Stručni kurs Razvoj bezbednog softvera

Izveštaj

Pronađene ranjivosti u projektu “ RealBookStore ”

Dusan Mircic, 325/2019

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:.....4
Cross-site scripting.....5
Napad: Ubacivanje novog usera u tabelu “persons”.....5
Metod napada:.....5
Predlog odbrane:.....5
Zaključak.....6

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*

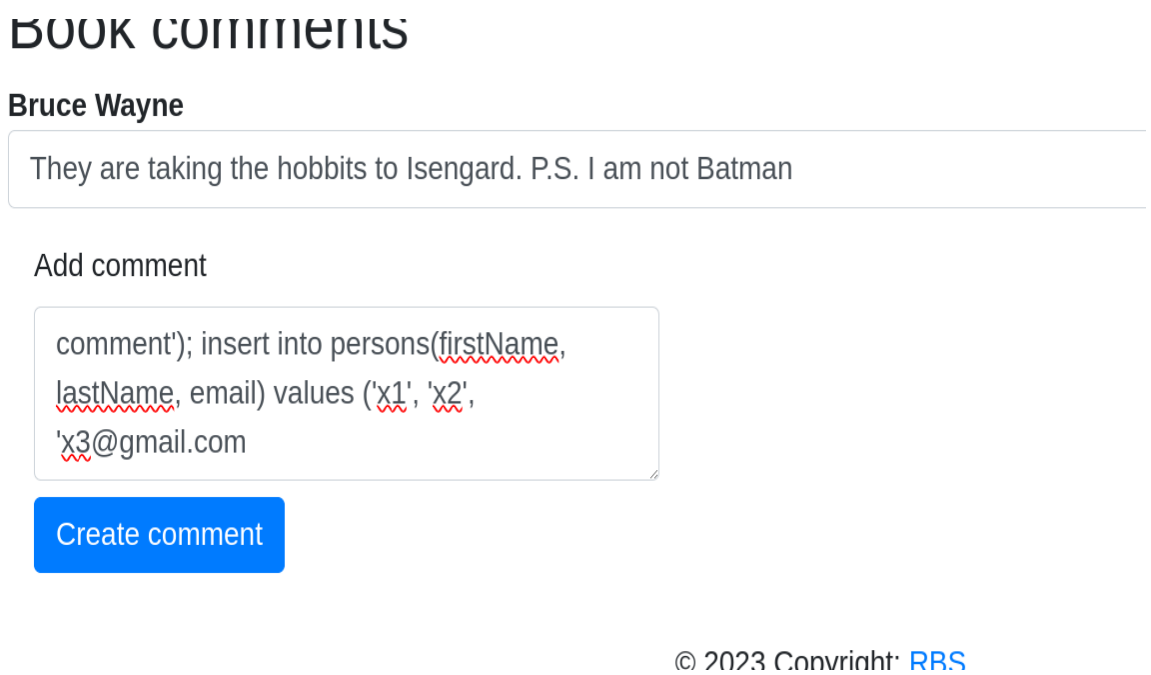
SQL injection

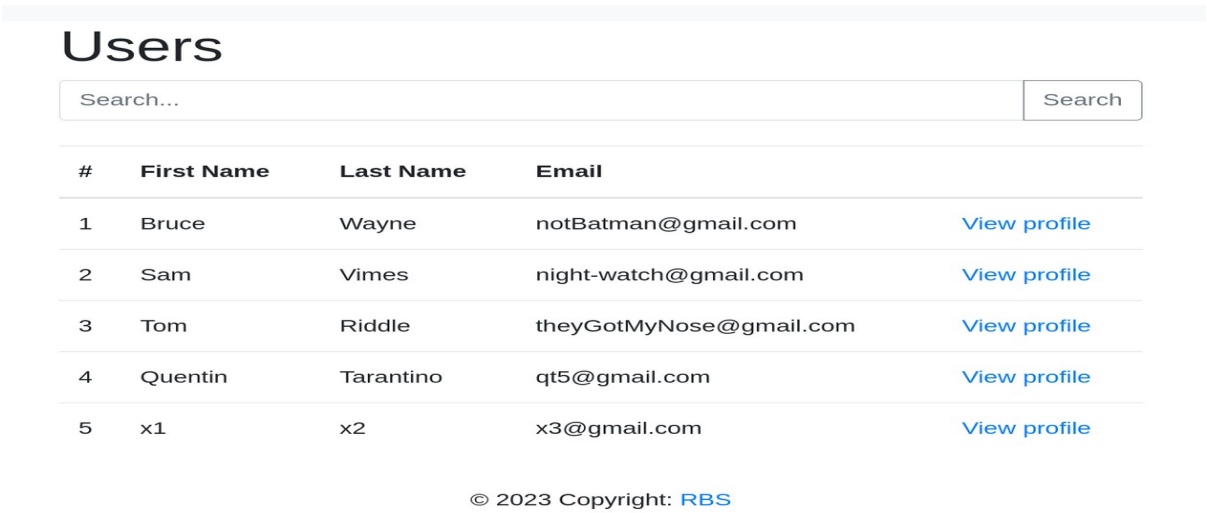
Napad: Ubacivanje novog usera u tabelu “persons” (SQL injection)

Metod napada:

Na stranici Persons aplikacije, uneti sledeći kod u input polje “First Name”:

> komentar'); insert into persons(firstName, lastName, email) values('intruderFirst', 'intruderLast', 'intruder@gmail.com





Odbrana:

Koristimo klasu PreparedStatement umesto klase Statement.

Cross-site scripting

Napad: Ubacivanje novog usera u tabelu “persons”

Metod napada:

Ubacujemo novog korisnika u tabelu persons koji ce kao polje da ima malicioznu skriptu.

komentar’); insert into persons(firstName, lastName, email) values ('A','B','

Predlog odbrane:

U persons.html menjamo innerHTML u textContent kako bi tretirali polja koja se unose kao obican tekst.

Cross-site request forgery

Napad: Ubacivanje novog usera u tabelu “persons”

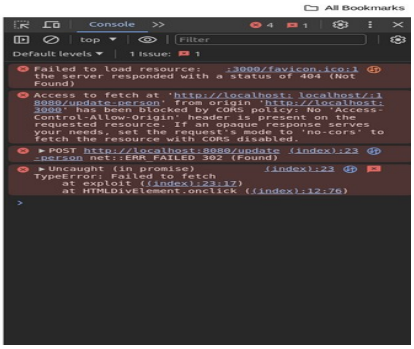
Metod napada:

Prevara korisnika da klikne dugme na stranici što će u pozadini poslati zahtev serveru za promenu podatka u bazi.

```
<script>
    function exploit() {
        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>
```

Predlog odbrane:

Dodajemo token, server će pri svakom zahtevu da proverava da li primljeni token odgovara onom uskladištenom u podacima sesije korisnika.



```
@PostMapping("/update-person")  no usages  Ana +1
public String updatePerson(Person person, HttpSession session, @RequestParam("csrfToken") String csrfToken) throws AccessDeniedException {
    String csrf = session.getAttribute( s: "CSRF_TOKEN").toString();
    if (!csrf.equals(csrfToken)) {
        throw new AccessDeniedException("Forbidden");
    }
    personRepository.update(person);
    return "redirect:/persons/" + person.getId();
}
```

Zaključak

Implementirane su popravke za SQL Injection, XSS I CSRF.