок 3 – Успешная эксплуатация Reflected XSS. Эта атака возможна из-за того, что данный скрипт ломает SQL запрос и в параметр сравнения вставляется картинка, которая вызывает скрипт XSS

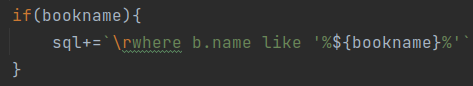


Рисунок 4 – Уязвимое место, где происходит поломка SQL запроса. Необходимо переписать этот фрагмент кода, как в случае с SQL Injection

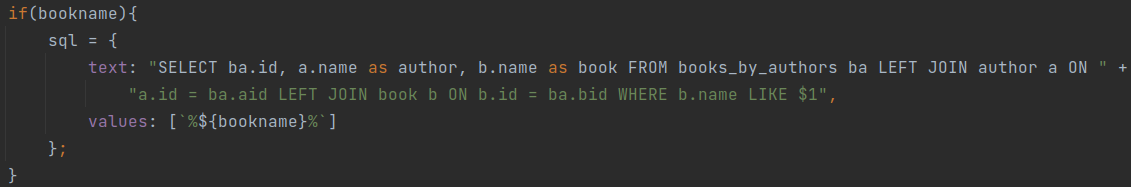


Рисунок 5 – Фрагмент с корректной обработкой SQL запросов. Таким образом, мы одновременно исправим SQL Injection и XSS

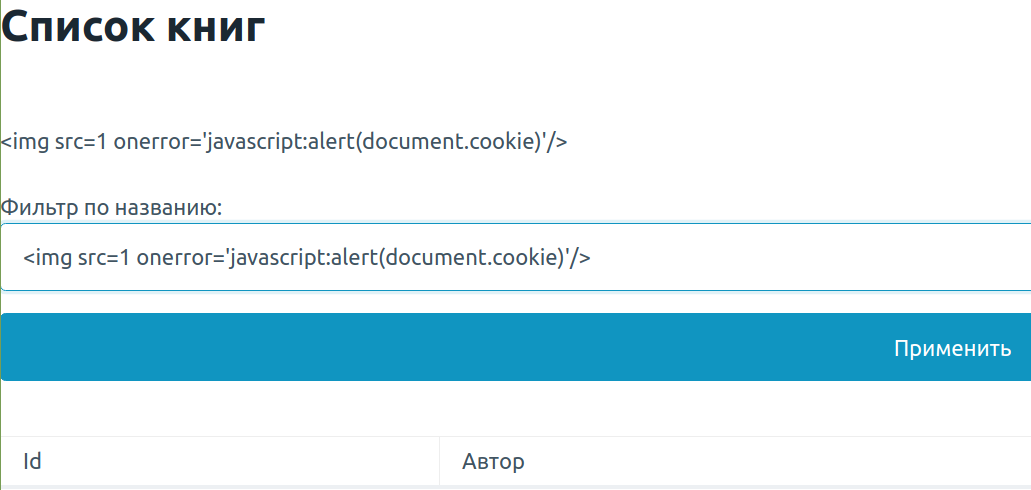


Рисунок 6 – Уязвимость Reflected XSS успешно исправлена и больше не эксплуатируется

1. **[Исправление уязвимости] Persisted (Stored) XSS при создании книги и отображении списка книг**

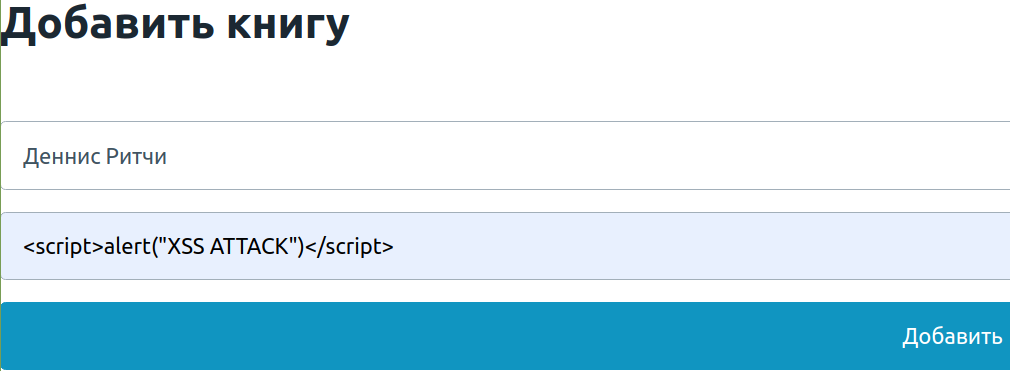


Рисунок 7 – Добавление Stored XSS в список книг

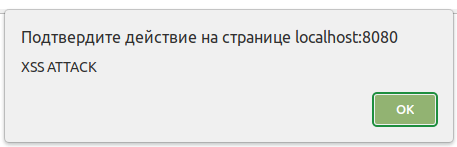


Рисунок 8 – Теперь каждый раз при обновлении страницы — выскакивает результат хранимой XSS

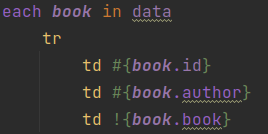


Рисунок 9 – Уязвимое место для хранимой XSS. Чтобы это исправить необходимо отключить интерпретацию html-кода у объекта book.book

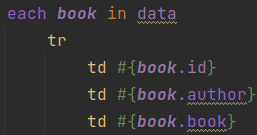


Рисунок 10 – Исправление хранимой XSS. Отключение интерпретации html-кода для объекта book.book с помощью решетки «#»

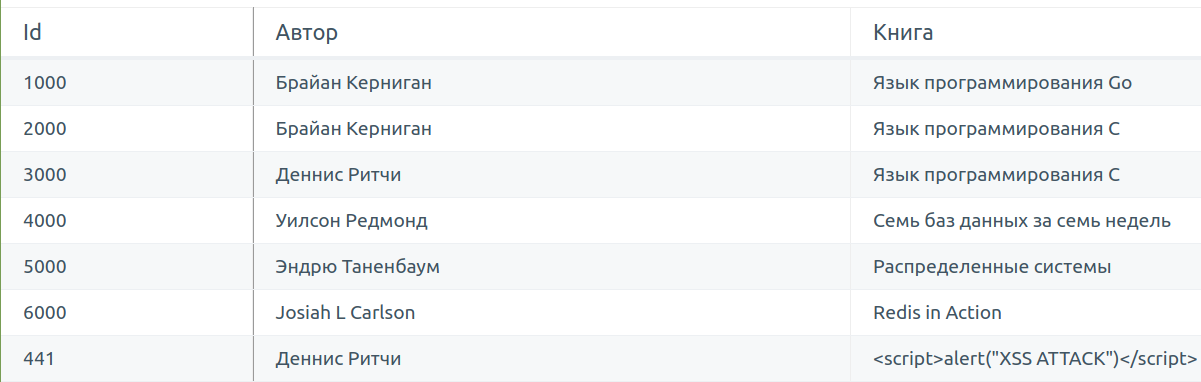


Рисунок 11 – Уязвимость хранимой XSS успешно исправлена

1. **[Исправление уязвимости] Потенциальная уязвимость через Cookie Injection**

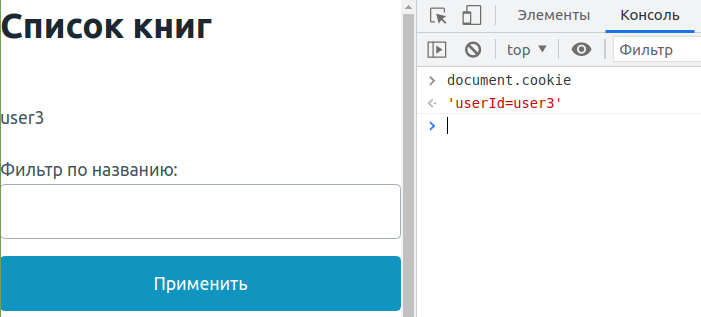


Рисунок 12 – Используя ранее рассмотренные уязвимости злоумышленник может получить доступ к DOM-объекту document.cookie и, например, перенаправить их на свой сервер. Такая уязвимость может быть очень опасной, если злоумышленнику удастся перехватить сессионную куку. В нашем случае сессионная кука — userId=user3



Рисунок 13 – Использование флага httpOnly при инициализации куки, чтобы javascript не мог с ней взаимодействовать. Таким образом, предотвращаем ее кражу

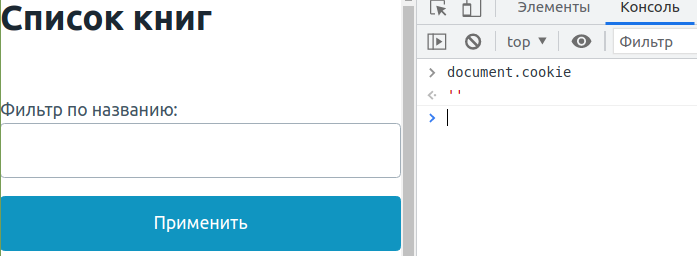


Рисунок 14 – Потенциальная уязвимость Cookie Injection успешно исправлена

1. **[Исправление уязвимости] Некорректно созданная сессионная cookie, которая приводит к захвату сессии (Session hijacking)**

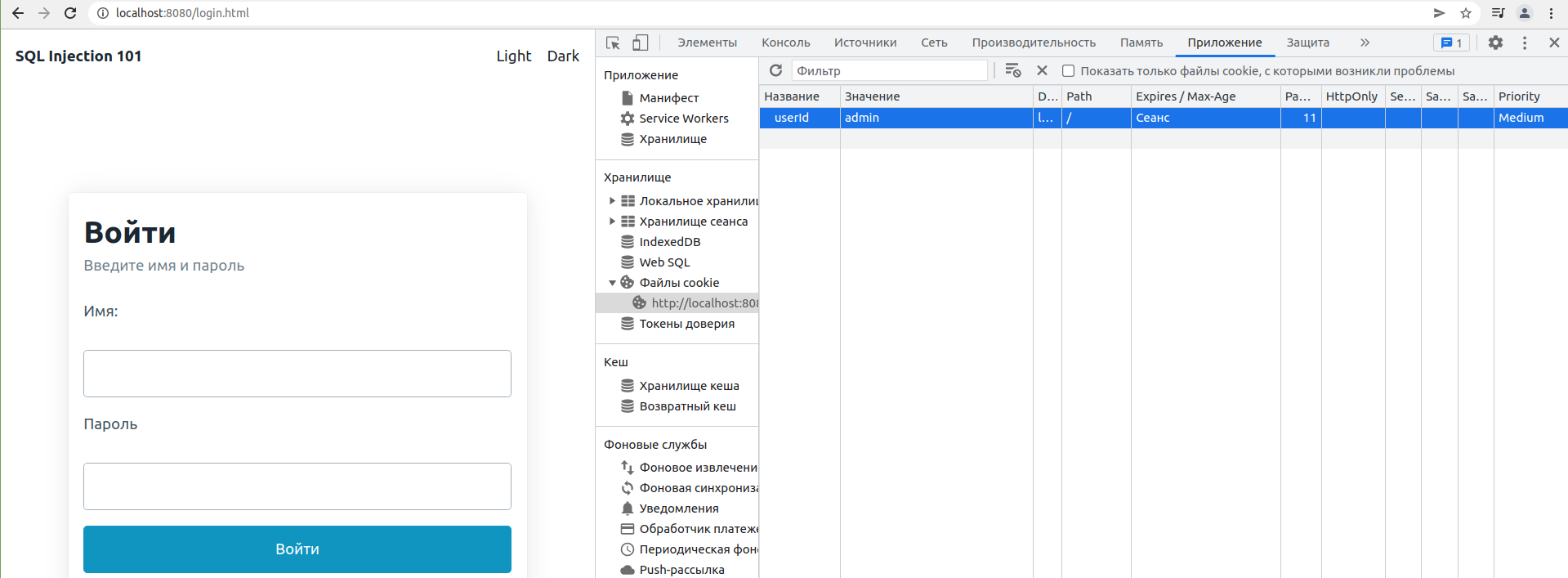


Рисунок 15 – Создание собственной сессионной куки для авторизации под пользователем admin

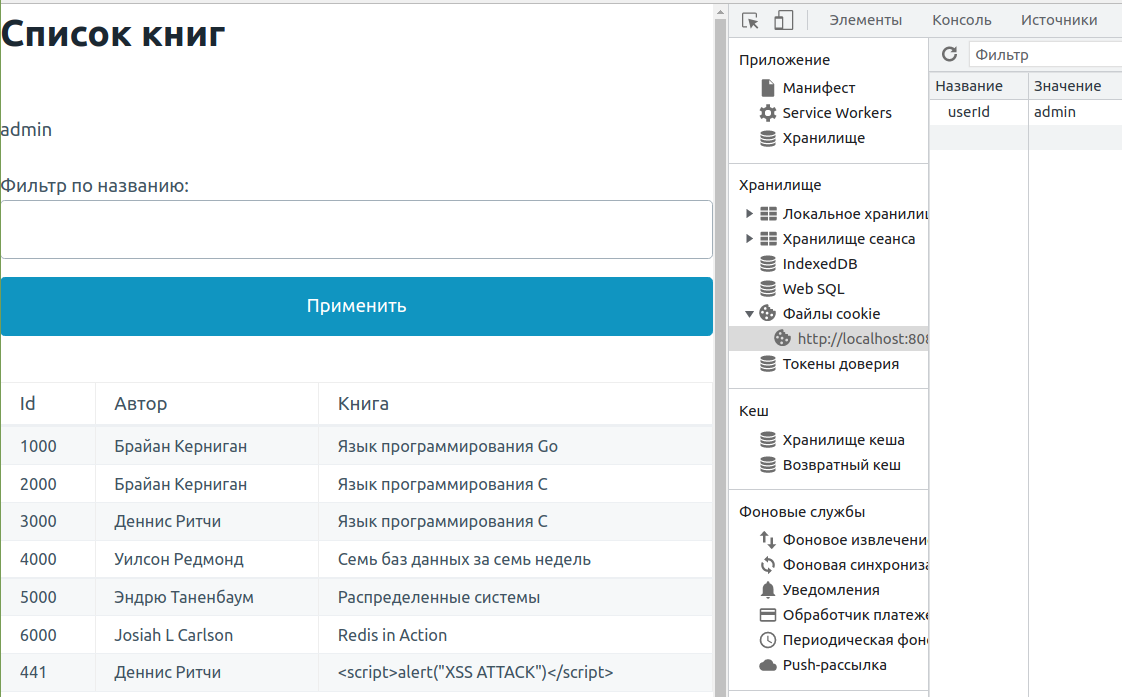


Рисунок 16 – Успешный доступ к списку книг от имени пользователя «admin» с помощью собственной созданной сессионной куки

Данная уязвимость легко осуществима, поскольку в качестве сессионной куки используется обычное имя пользователя. Настоящие куки должны иметь более сложный процесс формирования, например при объединении с каким-либо секретом с последующим хэшированием.

Таким образом, для нашего случая создадим сессионную куку SessionId, которая будет объединять в себе имя пользователя + секретную строку «secret12345», после этого полученная строка будет хэшироваться. Так как клиент сайта не будет знать нашего секрета и техники формирования хэша — ему не удастся подделать куку.

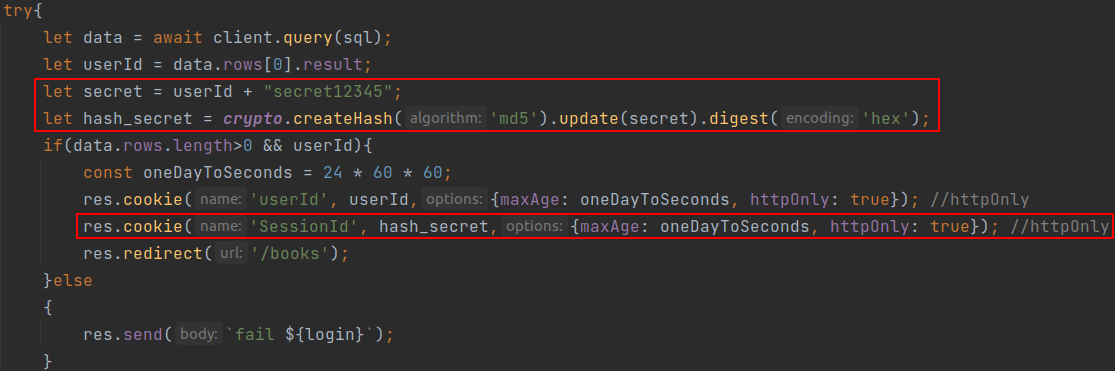


Рисунок 17 – Фрагмент кода с созданием секретной куки SessionId

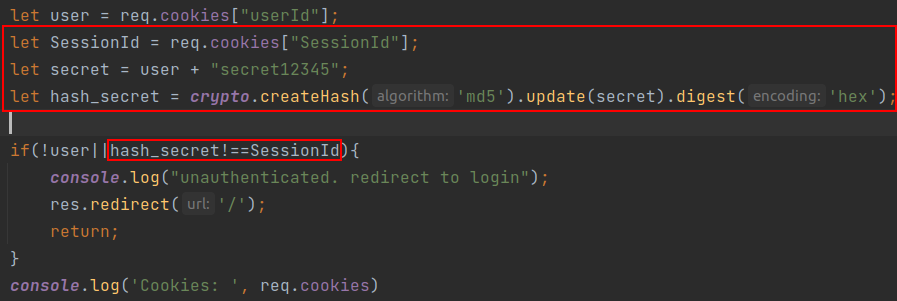


Рисунок 18 – Проверка секретной куки SessionId при обращении к странице с книгами

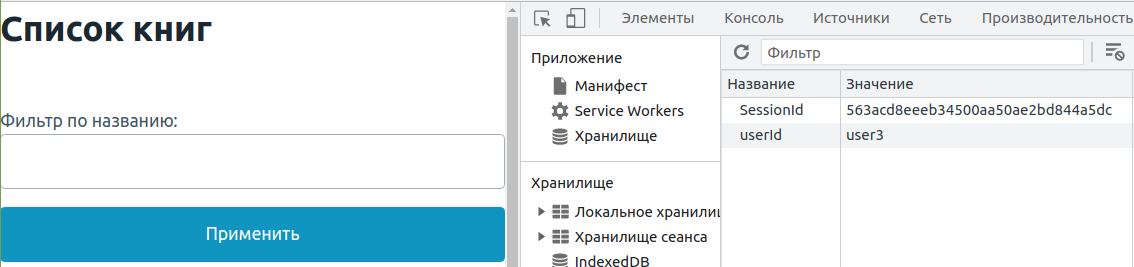


Рисунок 19 – Авторизация пользователем user3. Ему выдается секретная кука SessionId

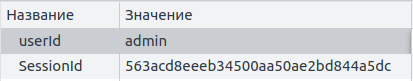


Рисунок 20 – Изменение куки userId с user3 на admin при неизменной секретной сессионной куки SessionId

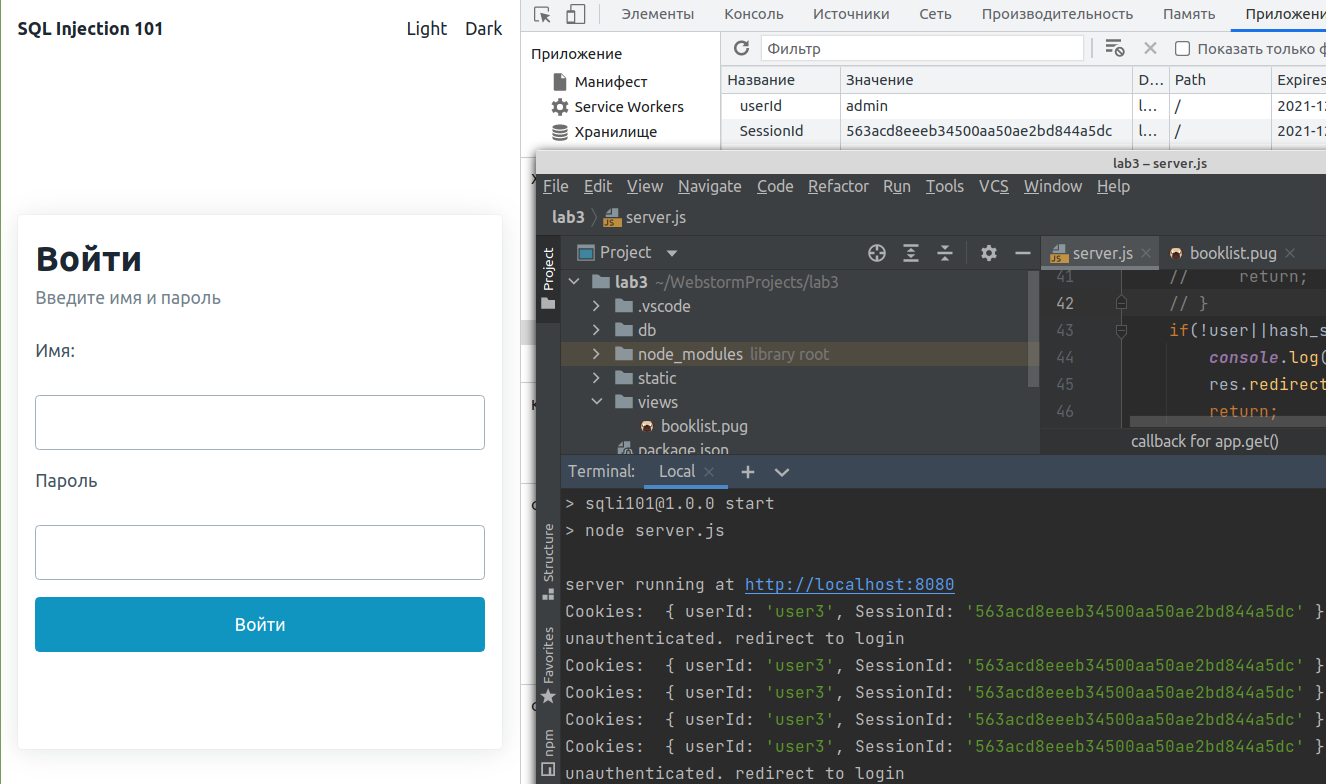


Рисунок 21 – При попытке подделки куки — происходит переброс на страницу. В реальности эту секретную сессионную куку SessionId можно еще лучше защитить, добавив в качестве энтропии значений — ее дату и время создания

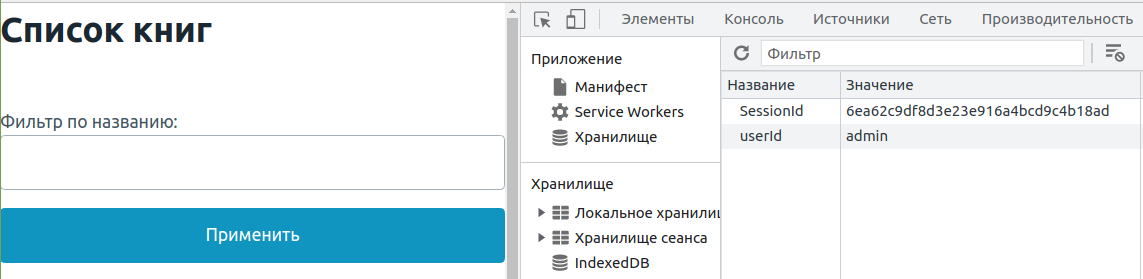


Рисунок 22 – При корректной авторизации от имени пользователя «admin» - видим, что ему соответствует совершенно друга секретная кука SessionId. Таким образом, мы успешно защитились от Session hijacking