# Cross-site scripting (XSS)

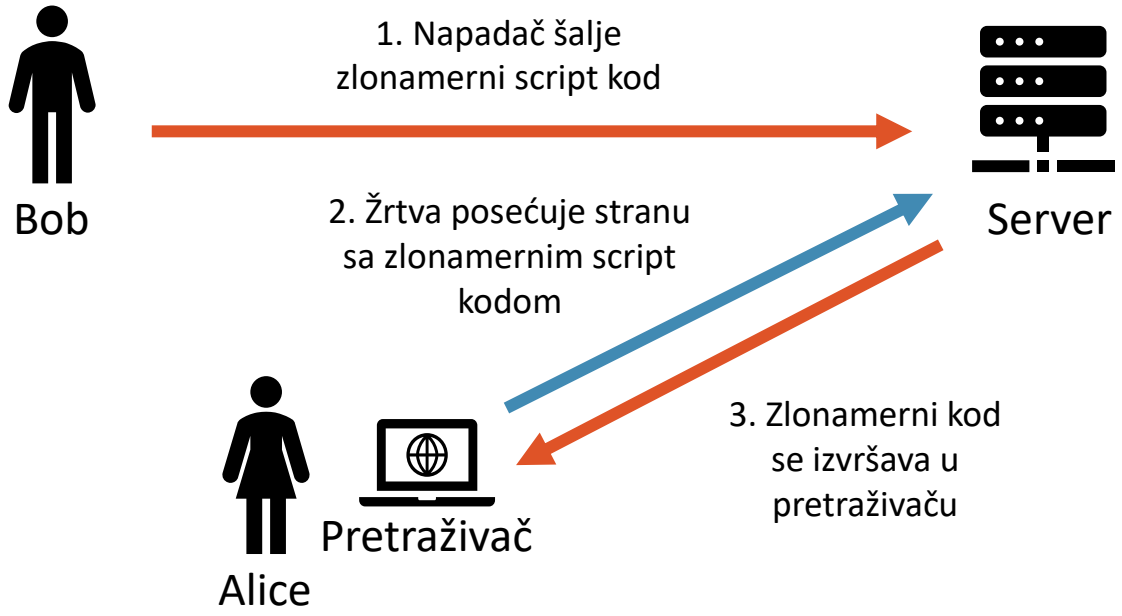
# ОБАВЕШТЕЊЕ ЗА СТУДЕНТЕ

- Настава на предмету Развој безбедног софтвера подразумева изучавање различитих механизама којима се нарушава информациона безбедност и врше напади на интернет апликације и софтверске системе.
- Студенти на предмету Развој безбедног софтвера могу ове методе за потребе изучавања да користе искључиво у оквиру затвореног лабораторијског окружења које је обезбеђено за наставу на предмету Развој безбедног софтвера.
- Студенти не могу да подразумевају да су на било који начин охрабрени од стране наставника или да им се препоручује да користе ове методе који се изучавају према другим апликацијама Електротехничког факултета или апликацијама било ког трећег правног или физичког лица.
- Свака евентуална активност коју би предузео неки студент коришћењем ових метода и механизама према апликацијама које нису у оквиру лабораторије на предмету искључива је одговорност студента.

# XSS

Ranjivost postoji u aplikacijama koje prikazuju podatke poslate od strane korisnika bez pravilne validacije ili sanitizacije tog sadržaja.

XSS je napad na pretraživač korisnika - ne na server.



# XSS vrste

XSS napade možemo podeliti na osnovu toga odakle zlonamerni kod dolazi:

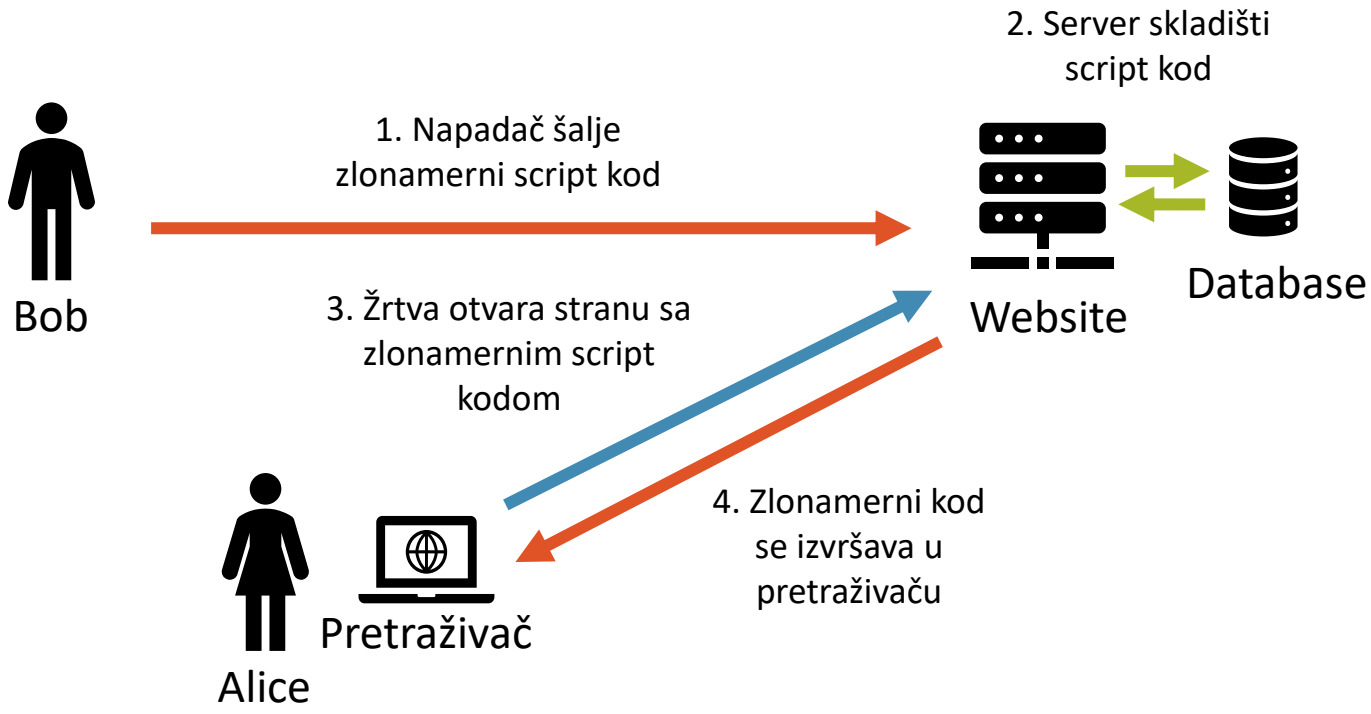
1. Uskladišteni XSS (perzistentni)
2. Reflektovani XSS (neperzistentni)
    1. Server-side
    2. Client-side

# Uskladišteni XSS

Postoji kada se korisnikov unos skladišti u bazi podataka, obično kroz element unosa na stranici, npr:

- forum tekst,
- komentar na proizvod u prodavnici automobila.

Script kod živi (zbog perzistencije) zauvek i važi za sve druge korisnike.

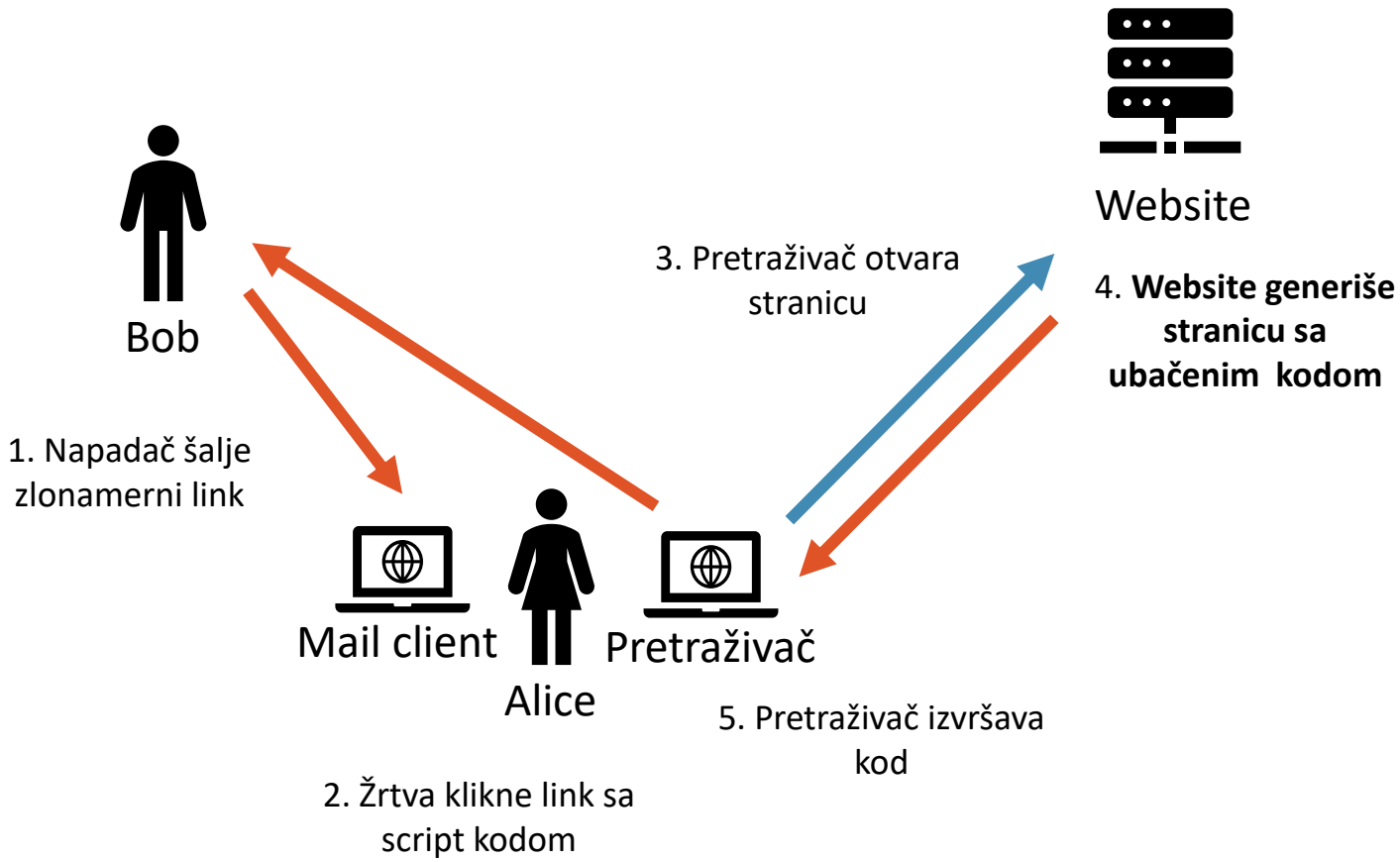


# Reflektovani XSS Server side

Postoji kada se korisnikov unos ispisuje na stranici

- rezultat pretrage
- grešku web sajta
- neki drugi odgovor koji uključuje deo ili ceo unos od strane korisnika

Targetira određenog korisnika.



# Reflektovani XSS
# Server primer:

https://www.mySite.com/greeting?greet=Hello+World

```
<!DOCTYPE html>
<html>
    <body>

        <h1>Hello world</h1>

    </body>
</html>
```

Imamo link na osnovu koga server stvara zahtevanu stranicu. Dakle napad se reflektuje preko servera!

https://www.mySite.com/greeting?q=<script>alert('stranger danger')</script>

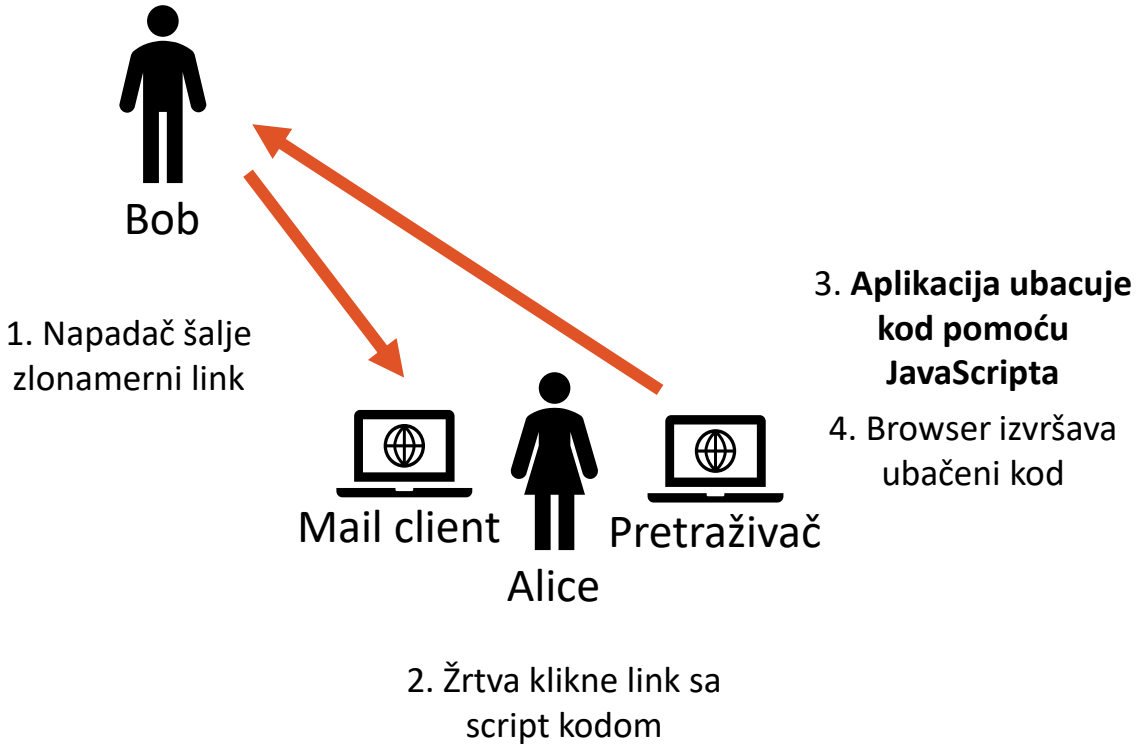
```
<!DOCTYPE html>
<html>
    <body>

    <h1><script>alert('stranger danger!')</script></h1>

    </body>
</html>
```

# Reflektovani XSS
# Client side

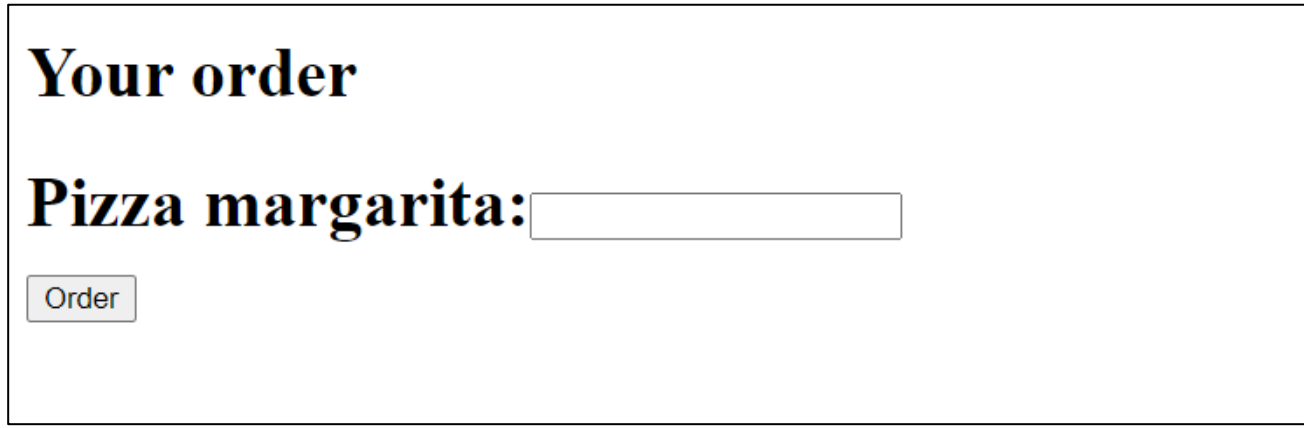
Postoji kada se korisnikov unos ispisuje na stranici, npr:

- rezultat pretrage,
- greška web sajta,
- neki drugi odgovor koji uključuje deo ili ceo unos od strane korisnika.

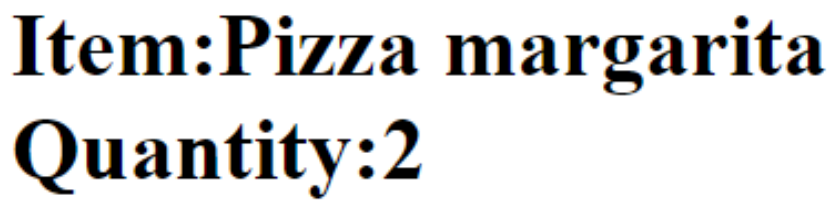
Targetira određenog korisnika.



```html
<!DOCTYPE html>
<html>
    <body>

    <h1>Your order<h1/>

    <label>Pizza margarita:</label><input type='number' />

    <br/>

    <button>Order</button>

    </body>
</html>
```




```html
<!DOCTYPE html>
<html>
    <body>

    <h1>Basket<h1/>

    <label>Item:</label><label>Pizza margarita</label>

    <br/>

    <label>Quantity:</label><label><script>document.getQuantity()...</script></label>

    </body>
</html>
```



# Posledice (rizik i uticaj)

- Poverljivost
- Integritet
- Autentifikacija i autorizacija
- Reputacija

# Posledice (rizik i uticaj)

- Poverljivost
  - Slanje podataka sa stranice napadaču (POST zahtev na napadačev server)
- Integritet
  - Izmena podataka na stranici sa korisnikovim privilegijama
- Autentifikacija i autorizacija
  - Krađa sesije (slanje cookie napadaču)
  - Eskalacije privilegija (administrator, direktor firme, državni zvaničnik...)
- Reputacija
  - Ubacivanje neprikladnih sadržaja na web sajt  (“website defacement”)

# Hackers still exploiting eBay's stored XSS vulnerabilities in 2017

**17th February, 2017**





Mar 12, 2019 |

Malware and Vulnerabilities

**Hackers abuse XSS vulnerability in cart plugin to target WordPress-based shopping sites**

Cyware Hacker News

# Microsoft Outlook for Android Open to XSS Attacks

Author:
Tara Seals

June 21, 2019 / 3:50 pm

# Server sesija

Šta je server sesija korisnika?

# Server sesija

Šta je server sesija korisnika?

- Server sesija korisnika se uspostavlja posle autentifikacije (logina) – završava se nakon logouta ili definisanog isteka
- Označava da je korisnikova autentifikacija trenutno važeća
- Pored toga omogućava skladištenje nekih dodatnih podataka u toku sesije (keširanje)

Sesija se obično identifikuje pomoću cookie-ja. Cookie se šalje sa svakim HTTP requestom.

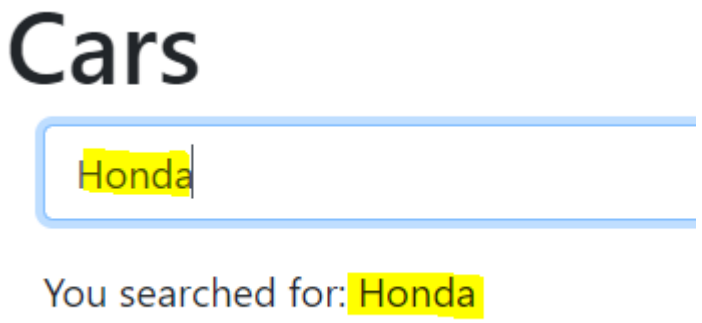
```
Connection: keep-alive
Cookie: JSESSIONID=0E89D83BDAE8DD893D42FA65FB9EFC68
Host: localhost:8080
```

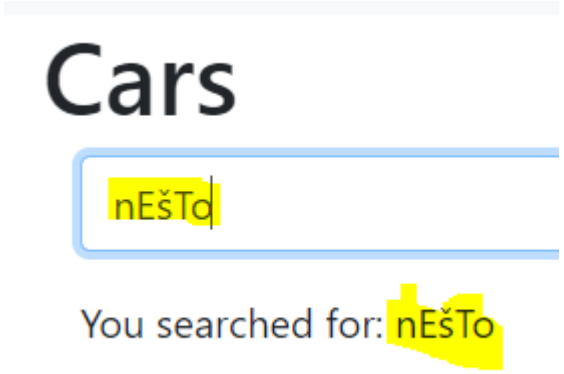
# Primer napada

Demonstracija prikazivanje cookie-ja

# Demonstracija

- U osnovi XSS napada je da aplikacija, odnosno pretraživač (browser), interpretira **korisnički unos** koji najčešće predstavlja zlonameran JavaScript kod
- Kada se pretražuju automobili, možemo primetiti da šta god unesemo u polje za pretragu ispisujemo i kao deo rezultata pretrage, kako bi korisnik imao uvid u kriterijum pretrage:



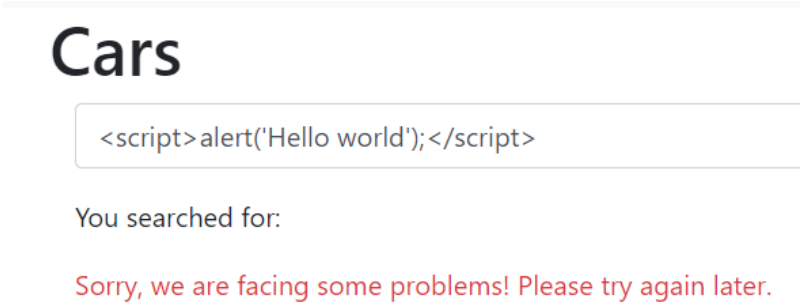



# Demontracija

- Ovakve situacije su interesantne za napadača i njegovo razmišljanje može biti sledeće:
  - Ukoliko umesto naziva ili modela kao kriterijum pretrage unesem neki JavaScript kod, da li će on biti izvršen?
- Postoji mnogo načina da se to postigne. S tim da su pretraživači (browseri) počeli da blokiraju neke od njih.
- Ukoliko, na primer, unesemo

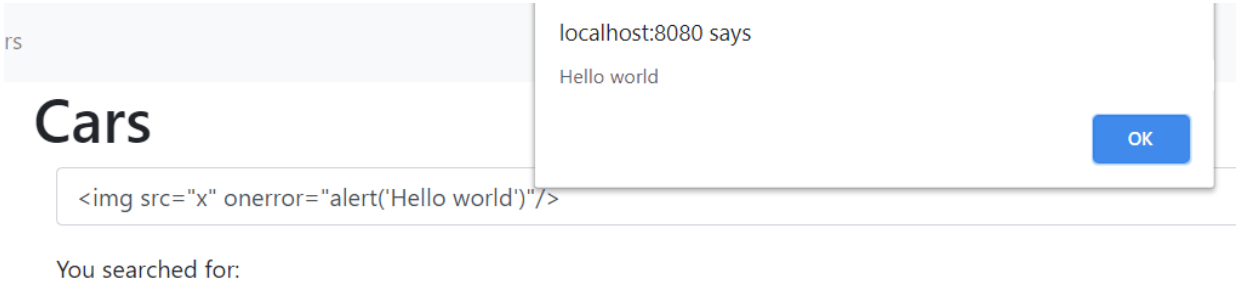
```
<script>alert('Hello world');</script>
```

kao kriterijum pretrage, to neće biti izršeno.



# Demontracija

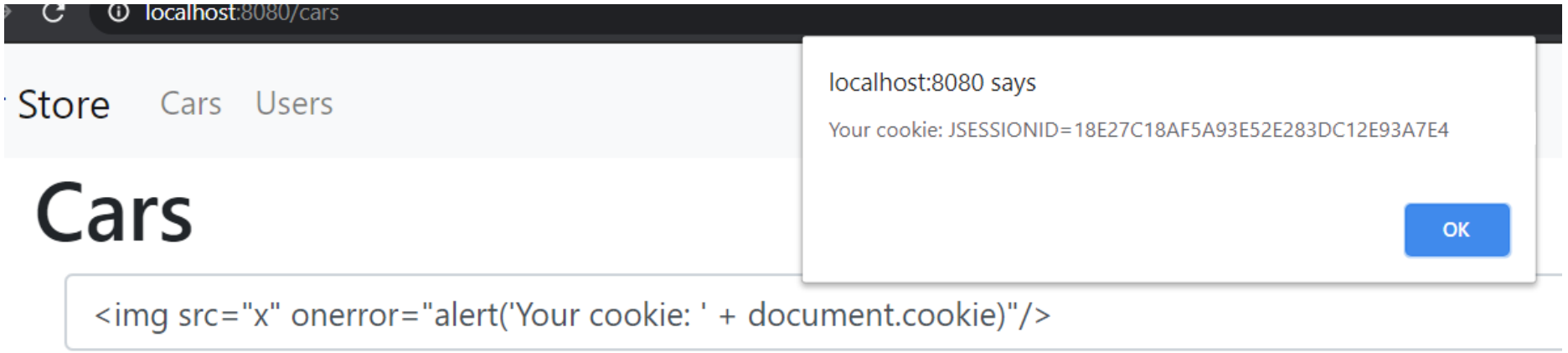
- Ali ukoliko unesemo `<img src="x" onerror="alert('Hello world')"/>` pretraživač (browser) će to interpretirati:




- Kao kriterijum pretrage uneli smo `img` tag koji se koristi za prikazivanje slike
- U slučaju greške, recimo pogrešnog URL-a, izvršava se `onerror` callback `img` taga
- Kako bi isprovocirali grešku, unosimo nevalidan URL slike, `src=“x”`

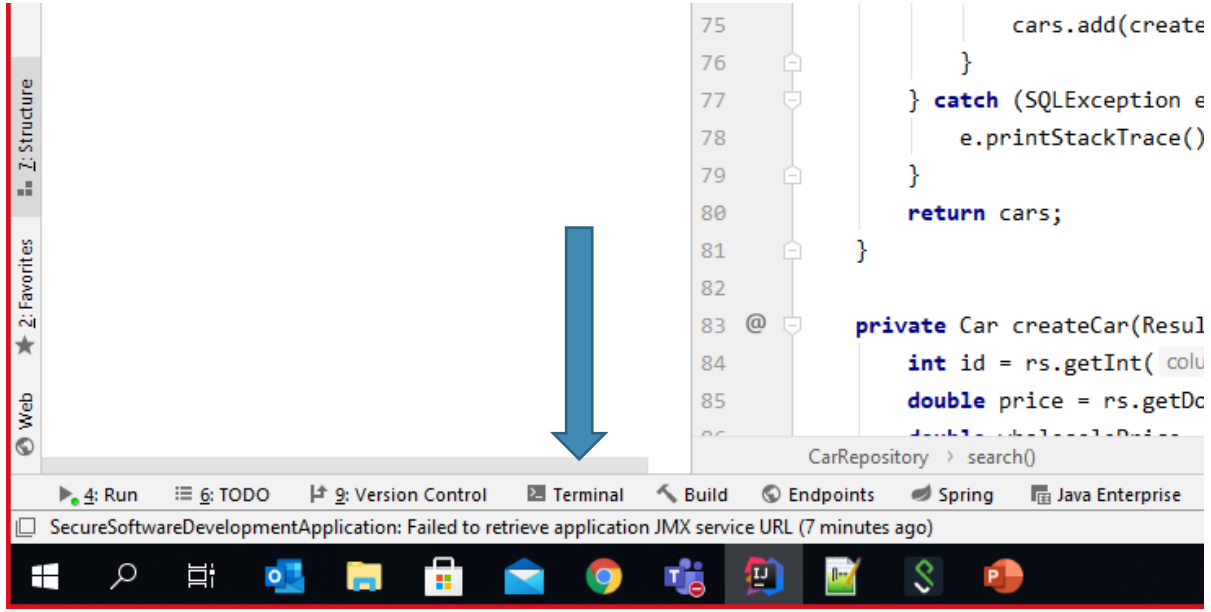
# Demontracija

- Korisnikov `cookie` možemo prikazati koristići `document.cookie`

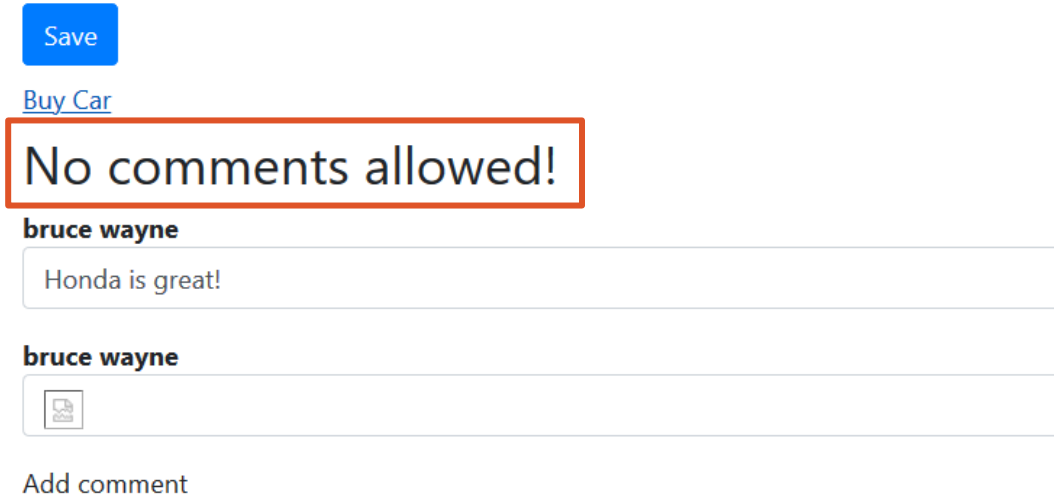


# Priprema za samostalni rad

1. Otvorite IntelliJ Idea
2. U terminalu izvršite u folderu **SecureSoftwareDevelopment**
   `git checkout xss`
3. Pokrenite aplikaciju
4. Prijavite se
   - username: bruce
   - password: wayne



# Samostalni rad




Opis

- Promenite naslov “Car comments”
- Klasifikujte koja je vrsta napada izvedena

(reflektovani server ili client-side, ili uskladišteni)

- Vreme: **~30 minuta**

# Rešenje zadatka

Demonstracija

# Zaštita od napada

# Zaštita

Kako bismo sve mogli da se zaštitimo od napada?

# Zaštita

- Dozvoliti samo sigurne karaktere kao unos korisnika („whitelist“)
- „Sanitizacija“ kroz izbacivanje ili zamenu nesigurnih karaktera („blacklist“)
- Korišćenje sigurnih outputting alata iz frameworka

# Sanitizacija

- Nesigurni karakteri mogu biti <, >, ? itd.
- Koristiti samo postojeće biblioteke za sanitizaciju
- Sanitizacija se uvek radi nad izlaznim podacima
- Nikada ne kreirajte svoje sanitizere!
  - Izuzetno je kompleksno kreirati sanitizere
  - Cheatsheat za različite napade: https://owasp.org/www-community/xss-filter-evasion-cheatsheet

```
<img src="x" onerror="alert('xss')">
```

```
<IMG SRC="/" onerror="alert(String.fromCharCode(88,83,83))"></img>
```

```
<img src=x
onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0
000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041">
```

Sanitizeri:

- JAVA: **HtmlUtils.htmlEscape**
- .NET: System.Web.Security.AntiXss.**AntiXssEncoder**

# Sanitizacija kroz framework

- JavaScript, na HTML Element DOM objektu, koristiti atribut

  **textContent** umesto **innerHTML**
- Thymeleaf, na HTML tagu, koristiti atribut

  **th:text** umesto **th:utext** (unescaped text)
- Angular automatski sanitizira sve outpute

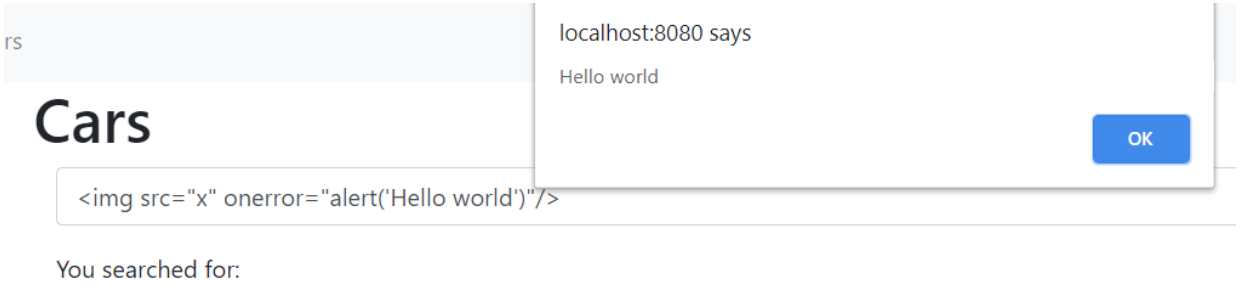
  Sanitizer mora eksplicitno da se zaobiđe sa nekoliko linija koda

# Popravka ranjivosti

Demonstracija

# Demonstracija

- Želimo da onemogućimo prethodno demonstrirani propust na pretrazi automobila



# Demonstracija

- Stranici za pretragu odgovara fajl *cars.html*
- Propust se javlja u okviru elementa čiji je id *searchTerm*

```
<p class="invisible">You searched for: <span id="searchTerm"></span></p>
```

- Kada se inicira pretraga klikom a *search* dugme, poziva se JavaSript metoda *search()* definisana u istom fajlu, *cars.html*

```
<button class="btn btn-outline-primary" id="carSearchButton" type="button" onclick="search()">
        Search
</button>
```

# Demonstracija

- U kodu metode *search()* primećujemo poziv metode *updateSearchTerm()*

```
function search() {
    clearSearchResults();
    hideErrorMessage();

    const searchQuery = getSearchQuery();
    updateSearchTerm(searchQuery);
    fetch(`/api/cars/search?query=${searchQuery}`)
        .then(handleErrors)
        .then(response => response.json())
        .then(updateSearchResults)
        .catch(showErrorMessage);
}
```

# Demonstracija

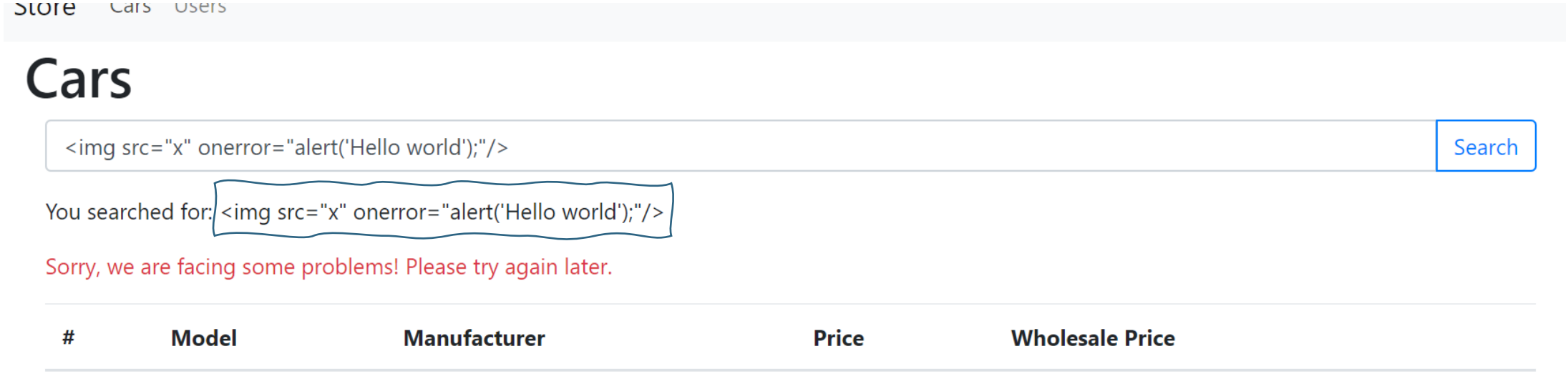
- Primećujemo da *updateSearchTerm()* koristi *innerHTML* za ažuriranje kriterijuma pretrage

```
function updateSearchTerm(searchQuery) {
    const searchTerm = document.getElementById('searchTerm');
    searchTerm.innerHTML = searchQuery;
    searchTerm.parentElement.classList.remove('invisible');
}
```

- Korišćenje *innerHTML* atributa *searchTerm* elementa dovodi do toga da pretraživač (browser) interpretira sadržaj tog elementa
- Popravka ranjivosti u ovom slučaju svodi se na korišćenje *textContent* atributa umesto *innerHTML*

# Demonstracija

- Nakon popravke ne prikazuje se *popup* prozor i kriterijum pretrage se ne interpretira



# Samostalni rad

- Opis: Popravite ranjivosti na dodavanju komentara
- Vreme: **15 minuta**

# Rešenje zadatka

Demonstracija