Istorija izmena

Verzija	Datum	Izmenio/la	Komentar
1.0	28.2.2024.	Uroš Dragojević	Kreiran izveštaj

Uvod

Ovaj izveštaj se bavi ranjivostima pronađenim u dole opisanoj veb aplikaciji.

O veb aplikaciji

RealBookStore je veb aplikacija koja pruža mogućnosti pretrage, ocenjivanja i komentarisanja knjiga.

Aplikacija RealBookStore omogućava sledeće:

- Pregled i pretragu knjiga.
- Dodavanje nove knjige.
- Detaljan pregleda knjige kao i komentarisanje i ocenjivanje knjige.
- Pregled korisnika aplikacije.
- Detaljan pregled podataka korisnika.

Kratak pregled rezultata testiranja

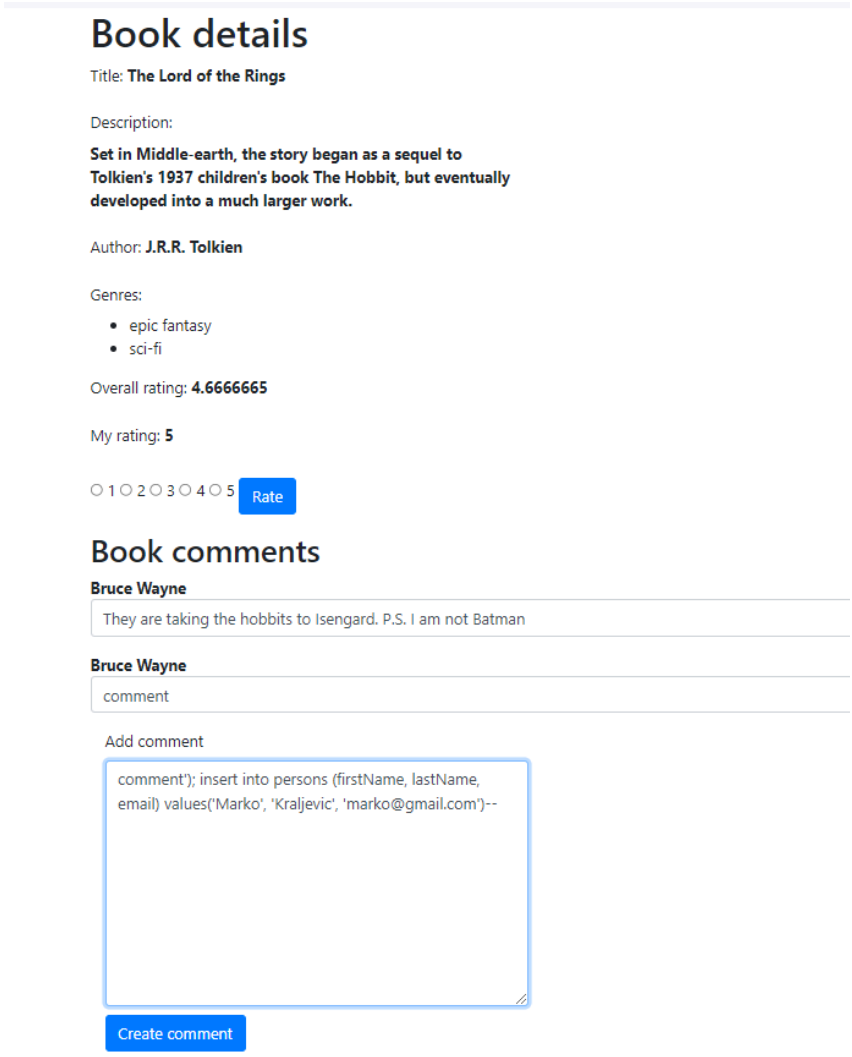
Ovde idu kratko opisani rezultati testiranja: pronađene ranjivosti i nivo opasnosti.

Nivo opasnosti	*Broj ranjivosti*
Low	*3*
Medium	*2*
High	*1*

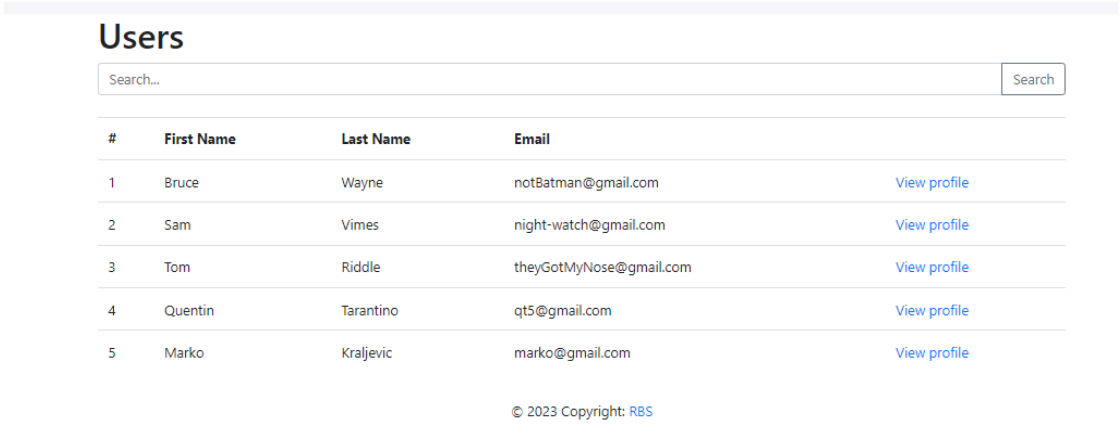
SQL injection

Napad: Ubacivanje novog usera u tabelu “persons” (SQL injection)

Metod napada:



Na stranici stranici za svaku knjigu postoji ranjivost u delu za komentare gde je moguce uneti novu personu. Nakon napada moze se videti da je dodat novi user Marko Kraljevic. Na sledecoj strani se moze videti rezultat napada.



Predlog odbrane:

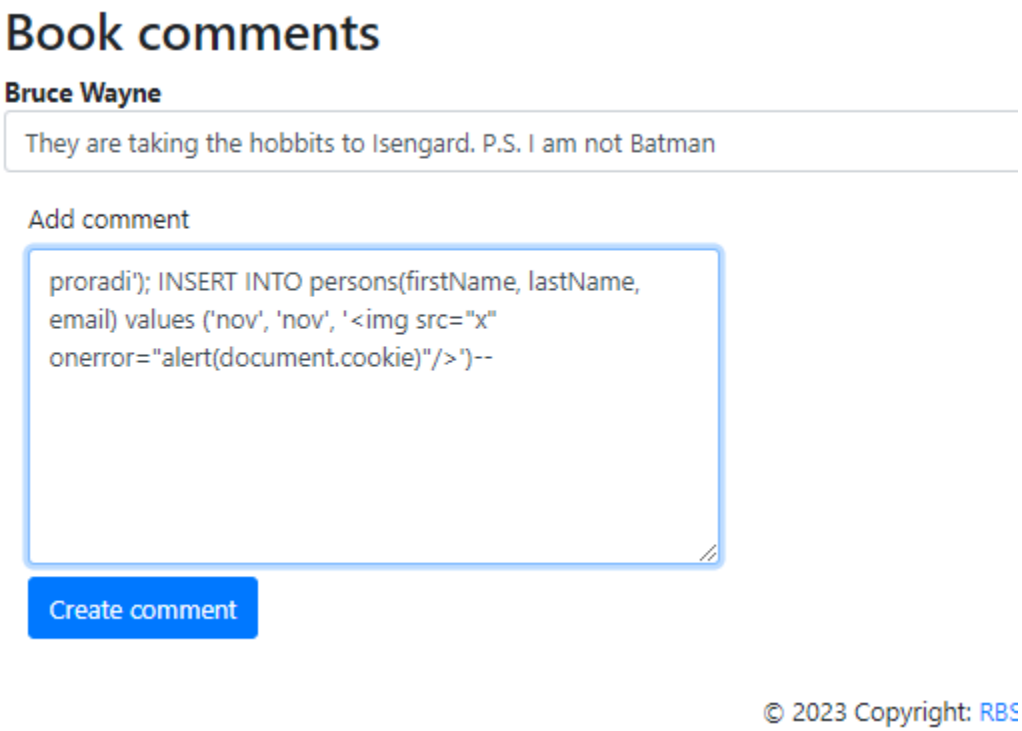
U klasi CommentRepository potrebno je Statement zameniti sa PreparedStatement kako ova vrsta napada ne bi bila moguca.

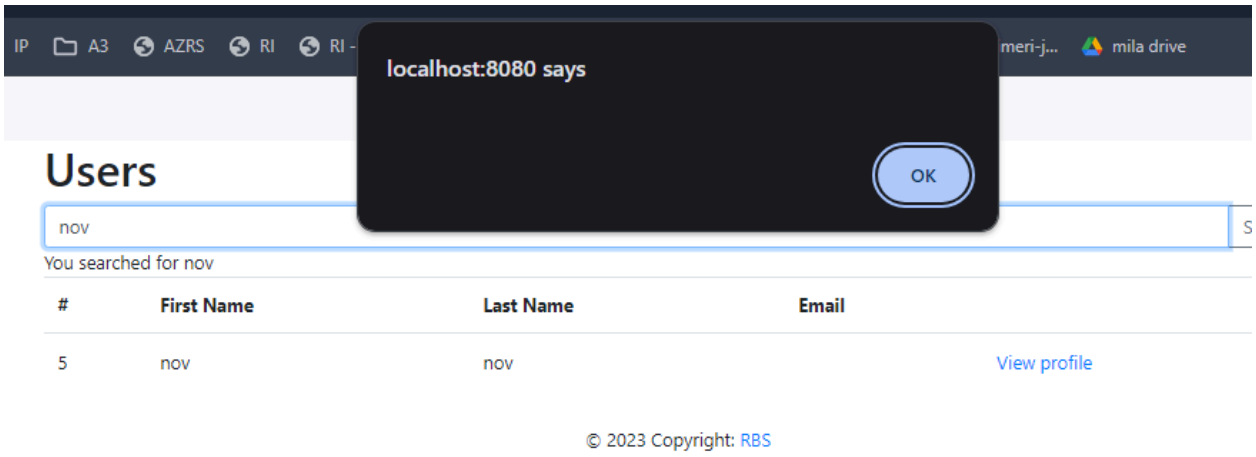
```
27 29
28 30     public void create(Comment comment) {
29    -        String query = "insert into comments(bookId, userId, comment) values (" + comment.getBookId() + ", " + comment.getUserId() + ", '" + comment.getComment() + "')"
   31 +        String query = "insert into comments(bookId, userId, comment) values (?, ?, ?)";
30 32
31 33         try (Connection connection = dataSource.getConnection();
32    -              Statement statement = connection.createStatement();
   34 +              PreparedStatement statement = connection.prepareStatement(query);
33 35         ) {
34    -            statement.execute(query);
   36 +            statement.setInt(1, comment.getBookId());
   37 +            statement.setInt(2, comment.getUserId());
   38 +            statement.setString(3, comment.getComment());
   39 +            statement.executeUpdate();
35 40         } catch (SQLException e) {
36 41             e.printStackTrace();
37 42         }
```

Cross-site scripting

Napad: Ubacivanje novog usera u tabelu “persons”

Metod napada:





Ubacujemo maliciozni js kod koji za zadatak ima da ispise cookie na konzolu

Predlog odbrane:

Potrebno je innerHtml zameni sa textContent kako nas brauzer ne bi interpretirao maliciozni kod vec ga tretirao samo kao tekst. Takodje radimo sanitizaciju unosa tako sto th:utext menjamo sa th:text, kako bi sprecili karaktere koji mogu da se iskoriste zlonamerno (<, >...)

```
52                          const tableContent = document.getElementById("tableContent");
 -                          tableContent.innerHTML = '';
53 +                        tableContent.textContent = '';
54
55                          persons.forEach(function(person) {
56                              const tableRowElement = document.createElement("tr");
57                              let tdElement = document.createElement("td");
 -                              tdElement.innerHTML = person.id;
58 +                            tdElement.textContent = person.id;
59                              tableRowElement.appendChild(tdElement);
60                              tdElement = document.createElement("td");
 -                              tdElement.innerHTML = person.firstName;
61 +                            tdElement.textContent = person.firstName;
62                              tableRowElement.appendChild(tdElement);
63                              tdElement = document.createElement("td");
 -                              tdElement.innerHTML = person.lastName;
64 +                            tdElement.textContent = person.lastName;
65                              tableRowElement.appendChild(tdElement);
66                              tdElement = document.createElement("td");
 -                              tdElement.innerHTML = person.email;
67 +                            tdElement.textContent = person.email;
68                              tableRowElement.appendChild(tdElement);
69                              tdElement = document.createElement("td");
 -                              tdElement.innerHTML = '<a href="/persons/' + person.id + '">View profile</a>';
70 +                            tdElement.textContent = '<a href="/persons/' + person.id + '">View profile</a>';
71                              tableRowElement.appendChild(tdElement);
72
73                              tableContent.appendChild(tableRowElement);
74                          });
75
76                          document.getElementById('searchContainer').className = '';
 -                          document.getElementById('searchTerm').innerHTML = searchTerm;
77 +                        document.getElementById('searchTerm').textContent = searchTerm;
78                      });
79              };
```

Cross-site request forgery

Napad: Preimenovanje user-a sa id 1

Metod napada:

```
function exploit() {
    let attackData = new FormData();
    attackData.append('id', 1);
    attackData.append('firstName', 'Batman');
    attackData.append('lastName', 'Dark Knight');

    fetch('http://localhost:8080/update-person', {method : 'POST',
        body : attackData, credentials : 'include'});
}
```

Nakon pokretanja skripte, dok smo ulogovani na sajt RealBookStore mozemo, korisnik otvara maliciozni sajt klicke na pehar i zbog sacuvanih kredencijala uspevamo da menjamo podatke o korisniku.

Rezultat napada:

#	First Name	Last Name	Email
1	Batman	Dark Knight	notBatman@gmail.com

Predlog odbrane:

Potrebno je kreirati token za autentikaciju sesije korisnika. To radimo u klasi CsrfHttpSessionListner pomocu SecureRandom klase. Sada server svaki put proveri da li token odgovara tokenu korisnika za tu sesiju.

Token se prosledjuje kao nevidljivo polje unutar forme

```
27              </div>
28              <input type="hidden" name="id" class="form-control" id="id" th:va
29 +            <input type="hidden" name="csrfToken" th:value="${CSRF_TOKEN}">
30              <button type="submit" class="btn btn-primary">Save</button>
31          </form>
```

U metodi person zaduzenoj za promenu detalja korisnika citamo token iz sesije i upisujemo ga u model. Dok u metodi updatePerson vrsimo poveru da li je token iz forme isti kao token iz sesije.

```
32 35
33 36       @GetMapping("/persons/{id}")
34    -     public String person(@PathVariable int id, Model model) {
   37 +     public String person(@PathVariable int id, Model model, HttpSession session) {
   38 +         model.addAttribute("CSRF_TOKEN", session.getAttribute("CSRF_TOKEN"));
35 39           model.addAttribute("person", personRepository.get("" + id));
36 40           return "person";
37 41       }
       @@ -52,8 +56,13 @@ public class PersonsController {
52 56       }
53 57
54 58       @PostMapping("/update-person")
55    -     public String updatePerson(Person person) {
56    -         personRepository.update(person);
   59 +     public String updatePerson(Person person, HttpSession session, @RequestParam("csrfToken") String csrfToken) throws AccessDeniedException
   60 +         String csrf = session.getAttribute("CSRF_TOKEN").toString();
   61 +         if (!csrf.equals(csrfToken)) {
   62 +             throw new AccessDeniedException("Forbidden");
   63 +         }
   64 +
   65 +        personRepository.update(person);
57 66           return "redirect:/persons/" + person.getId();
58 67       }
59 68
```

Authorization

Dodavanje odgovarajucih rola, permisija i njihove povezanosti u bazu.

```
insert into roles(id, name)
values (1, 'ADMIN'),
       (2, 'MANAGER'),
       (3, 'REVIEWER');

insert into permissions(id, name)
values (1, 'ADD_COMMENT'),
       (2, 'VIEW_BOOKS_LIST'),
       (3, 'CREATE_BOOK'),
       (4, 'VIEW_PERSONS_LIST'),
       (5, 'VIEW_PERSON'),
       (6, 'UPDATE_PERSON'),
       (7, 'VIEW_MY_PROFILE'),
       (8, 'RATE_BOOK');

insert into user_to_roles(userId, roleId)
values (1, 3),
       (2, 3),
       (3, 1),
       (4, 2);

insert into role_to_permissions(roleId, permissionId)
values (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8),
       (2, 1), (2, 2), (2, 3), (2, 4), (2, 6),(2, 7), (2, 8),
       (3, 1), (3, 2), (3, 6), (3, 7), (3, 8);
```

Identicno kao u vezbama dodata autorizacija. Kako bi za role Manager i Reviewer bilo moguce da menjaju samo svoj profil, dodata je autentikacija preko csrf tokena.

Logging and auditing

Logging dodat u slucaju svakog exception-a, dok auditing dodat kada se desavaju promene nad bazom (create, update, delete) i u slucaju validacije podataka za korisnika.