|  |
| --- |
| Stručni kurs Razvoj bezbednog softvera |
| Izveštaj |
| Pronađene ranjivosti u projektu “RealBookStore” |

|  |
| --- |
| Vladeta Vujačić  10.5.2024 |

# Istorija izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Verzija** | **Datum** | **Izmenio/la** | **Komentar** |
| **1.0** | 28.2.2024. | Uroš Dragojević | Kreiran izveštaj |
| **2.0** | 10.5.2024. | Vladeta Vujačić | Kompletiran izveštaj |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

[Istorija izmena 1](#_Toc160050508)

[Uvod 3](#_Toc160050509)

[O veb aplikaciji 3](#_Toc160050510)

[Kratak pregled rezultata testiranja 3](#_Toc160050511)

[SQL injection 4](#_Toc160050512)

[Napad: Ubacivanje novog usera u tabelu “persons” (SQL injection) 4](#_Toc160050513)

[Metod napada: 4](#_Toc160050514)

[Predlog odbrane: 4](#_Toc160050515)

[Cross-site scripting 5](#_Toc160050516)

[Napad: Ubacivanje novog usera u tabelu “persons” 5](#_Toc160050517)

[Metod napada: 5](#_Toc160050518)

[Predlog odbrane: 5](#_Toc160050519)

[Zaključak 6](#_Toc160050520)

# 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

*Napomena: svi dokumentovani napadi i rezultati izmena nalaze se u podfolderu report u vidu jednog markdown fajla.*

|  |  |
| --- | --- |
| ***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 za ocenjivanje pojedinačne knjige, uneti sledeći kod kao komentar:

‘); insert into persons(firstName, lastName, email) values(‘Vladeta’, ‘Vujacic’, ‘bilo sta’); --

## Predlog odbrane:

Kako bismo popravili SQLi ranjivost, potrebno je da umesto običnog Statement-a koristimo PreparedStatement koji nam omogućava da se potencijalno maliciozan kod moze jedino interpretirati kao komentar a ne i kao deo naredbe.

Stari deo koda:

String query = "insert into comments(bookId, userId, comment) values (" + comment.getBookId() + ", " + comment.getUserId() + ", '" + comment.getComment() + "')";

try (Connection connection = dataSource.getConnection();

Statement statement = connection.createStatement();

){

statement.execute(query);

} catch (SQLException e) {

e.printStackTrace();

}

Novi deo koda:

String query = "insert into comments(bookId, userId, comment) values (?,?,?)";

try (Connection connection = dataSource.getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(query);

){

preparedStatement.setInt(1,comment.getBookId());

preparedStatement.setInt(2,comment.getUserId());

preparedStatement.setString(3,comment.getComment());

preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

# Cross-site scripting

## Napad: Ubacivanje novog user-a u tabelu “persons” sa email-om koji je maliciozan kod koji će nam omogućiti da dohvatimo cookie svaki put kada se poseti stranica koja izlistava osobe

### Metod napada:

‘); insert into persons(firstName, lastName, email) values(‘Vladeta’, ‘Vujacic’, ‘<img src=”x” onerror=”console.log(document.cookie);”/>’); --

## Predlog odbrane:

Kako bismo onemogućili da neko kao komentar unese nešto što u sebi ima kod koji eksploatiše cross-site scripting ranjivost, moramo u persons.html fajlu na mestima gde se koristi innerHTML zameniti sa textContent što onemogućava da pretraživač interpretira dohvaćen tekst kao JavaScript kod. Takođe, u book.html fajlu treba zameniti th:utext sa th:text kako bismo onemogućili interpretiranje sadržaja kao kod a ne kao tekst.

# Cross-site request forgering

## Napad: Korišćenje trenutno uspostavljene sesije korisnika kako bismo izvršili promenu imena osobe

### Metod napada

Kako bismo izvršili CSRF napad, potrebno je da u /csrf-exploit/index.html fajlu implementiramo funkciju exploit koja će imati sledeći kod:

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', { mode: 'no-cors', method : 'POST', body: formData, credentials: 'include'});

}

Sada, svaki put kada žrtva klikne na trofej, uputiće se zahtev na endpoint update-person od strane trenutno ulogovanog korisnika.

## Predlog odbrane:

Kako bismo onemogućili da neko eksploatiše CSRF ranjivost, potrebno je da implementiramo generisanje, razmenu i proveru ispravnosti CSRF tokena na update-person endpoint-u.

# Zaključak

Izmenama koje su gore navedene onemogućeno je izvršavanje SQLi i XSS napada, kao i CSRF napada na update-person endpoint.

Ono što je još rađeno na kursu, a što bi moglo da se implementira kako bismo poboljšali sigurnost naše aplikacije jeste dvofaktorska Autentifikacija i Autorizacija, kao i Logging i Auditing.