

Razvoj bezbednog softvera

Jovan Vukić

Januar 19, 2024.

3. SQL injection i Cross-site scripting

3.1. Realizacija napada

Moguće je na [stranici pojedinačnog poklona](#), preko forme za ostavljanje komentara izvršiti *SQL injection* napad. Deo koda koji obrađuje dodavanje komentara je sledeći:

```
Java
public void create(Comment comment) {
    String query = "insert into comments(giftId, userId, comment) values (" +
        + comment.getGiftId() +
        + ", " + comment.getUserId() +
        + ", '" + comment.getComment() + "' )";
    ...
}
```

Komentar koji unesemo na stranici će biti sadržan u upitu iznad, na sledeći način:

```
Java
"..., '<komentar>')"
```

Zaključimo da treba umesto „<komentar>“ uneti sledeći sadržaj:

```
Java
"..., 'neki tekst'); <zlonamerni_kod> --'")
```

Na kraju, umesto „<zlonamerni_kod>“ treba uneti SQL upit kojim se dodaje novi korisnik u tabelu *Persons* u bazi podataka:

```
Unset
INSERT INTO Persons (firstName, lastName, email) VALUES
('marco', 'polo', '')
```

Razvoj bezbednog softvera

Primetimo da je umesto „email“ atributa prosleđen kod koji će izbaciti obaveštenje korisniku sa sadržajem kolačića sesije. Radi se o XSS napadu. Obaveštenje će se prikazati pri pretrazi korisnika na [stranici persons.html](#).

Na slikama ispod vidimo spisak korisnika u bazi i komentar sa zlonamernim kodom.

#	First Name	Last Name	Email	
1	bruce	wayne	notBatman@gmail.com	View profile
2	Peter	Petigrew	oneFingernailFewerToClean@gmail.com	View profile
3	Tom	Riddle	theyGotMyNose@gmail.com	View profile
4	Santa	Clause	st@northPole.com	View profile

© 2023 Copyright: Christmas Gift Shop

Slika 3.1.1. Trenutni spisak korisnika u bazi podataka

Name: Christmas tree decoration

Description: Decoration for a Christmas tree

Price: 500.01

Tags

- small
- wholesome

Rating: 4.6666666666666667

My rating: 5

0 1 0 2 0 3 0 4 0 5 Rate

Buy gift

Gift comments

bruce wayne

Very nice.

Add comment

```
nekij tekst'); INSERT INTO Persons (firstName, lastName, email) VALUES ('marco', 'polo', '')
```

Create comment

Slika 3.1.2. Unošenje komentara sa zlonamernim kodom

Razvoj bezbednog softvera

Na stranici za unos komentara ostao je komentar ([slika 3.1.3](#)), ali je izvršen i maliciozni kod, jer je na stranici `persons.html` dodat novi korisnik ([slika 3.1.4](#)).

Gift details

Name: **Christmas tree decoration**

Description: **Decoration for a Christmas tree**

Price: **500.01**

Tags

small

wholesome

Rating: **4.6666666666666667**

My rating: **5**

1 2 3 4 5 [Rate](#)

[Buy gift](#)

Gift comments

bruce wayne

Very nice.

bruce wayne

neki tekst

Add comment

Comment...

[Create comment](#)

Slika 3.1.3. Novi komentar sa tekstualnim sadržajem „neki tekst“

ft Shop

Gifts Users My Profile Register second fa

Users

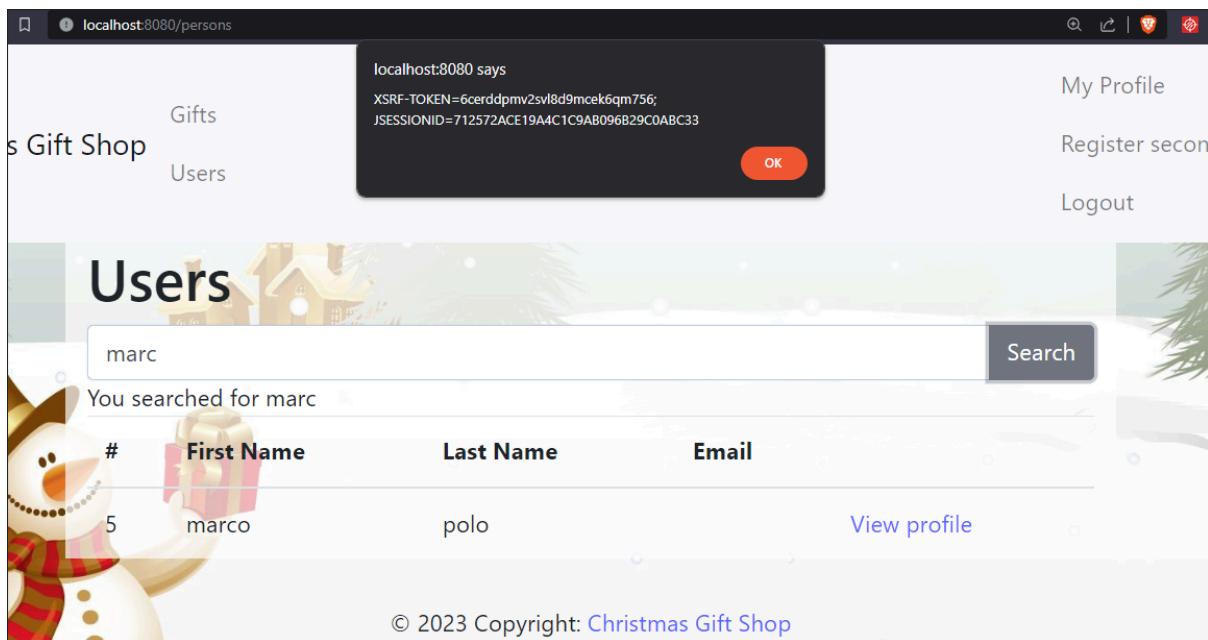
Search... [Search](#)

#	First Name	Last Name	Email	
1	bruce	wayne	notBatman@gmail.com	View profile
2	Peter	Petigrew	oneFingernailFewerToClean@gmail.com	View profile
3	Tom	Riddle	theyGotMyNose@gmail.com	View profile
4	Santa	Clause	st@northPole.com	View profile
5	marco	polo		View profile

© 2023 Copyright: Christmas Gift Shop

Slika 3.1.4. Dodat novi korisnik

Jednostavnom pretragom korisnika aktiviraće se skripta sačuvana u atributu novounetog korisnika ([slika 3.1.5](#)).



Slika 3.1.5. Ispis vrednosti kolačića sesije

3.2. Realizacija odbrane

Da bismo sprečili prethodne *SQLi* i *XSS* napada potrebno je i dovoljno:

1. Koristiti parametrizovani upit u kodu u klasi *CommentRepository*. Tada se unošenjem komentara ([slika 3.2.1](#)) neće uneti novi korisnik ([slika 3.2.2](#)).
2. Sanitizovati polje za unos komentara na stranici *gift.html* da bi se sprečili nezavisni *XSS* napadi, uvođenjem atributa *th:text*.

Ispod je kod metode *create* u fajlu *CommentRepository* zaštićen od *SQLi* napada.

```
Java
public void create(Comment comment) {
    String query = "insert into comments(giftId, userId, comment) values (?, ?, ?)";
    try (Connection connection = dataSource.getConnection();
         PreparedStatement statement = connection.prepareStatement(query)) {
        statement.setInt(1, comment.getGiftId());
        statement.setInt(2, comment.getUserId());
        statement.setString(3, comment.getComment());
        statement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Da bismo se zaštitali i od *XSS* napada, u *gift.html* dodali smo sanitizaciju.

```
JavaScript
<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>
```

Sada na slikama ispod vidimo da ne uspevaju ni *SQLi* niti *XSS* napadi.



Slika 3.2.1. Maliciozni kod za SQLi i XSS napade tumači se kao tekstualni komentar

#	First Name	Last Name	Email	Action
1	bruce	wayne	notBatman@gmail.com	View profile
2	Peter	Petigrew	oneFingernailFewerToClean@gmail.com	View profile
3	Tom	Riddle	theyGotMyNose@gmail.com	View profile
4	Santa	Clause	st@northPole.com	View profile

© 2023 Copyright: [Christmas Gift Shop](#)

Slika 3.2.2. Novi korisnik nije dodat

Dodatno je traženo da se na stranici `persons.html` izvrši sanitizacija od XSS napada, odnosno da se `innerHTML` zameni adekvatno sa `textContent`.

```
Java
let search = function () {
    const searchTerm = document.getElementById("searchInput").value;

    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;
});
};
```



Slika 3.2.3. Neuspešan XSS napad na stranicu persons.html

4. Cross-site request forgery

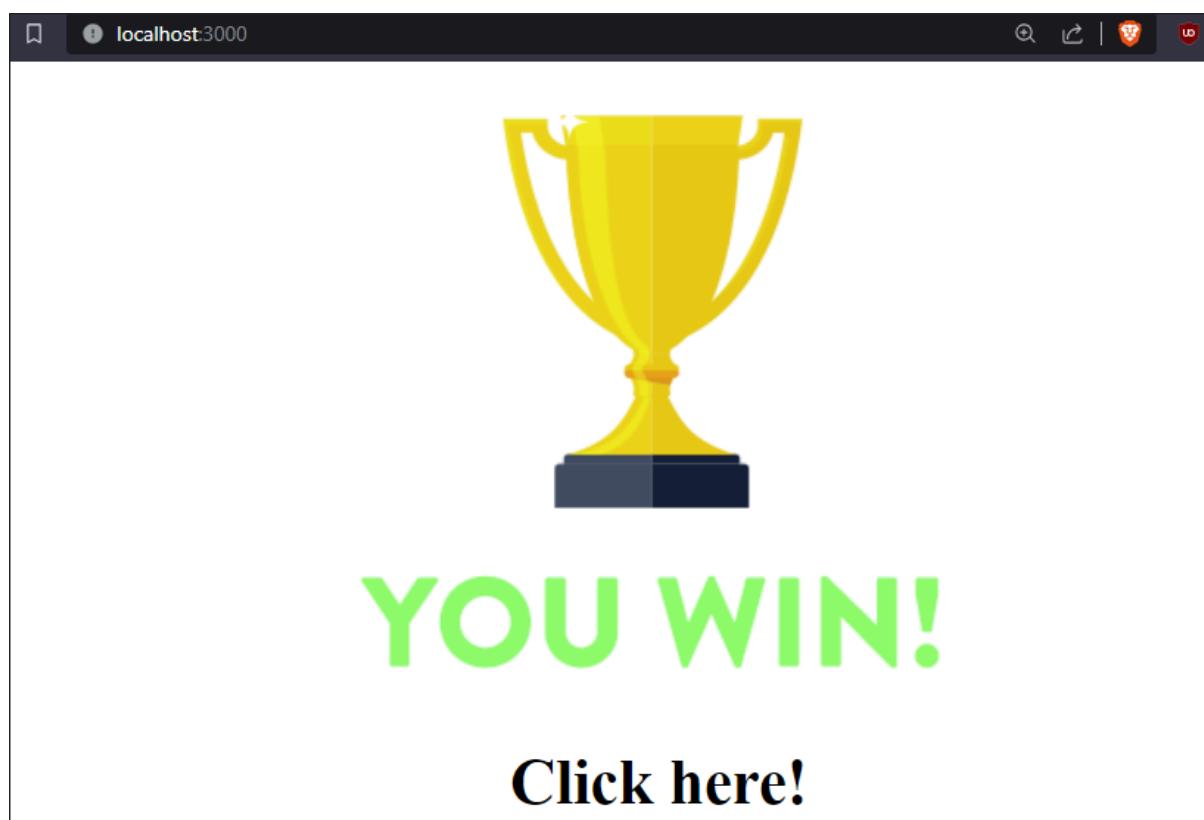
4.1. Realizacija napada

Ispod vidimo [početni spisak korisnika](#) u našoj aplikaciji ([slika 4.1.1](#)), odnosno vidimo izgled stranice kada pokrenemo [server aplikacije napadača](#) ([slika 4.1.2](#)).

The screenshot shows a browser's developer tools interface with the Network tab selected. At the top, there is a table titled 'Users' displaying four entries. Below the table, the Network tab shows a list of network requests. The requests listed are:

Name	Method	Status	Type	Size	Time	Waterfall
persons	GET	200	document	8.0 kB	43 ms	
jquery-3.4.1.slim.min.js	GET	200	script	(memory cache)	0 ms	
bootstrap.min.js	GET	200	script	(memory cache)	0 ms	
qrcode.min.js	GET	200	script	(memory cache)	0 ms	
bootstrap.min.css	GET	200	stylesheet	(disk cache)	5 ms	
background.jpg	GET	200	jpeg	730 kB	34 ms	

Slika 4.1.1. Prikaz spiska korisnika



Slika 4.1.2. Stranica napadača sa slikom pehara na koji treba kliknuti

Razvoj bezbednog softvera

Na slici 4.1.1. iznad se vidi da je korisnik sa *id* jednakim 1 *bruce wayne*. Zato ćemo u *index.html* aplikacije napadača dodati kod koji šalje zahtev za promenom podataka o korisniku u *Dobby Free Elf*.

```
JavaScript
<script>
function exploit() {
  const formData = new FormData();
  formData.append('id', 1);
  formData.append('firstName', 'Dobby');
  formData.append('lastName', 'Free Elf');

  fetch('http://localhost:8080/update-person', {
    method: 'POST',
    body: formData,
    mode: 'no-cors',
    credentials: 'include'
  });
}
</script>
```

Klikom na pehar ([na slici 4.1.2](#)) ovaj zahtev će se izvršiti. Videćemo u *Network* tabu da se napad izvršio, a da je *POST* zahtev uspešno poslat ([slika 4.1.3](#)).



YOU WIN!

Click here!

DevTools - localhost:3000/

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights

Preserve log Disable cache No throttling

Name	Method	Status	Type	Size	Time	Waterfall
update-person	POST	302	fetch / Redirect	615 B	24 ms	
1	GET	200	fetch	599 B	42 ms	

2 requests | 1.2 kB transferred | 0 B resources

Slika 4.1.3. CSRF napad na sajtu napadača, klikom na pehar

#	First Name	Last Name	Email	
1	Dobby	Free Elf	notBatman@gmail.com	View profile
2	Peter	Petigrew	oneFingernailFewerToClean@gmail.com	View profile
3	Tom	Riddle	theyGotMyNose@gmail.com	View profile
4	Santa	Clause	st@northPole.com	View profile

Slika 4.1.4. Korisnik sa id = 1 se sada zove Dobby Free Elf

4.2. Realizacija odbrane

Za realizaciju odbrane trebalo bi prvo kreirati klasu `CsrfHttpSessionListener` u paketu `config`, ali je ona za nas već kreirana. U ovoj klasi kreira se CSRF token.

Realizujemo sledeće **korake u zaštiti od CSRF napada**:

1. U kod koji dohvata osobu sa datim `id` (fajl `PersonsController`) dodamo i parametar `HttpSession`. Iz tog parametra izvlačimo CSRF token i dodajemo u `model`, kao atribut koji se zove **CSRF_TOKEN**. Ovime je token upisan kao deo `html` stranice za promenu detalja osobe koja će biti prikazana u pregledaču.

Java

```
@GetMapping("/persons/{id}")
public String person(@PathVariable int id, Model model, HttpSession session) {
    model.addAttribute("CSRF_TOKEN", session.getAttribute("CSRF_TOKEN"));
    model.addAttribute("person", personRepository.get("") + id));
    return "person";
}

@GetMapping("/myprofile")
public String self(..., HttpSession session) {
    ...
    model.addAttribute("CSRF_TOKEN", session.getAttribute("CSRF_TOKEN"));
    model.addAttribute("person", personRepository.get("") + user.getId()));
    ...
}
```

2. U formi na stranici za promenu detalja korisnika (`person.html` fajl), koji je kodom iznad dovučen, treba dodati `hidden` polje sa vrednošću **CSRF_TOKEN**.

Java

```
<form method="POST" action="/update-person" class="col-5">
    ...
    <input type="hidden" name="csrfToken" th:value="${CSRF_TOKEN}">
    <button type="submit" class="btn btn-primary">Save</button>
</form>
```

3. Zbog ovoga će token za svaki `submit` forme biti poslat. Server će proveriti da li primljeni token odgovara onom uskladištenom u podacima sesije korisnika.
 - a. Primer ove provere je u metodi `updatePerson` (fajl `PersonsController`).

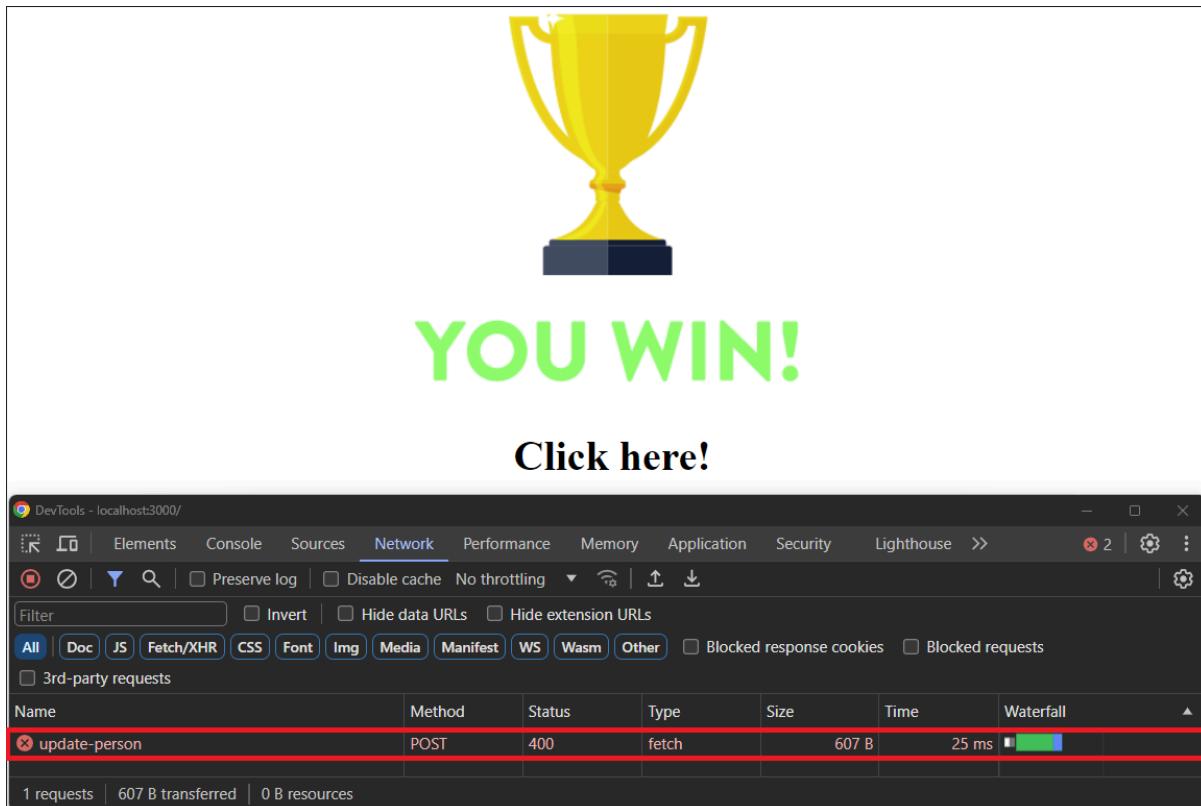
- b. Pre izvršenja samog *HTTP* zahteva za promenom sadržaja baze proverava se da li je token isti kao token uskladišten u podacima sesije.

Java

```
@PostMapping("/update-person")
public String updatePerson(
    Person person,
    HttpSession session,
    @RequestParam("csrfToken") String csrfToken
) throws AccessDeniedException {
    String csrf = session.getAttribute("CSRF_TOKEN").toString();
    if (!csrf.equals(csrfToken)) {
        throw new AccessDeniedException("Forbidden");
    }

    personRepository.update(person);
    return "redirect:/persons/" + person.getId();
}
```

Nakon zaštite programa od ovog napada, vidimo da se ništa ne dešava kada kliknemo na pehar, tj. korisnik sa *id* = 1 ostaje bruce wayne ([slika 4.2.1](#)).



Slika 4.2.1. Neuspeli CSRF napad nakon primene zaštite

5. Implementacija autorizacije

Potrebno je dodati rolu *BUYER*, kao i povezati korisnike sa određenim rolama u fajlu „*data.sql*“. Potom dodajemo permisije, i veze rola sa permisijama.

```
JavaScript
insert into roles(id, name)
values (1, 'ADMIN'),
       (2, 'MANAGER'),
       (3, 'BUYER');

insert into user_to_roles(userId, roleId)
values (4, 1),
       (3, 2),
       (2, 3),
       (1, 3);

insert into permissions(id, name)
values (1, 'ADD_COMMENT'),
       (2, 'VIEW_GIFT_LIST'),
       (3, 'CREATE_GIFT'),
       (4, 'VIEW_PERSONS_LIST'),
       (5, 'VIEW_PERSON'),
       (6, 'UPDATE_PERSON'),
       (7, 'UPDATE_SELF'),
       (8, 'VIEW_MY_PROFILE'),
       (9, 'RATE_GIFT'),
       (10, 'BUY_GIFT');

insert into role_to_permissions(roleId, permissionId)
values (1, 1), (2, 1), (3, 1),
       (1, 2), (2, 2), (3, 2),
       (1, 3), (2, 3),
       (1, 4), (2, 4),
       (1, 5),
       (1, 6),
       (1, 7), (2, 7), (3, 7),
       (1, 8), (2, 8), (3, 8),
       (1, 9), (3, 9),
       (1, 10), (2, 10), (3, 10);
```

Treba izmeniti metod *authenticate* iz klase *DatabaseAuthenticationProvider*, kojim se ulogovanom korisniku dohvataju permisije, što je već urađeno za nas.

5.1. Permisija „ADD_COMMENT“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu dozvoliti pristup *endpoint*-u samo korisnicima sa permisijom.
Treba dodati adekvatnu anotaciju u fajlu *CommentController*.

Java

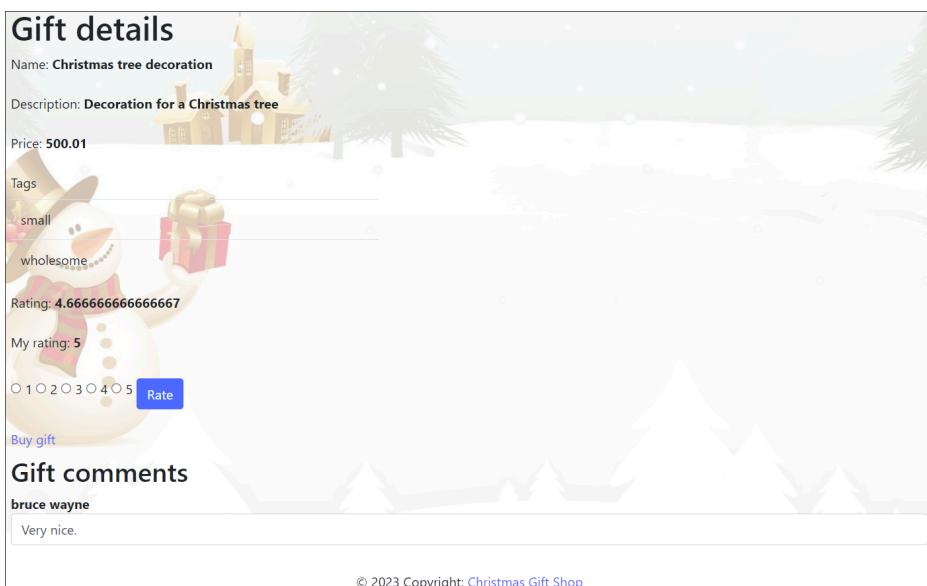
```
@PostMapping(value = "/comments", consumes = "application/json")
@PreAuthorize("hasAuthority('ADD_COMMENT')")
public ResponseEntity<Void> createComment(...) { ... }
```

2. Na frontendu uslovno renderovati elemente za dodavanje komentara. Treba dodati adekvatan atribut u fajlu *gift.html*.

JavaScript

```
<div class="form-group" sec:authorize="hasAuthority('ADD_COMMENT')">
    <label for="addComment">Add comment</label>
    ...
</div>

...
if (document.getElementById("createComment") !== null) {
    document.getElementById("createComment")
        .addEventListener("click", function () {
            ...
        });
}
```



Slika 5.1.1. Korisnik bez permisije ne može da dodaje komentare

5.2. Permisija „VIEW_GIFT_LIST“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu, u fajlu *GiftController*, dozvoliti pristup *endpoint-u* samo korisnicima sa permisijom.
 - a. Treba dodati adekvatnu anotaciju za metodu *search*.
 - b. Treba vratiti praznu listu da ne bi „pukla“ stranica ako korisnik može da kupuje poklone (ako sme da koristi link „Add Gift“ na stranici). Pošto nije zabranjeno da korisnik vidi stranice za pojedinačne poklone, treba dodati proveru da li je *id == null* u *showGift* metodi.

Java

```
@GetMapping(value = "/api/gifts/search", produces = "application/json")
@ResponseBody
@PreAuthorize("hasAuthority('VIEW_GIFT_LIST')")
public List<Gift> search(...) throws SQLException {
    ...
}

@GetMapping("/")
@PreAuthorize("hasAuthority('VIEW_GIFT_LIST') or
hasAuthority('CREATE_GIFT')")
public String showSearch(Model model) {
    if (!SecurityUtil.hasPermission("VIEW_GIFT_LIST")) {
        model.addAttribute("gifts", new ArrayList<>()); // Empty list
        return "gifts";
    }
    ...
}

@GetMapping("/gifts")
@PreAuthorize("hasAuthority('VIEW_GIFT_LIST') or
hasAuthority('CREATE_GIFT')")
public String showGift(...) {
    if (!SecurityUtil.hasPermission("VIEW_GIFT_LIST") && id == null) {
        model.addAttribute("gifts", new ArrayList<>()); // Empty list
        return "gifts";
    }
    ...
}
```

2. Na frontendu, u fajlu *gifts.html*, uslovno renderovati listu poklona i pretragu.

JavaScript

```
<div layout:fragment="content">
```

```
<h1>Gifts</h1>
<div class="container">
    <div class="input-group mb-3"
sec:authorize="hasAuthority('VIEW_GIFT_LIST')">
        <input type="text" id="searchQuery" class="form-control">
        ...
    </div>
    ...
    <table class="table">
        <thead sec:authorize="hasAuthority('VIEW_GIFT_LIST')">
            ...
        </thead>
        <tbody id="giftsTableContent"
sec:authorize="hasAuthority('VIEW_GIFT_LIST')">
            ...
        </tbody>
    </table >
    <a href="/create-form">Add Gift</a>
</div>
...
<script>
    ...
    if (document.getElementById('searchQuery') !== null) {
        const searchQueryInput = document.getElementById('searchQuery');
        searchQueryInput.addEventListener('keyup', onEnter);
    }
</script>
</div>
```

3. Na frontendu, u *layout.html*, uslovno renderovati link „Gifts“.

JavaScript

```
<li class="nav-item" sec:authorize="hasAuthority('VIEW_GIFT_LIST') or
hasAuthority('CREATE_GIFT')">
    <a class="nav-link" th:href="@{/gifts}">Gifts</a>
</li>
```



Slika 5.2.1. Stranice sa listom i pretragom poklona, ako korisnik nema permisiju

5.3. Permisija „CREATE_GIFT“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu dozvoliti pristup *endpoint*-u samo korisnicima sa permisijom.

Treba dodati adekvatne anotacije u fajlu *GiftController*.

- a. Prva anotacija vezana je za prikaz [stranice za dodavanje poklona](#).
- b. Druga anotacija vezana je za kreiranje novog poklona u bazi.

Java

```
@GetMapping("/create-form")
@PreAuthorize("hasAuthority('CREATE_GIFT')")
public String CreateForm(Model model) {
    ...
}

@PostMapping("/gifts")
@PreAuthorize("hasAuthority('CREATE_GIFT')")
public String createGift(NewGift newGift) throws SQLException {
    ...
}
```

2. Na frontendu uslovno renderovati element koji nas vodi na stranicu za dodavanje novog poklona. Treba dodati adekvatan atribut u fajlu *gifts.html*.

JavaScript

```
<a href="/create-form" sec:authorize="hasAuthority('CREATE_GIFT')">Add
Gift</a>
```

#	Title	Description	Price	
1	Christmas tree decoration	Decoration for a Christmas tree	500.01	Details
2	Christmas tree	A fake christmas tree	1500.99	Details
3	Santa Clause	A small Santa Clause for decoration	750.0	Details
4	Sweater	New Year sweater	3000.0	Details

Slika 5.3.1. Stranica sa listom poklona, bez opcije „Add Gift“

5.4. Permisija „VIEW_PERSONS_LIST“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu, u fajlu *PersonsController*, dozvoliti pristup endpoint-u samo korisnicima sa permisijom.

Java

```
@GetMapping(value = "/persons/search", produces = "application/json")
@ResponseBody
@PreAuthorize("hasAuthority('VIEW_PERSONS_LIST')")
public List<Person> searchPersons(...) throws SQLException {
    return personRepository.search(searchTerm);
}

@GetMapping("/persons")
@PreAuthorize("hasAuthority('VIEW_PERSONS_LIST')")
public String persons(Model model) {
    ...
}
```

2. Na frontendu, u *layout.html*, uslovno renderovati link „Users“.

JavaScript

```
<li class="nav-item" sec:authorize="hasAuthority('VIEW_PERSONS_LIST')">
    <a class="nav-link" th:href="@{/persons}">Users</a>
</li>
```

5.5. Permisija „VIEW_PERSON“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu dozvoliti pristup *endpoint*-u samo korisnicima sa permisijom, ili ako se radi o korisniku koji gleda svoj profil¹.

Java

```
@GetMapping("/persons/{id}")
public String person(@PathVariable int id, ...) {
    if (!SecurityUtil.hasPermission("VIEW_PERSON") &&
        SecurityUtil.getCurrentUser().getId() != id) {
        throw new AccessDeniedException("Forbidden");
    }
    ...
}
```

2. Na frontendu uslovno renderovati element „View profile“.

- a. Treba dodati adekvatan sec:authorize.
- b. Zatim u funkciji koja obrađuje search funkcionalnost proveriti da li je renderovan zbog sec:authorize dati element, pa ako jeste, opet ga prikazati, a u suprotnom ga ne prikazivati.

JavaScript

```
<td id="viewProfile" sec:authorize="hasAuthority('VIEW_PERSON')">
    <a th:href="@{/persons/{id}(id=${person.id})}">View profile</a>
</td>

...
const tableContent = document.getElementById("tableContent");
const viewProfileLink = document.getElementById("viewProfile");
tableContent.textContent = '';

persons.forEach(function(person) {
    ...
    if (viewProfileLink !== null) {
        tdElement = document.createElement("td");
        tdElement.innerHTML = '<a href="/persons/' + person.id + '">View
profile</a>';
        tableRowElement.appendChild(tdElement);
    }
    ...
});
```

¹ Razlog za ovaj uslov je to što nakon promene podataka na <http://localhost:8080/myprofile> korisnik biva preusmeren na stranicu svog profila <http://localhost:8080/persons/:id>.

#	First Name	Last Name	Email
1	bruce	wayne	notBatman@gmail.com
2	Peter	Petigrew	oneFingernailFewerToClean@gmail.com
3	Tom	Riddle	theyGotMyNose@gmail.com
4	Santa	Clause	st@northPole.com

Slika 5.1.1. Stranica sa listom korisnika, bez opcije „View profile“

5.6. Permisija „UPDATE_PERSON“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu dozvoliti pristup *endpoint*-u samo korisnicima sa permisijom.
 - a. Treba dodati adekvatnu anotaciju. Zahtev mogu da izvrše korisnici koji:
 - i. menjaju podatke drugih osoba (permisija *UPDATE_PERSON*),
 - ii. menjaju sopstvene podatke (permisija *UPDATE_SELF*).
 - b. Role *MANAGER* i *BUYER* mogu menjati samo svoje podatke. Da li se radi o njima proveravamo ispod, u kodu metode *updatePerson*.

Java

```
@PostMapping("/update-person")
@PreAuthorize("hasAuthority('UPDATE_PERSON') or
hasAuthority('UPDATE_SELF')")
public String updatePerson(Person person, ...) throws AccessDeniedException
{
    ...
    // For 'MANAGER' and 'BUYER'
    if (!SecurityUtil.hasPermission("UPDATE_PERSON") &&
        SecurityUtil.hasPermission("UPDATE_SELF")) {
        User currentUser = SecurityUtil.getCurrentUser();
        if (currentUser.getId() != Integer.parseInt(person.getId())) {
            throw new AccessDeniedException("Forbidden");
        }
    }
    personRepository.update(person);
    return "redirect:/persons/" + person.getId();
}
```

2. Na frontendu korisnici koji nemaju ovu dozvolu svakako ne mogu da vide profile drugih osoba, jer sigurno nemaju ni dozvolu *VIEW_PERSON*. Svi korisnici mogu da vide i menjaju svoj profil. Zato na frontendu nema izmena.
3. Izmenimo kod da isto važi i za brisanje korisnika:

Java

```
@DeleteMapping("/persons/{id}")
@PreAuthorize("hasAuthority('UPDATE_PERSON') or
hasAuthority('UPDATE_SELF')")
public ResponseEntity<Void> person(@PathVariable int id) {
    // For 'MANAGER' and 'BUYER'
    if (!SecurityUtil.hasPermission("UPDATE_PERSON") &&
        SecurityUtil.hasPermission("UPDATE_SELF")) {
        User currentUser = SecurityUtil.getCurrentUser();
        if (currentUser.getId() != id) {
```

```
        throw new AccessDeniedException("Forbidden");
    }
}
...
}
```

5.7. Permisija „VIEW_MY_PROFILE“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu dozvoliti pristup *endpoint*-u samo korisnicima sa permisijom.

Treba dodati adekvatnu anotaciju u fajlu *PersonsController*.

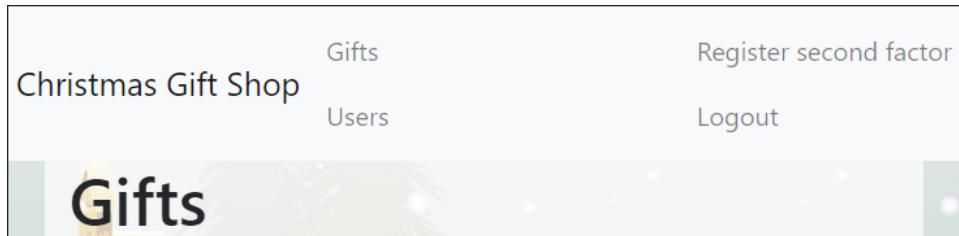
Java

```
@GetMapping("/myprofile")
@PreAuthorize("hasAuthority('VIEW_MY_PROFILE')")
public String self(...) {
    ...
}
```

2. Na frontendu uslovno renderovati element sa linkom „My Profile“. Treba dodati adekvatan atribut u fajlu *layout.html*.

JavaScript

```
<li class="nav-item" sec:authorize="hasAuthority('VIEW_MY_PROFILE')">
    <a class="nav-link" th:href="@{/myprofile}">My Profile</a>
</li>
```



Slika 5.7.1. Zaglavje sa navigacijom bez opcije „My Profile“

5.8. Permisija „RATE_GIFT“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu dozvoliti pristup *endpoint*-u samo korisnicima sa permisijom.

Treba dodati adekvatnu anotaciju u fajlu *RatingsController*.

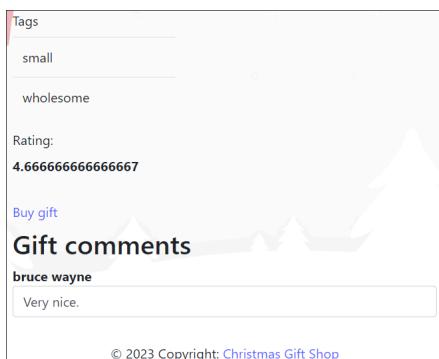
Java

```
@PostMapping(value = "/ratings", consumes = "application/json")
@PreAuthorize("hasAuthority('RATE_GIFT')")
public String createOrUpdateRating(...) {
    ...
}
```

2. Na frontendu uslovno renderovati elemente vezane za ocenu poklona. Treba dodati adekvatne atribute u fajlu *gift.html*.

JavaScript

```
<div class="form-group" sec:authorize="hasAuthority('RATE_GIFT')">
    <label>My rating:</label>
    <b><label th:text="${userRating}"></label></b>
</div>
<div class="form-group" sec:authorize="hasAuthority('RATE_GIFT')">
    ...
    <button id="rateGift" class="btn btn-primary mt-2">Rate</button>
</div>
...
if (document.getElementById("rateGift") !== null) {
    document.getElementById("rateGift")
        .addEventListener("click", function () {
            ...
        });
}
```



Slika 5.8.1. Stranica o poklonu, bez mogućnosti za dodavanje ocene

5.9. Permisija „BUY_GIFT“

Ovo podrazumeva sledeće izmene u kodu:

1. Na bekendu dozvoliti pristup *endpoint*-u samo korisnicima sa permisijom.

Treba dodati adekvatnu anotaciju u fajlu *GiftController*.

```
Java
@GetMapping("/buy-gift/{id}")
@PreAuthorize("hasAuthority('BUY_GIFT')")
public String showBuyCar(
    ...
}

@PostMapping("/buy-gift/{id}")
@PreAuthorize("hasAuthority('BUY_GIFT')")
public String buyCar(...) {
    ...
}
```

2. Na frontendu uslovno renderovati element sa linkom „Buy gift“. Treba dodati adekvatne atribute u fajlu *gift.html*.

```
JavaScript
<div class="mt-2 mb-2" sec:authorize="hasAuthority('BUY_GIFT')">
    <a th:href="@{'/buy-gift/' + ${gift.id}}>Buy gift</a>
</div>
```