Projektni zadatak - napadi

Razvoj Bezbednog Softvera

**Autor:** Ilija Obradović 2024/3090

Ovaj dokument sadrži korake koji su potrebni da bi se uspešno izveli traženi napadi na projektnom zadatku iz predmeta Razvoj Bezbednog Softvera.

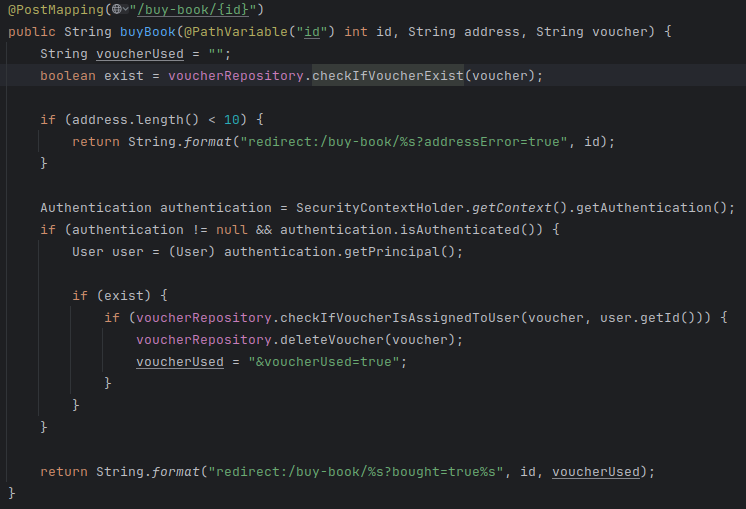
# SQLi

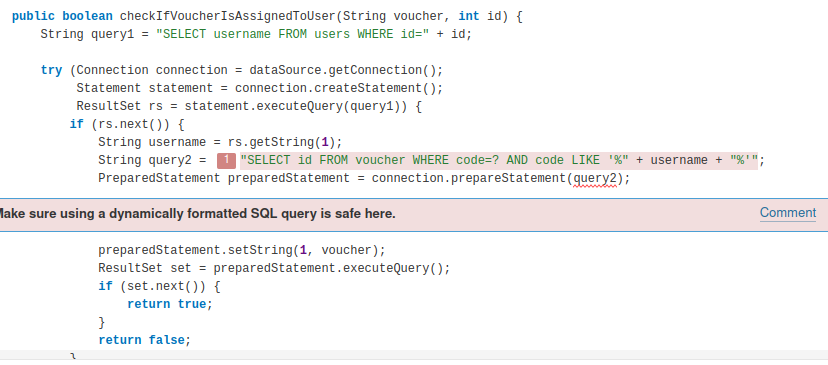
## Login stranica

Da bi izrada zadatka uopšte počela potrebno je prvo pristupiti web sajtu pomoću nekog naloga. Kako ovaj sajt nema opciju za registraciju korisnika jedino što je moguće jeste ulaz na postojeće naloge. Pošto se napad izvršava iz ugla napadača koji nema pristup informacijama o bazi podataka potrebno je izvršiti prvi SQLi na login stranici.

Ovime dolazimo na stranicu sa listom knjiga.

## Analiza koda kupovine knjige

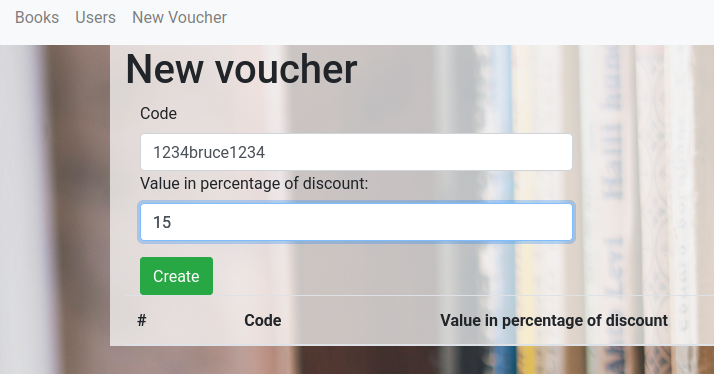
Stranica za kupovinu knjige zahteva unos broja kartice i adrese, i opcionog koda vaučera. Pogledom na kod, vidi se da se broj kartice i adresa ne koriste, već da je tu samo provera zadatog vaučera.  
  
Ono što je sigurno je da jedini SQL upiti koji se pri ovom zahtevu izvršavaju jesu oni koji se tiču vaučera. 

Iz ranije statičke analize se vidi ranjivost u funkciji **checkIfVoucherIsAssignedToUser**.

Ukoliko korisničko ime u sebi ima SQLi može se postići zahtev brisanja svih knjiga iz baze. Primer ovakvog malicioznog korisničkog imena je: “**something’; drop table book;**” **–**

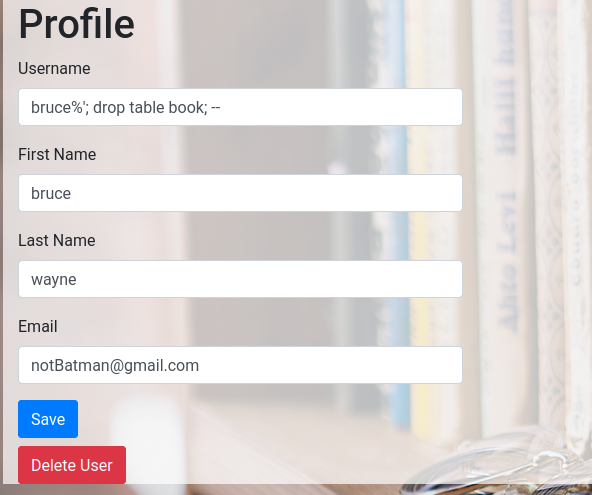
Međutim, ono što pravi problem jeste početak funkcije **buyBook**, sa pozivom **checkIfVoucherExist.** Da bi se uopšte došlo do ranjivog koda, mora se obezbediti postojanje bilo kakvog vaučera čija šifra je poznata.

## Novi vaučer

Pritiskom na navigaciono dugme **“New voucher”** odlazi se na stranicu za pravljenje novog vaučera. Da bi se ispunili zahtevi koda vaučera (**bruce -** nalog kom napadač ima pristup) potrebno je popuniti polja kao na primeru ispod. 

## Promena korisničkog imena

## Sada kada je obezbeđen vaučer potrebno je postaviti maliciozno koriničko ime. To se vrlo rako može uraditi pritiskom na dugme **“My Profile”** a zatim je potrebno promeniti username i pritisnuti **“Save”.**



## 

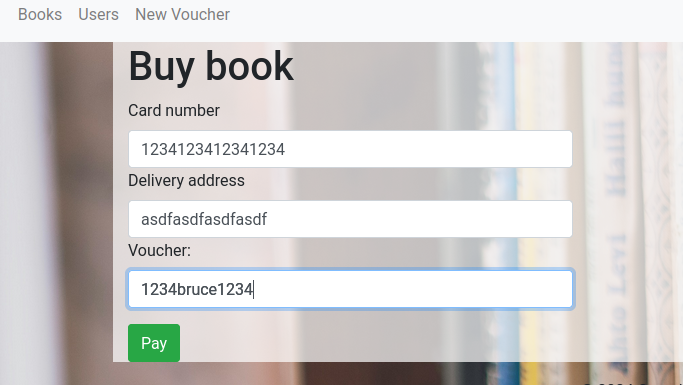
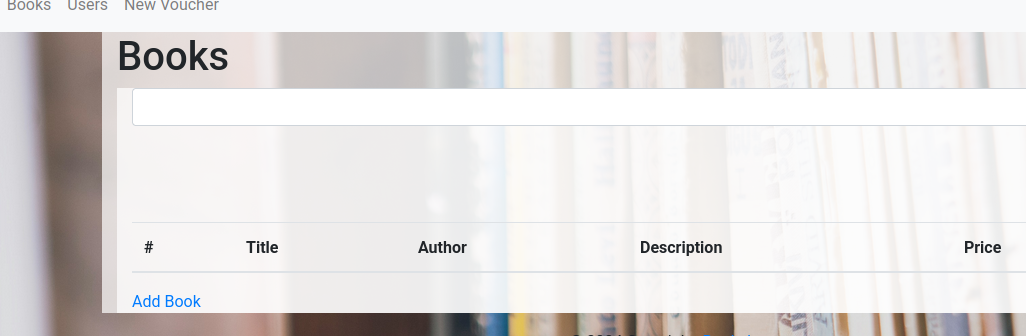
## Brisanje knjiga

Nakon celokupne pripreme potrebno je otići na stranicu za kupovinu, uneti potrebne podatke (**PAŽNJA:** unutar polja za šifru vaučera potrebno je uneti verziju sa starim imenom korisnika, kako je ona ostala zapamćena). Na kraju pritiskom na dugme **PAY**, dobija se željeni efekat.

**PAŽNJA:** Redosled akcija koje su urađene u napadu je bitan.

1. kreiranje vaučera
2. promena imena

Zato što je inject-ovan sql unutar imena korisnika, potrebno je da ono sadrži specijalne znakove poput razmaka ili crtica, što nije dozvoljeno u šifri vaučera (a ona mora sadržati ime korisnika u centru). Tako je prvo potrebno iskoristiti ime bez razmaka pa tek onda promeniti ime u ono koje sadrži maliciozni upit.

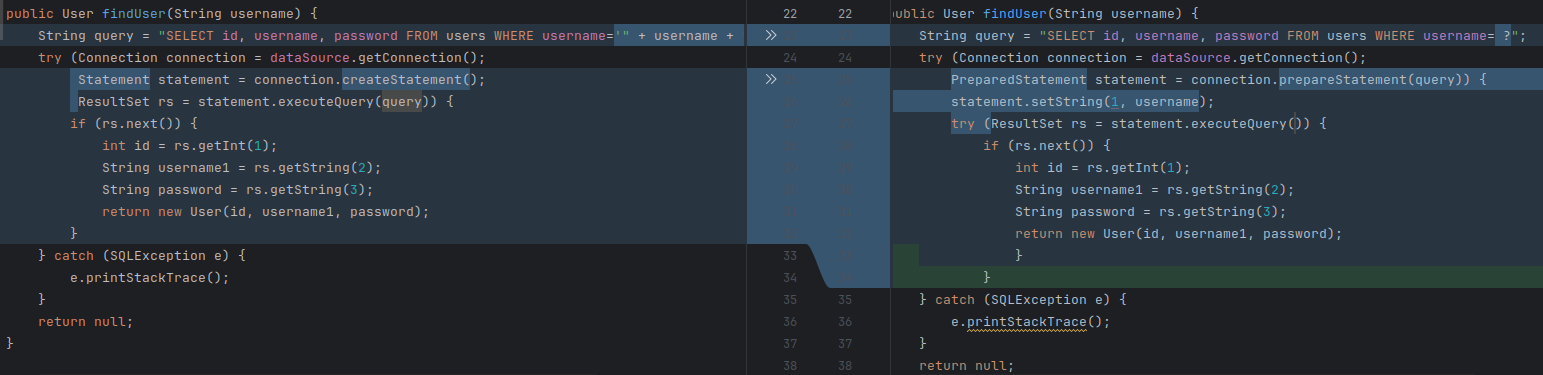


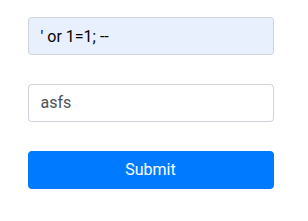
## Zaštita

Kao što je viđeno problem je u tome što se string koji se koristi za upit u SQL bazu pravi dinamički pomoću sabiranja stringova bez ikakve sanitizacije ili escape-ovanja. To se može lako rešiti zamenom takve konstrukcije **PreparedStatement.** Ispravljeni upiti:

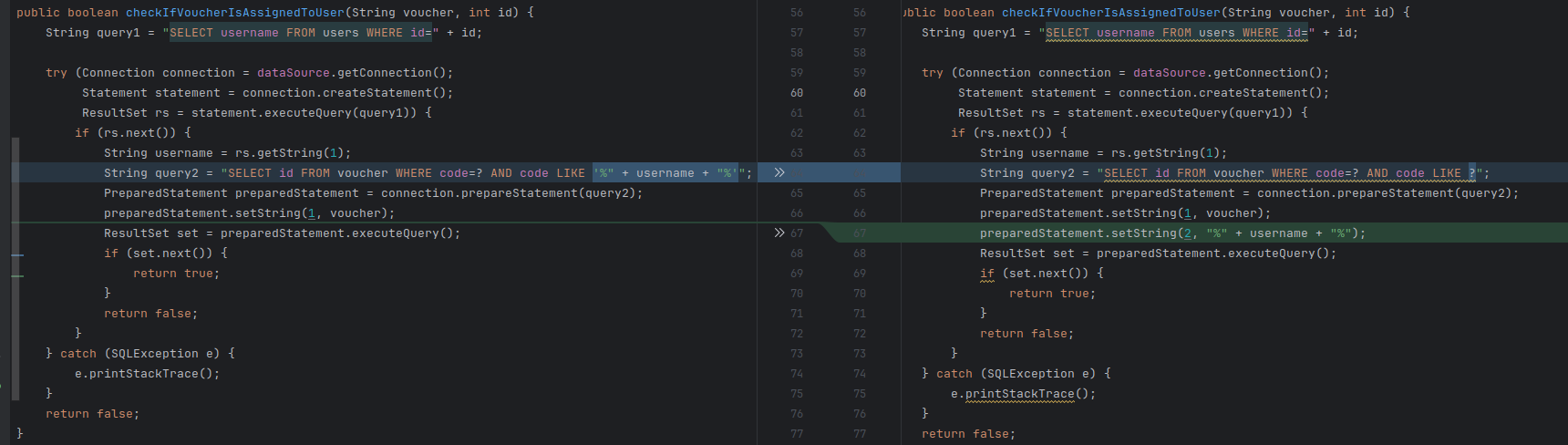
* + 1. Login SQLi fix

**UserRepository.java:**

**Rezultat:** nije moguće ulogovati se pomoću SQLi.



### Book buying SQLi fix

**VoucherRepository.java:**

**Rezultat:** I sa imenom u kom se nalazi SQLi, više nije moguće uraditi maliciozni SQL upit pri kupovini knjige.

# CSRF + XSS

Ovaj napad je lako izvesti kako ne postoji zastita os CSRF-a ili XSS-a na stranici sa komentarima.

## Analiza

Inspec-ovanjem HTML koda na stranici za dodavanje komentara može se videti da na se na klik dugmeta Create comment šalje određeni POST request:

## CSRF

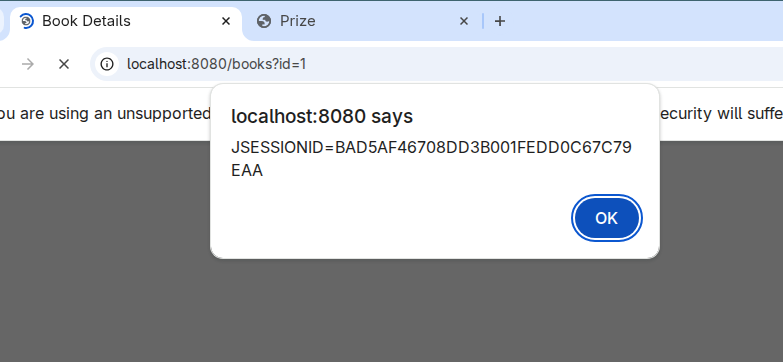
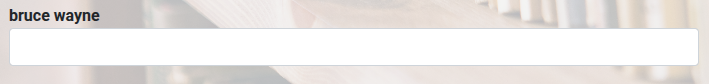
Kod naveden gore, može se kopirati u sajt napadača i to u funkciju **exploit()**, Promenljivih **bookID, i comment je potrebno zameniti** konkretnim vrednostima generisanim na napadačevom sajtu. **A putanju sa koje se poziva fetch, potrebno je zameniti sa celokupnom putanjom do ranjivog sajta.** Ovime se klikom na pehar u ime žrtve dodaje komentar sa XSS-om na knjigu 1 koja dodaje.

## Mamac

1. Prvo treba pokrenuti napadački web sajt, po uputstvu **“csrf-exploit/readme.txt”.** Može se videti da je napadački server na adresi **“http://localhost:3000”**
2. Zatim je potrebno navigirati na stranicu knjige 1
3. Pomoću XSS ostaviti komentar sa linkom koji izgleda primamljivo korisniku

## Zamka

1. Sada klikom na “here” korisnik biva poslat na napadački sajt.
2. Kada žrtva pritisne **“Click here!”** na napadačkom sajtu, iz njene perspektive neće se desiti ništa ali će sa njenim autentikacionim tokenom biti poslat zahtev na sajt za prodavnicu knjiga koji će dodati komentar sa skriptom koja korisnicima prikazuje njigov token na knjigu broj 1.



## Zaštita

* + 1. XSS fix

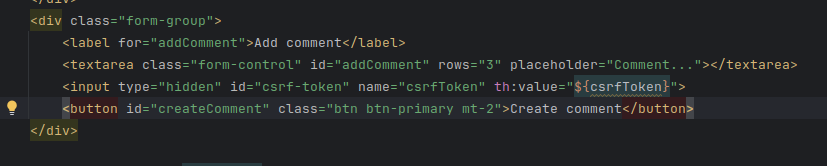
Potrebno je promeniti da se komentari ispisuju kao “th:text” a ne kao “th:utext”. Time se escape-uju problematični karakteri.

**templejt book.html (th:utext -> th:text)**:

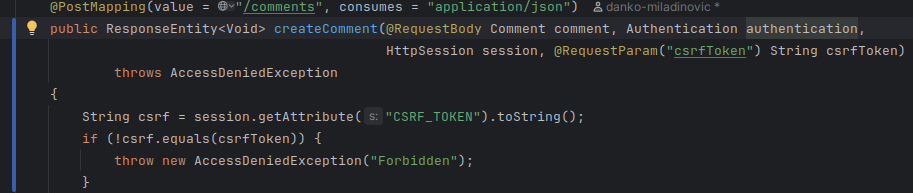
* + 1. CSRF fix

Dodavnje CSRF tokena, koji je kreiran pomoću **CsrfHttpSessionListener-a** i postavljen u podatke o Http Sesiji (HttpSession - session), u formu za pravljenje komentara (kao skriveno polje).

## 



Dodavanje provere csrf tokena prilikom POST request-a za dodavanje komentara.



Kada se klikne na “Click here” na napadačkom sajtu sada se dobija greška da tokena nema. Pošto se token kreira pri svakoj sesiji i čuva se na serveru, napadač ne može znati njega unapred već mora nekako doći do njega ukoliko želi da izvede ranije prikazan napad:

