

SQL Injection



ОБАВЕШТЕЊЕ ЗА СТУДЕНТЕ

- Настава на предмету Развој безбедног софтвера подразумева изучавање различитих механизма 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 механизма према апликацијама које нису у оквиру лабораторије на предмету искључива је одговорност студента.

SQL Injection

- Javlja se u aplikacijama koje koriste relacione baze podataka
- Umesto podataka, zlonamerni korisnik aplikacije unosi komande koje SQL baza interpretira

Primer: Monthly Newsletter

Subscribe to our monthly newsletter

Your email:

SUBSCRIPTIONS	
EMAIL	CREATED_TIMESTAMP
mj23@bulls.com	30.12.2019 12:44:56
kb24@lakers.com	11.01.2020 20:18:42

Primer: Monthly Newsletter

Subscribe to our monthly newsletter

Your email:

SUBSCRIPTIONS	
EMAIL	CREATED_TIMESTAMP
mj23@bulls.com	30.12.2019 12:44:56
kb24@lakers.com	11.01.2020 20:18:42

```
String email = request.getEmail();  
String insertQuery = "INSERT INTO subscriptions(email) VALUES ('" + email + "')";
```

Primer napada

Subscribe to our monthly newsletter

Your email:

Primer napada

Subscribe to our monthly newsletter

Your email:

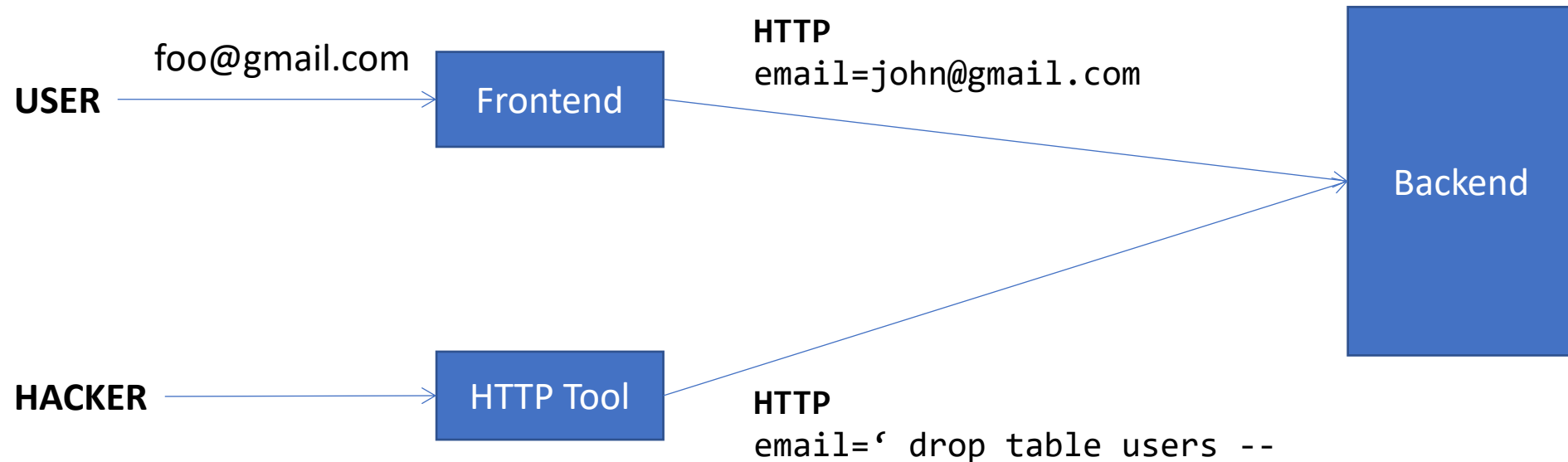
```
String insertQuery =  
    "INSERT INTO subscriptions(email) values (''); drop table subscriptions --');
```

Zašto se ovo desilo?

Zašto se ovo desilo?

- E-mail nije validiran
 - Validacija treba da se radi i na frontend-u na backend-u
 - Validacija treba da bude identična
 - Treba da bude što restriktivnija
 - Koristiti whitelist umesto blacklist pristupa
- Da li je validacija dovoljna?

Frontend i backend validacija



Zašto se ovo desilo?

- Šta kada imamo search polje?
- Ulazni podatak nema striktan format

Zašto se ovo desilo?

- Šta kada imamo search polje?
- Ulazni podatak nema striktan format
- Rešenje je:
 - Korišćenje parametrizovanih upita
 - PreparedStatement u Javi
 - SqlCommand i SqlParameter u .NET
 - Korišćenje ORM-a
- Šta su parametrizovani upiti?
- Šta je ORM?

Šta su parametrizovani upiti?

```
INSERT INTO products (name, price) VALUES (?, ?);
```

- Aplikacija bazi šalje SQL Query
- Baza ga kompajlira i pripremi za upotrebu
- Aplikacija bazi šalje podatke

Šta je ORM?

- Object Relational Mapping
- Pristup SQL bazi bez pisanja SQL-a
- Tabele su predstavljene klasama

Šta je ORM?

Product	
NAME	PRICE
Juice	200

```
class Product {  
    String name;  
    int price;  
}
```

Šta je ORM?

Product	
NAME	PRICE
Juice	200

```
class Product {  
    String name;  
    int price;  
}
```

Dodavanje proizvoda u tabelu

Bez ORM	Koristeći ORM
<pre>statement.executeQuery("INSERT INTO products (name, price) VALUES ('Car', '200')");</pre>	<pre>dbSession.save(new Product("Car", 200));</pre>

Koje su posledice SQL napada?

- Poverljivost
- Autentifikacija
- Autorizacija
- Integritet

Koje su posledice?

- Poverljivost
 - Baze podataka često čuvaju osetljive podatke
- Autentifikacija
 - Napadač se može prijaviti na sistem kao administrator bez poznavanja administratorske lozinke
- Autorizacija
 - Ukoliko se podaci vezani za autorizaciju čuvaju u bazi, napadač može manipulirati njima
- Integritet
 - Promena i brisanje postojećih podataka

Nije samo SQL ranjiv na injection

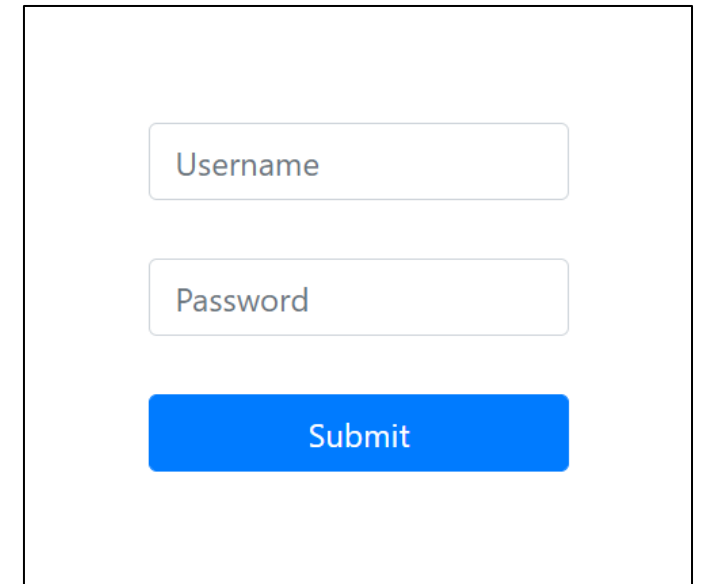
- NoSQL baze su takođe ranjive na injection
- Pored baza podataka, ima i drugih scenarija gde se ranjivost javlja
 - Bilo koja konkatencija korisničkog unosa na neku komandu
 - Evaluacija korisničkih unosa kao komande

Primer napada

Demonstracija na formi za prijavu

Demonstracija

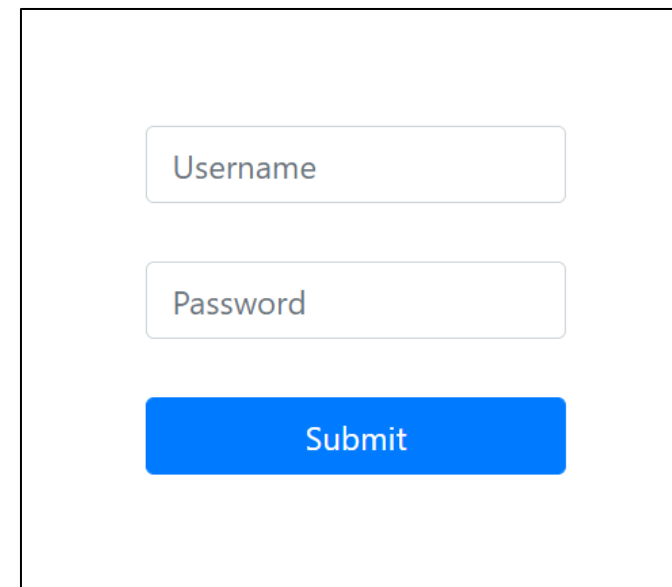
- Pred napadačem je forma za prijavu korisnika kao na slici
- **Njegov cilj je da se uspešno prijavi bez poznavanja korisničkog imena ili lozinke bilo kog korisnika aplikacije**
- Prvi korak napadača je analiza stranice na kojoj se nalazi, kako bi pronašao ulazne podatke koji potencijalno dolaze do baze podataka
- Ulazni podaci mogu biti očigledni, kao:
 - Polja forme
 - Parametri prosleđeni putem URL-a
<https://www.facebook.com/profile.php?id=100008916518443>
- Ali se mogu pronaći i analizom izvornog koda stranice sajta, uglavnom JavaScript koda
 - JavaScript se koristi za učitavanje podataka sa servera bez reload-a stranice
 - Desni klik na stranicu i *View page source* prikazuje izvorni kod stranice



A login form with two input fields labeled "Username" and "Password", and a blue "Submit" button.

Demonstracija

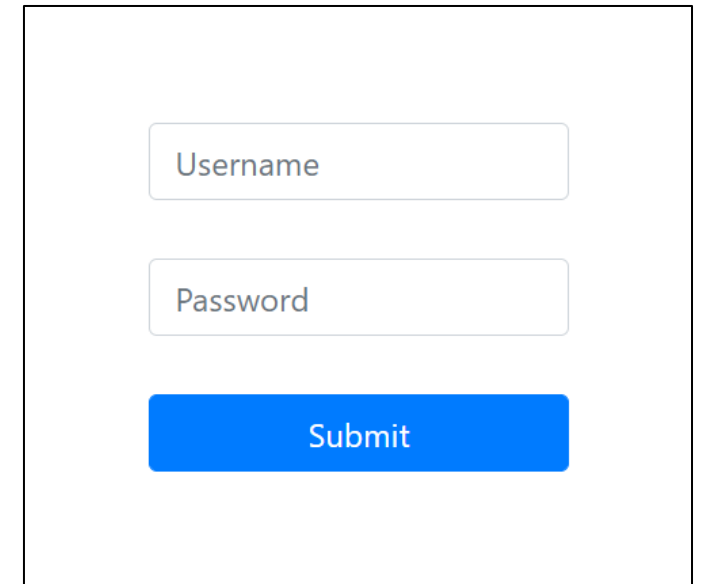
- U našem slučaju URL na kome se napadač nalazi je <http://localhost:8080/login> i pred njim je forma za prijavu
- Jedini ulazni podaci koje primećuje su korisničko ime i lozinka



A login form is displayed within a black rectangular border. It contains two white input fields with rounded corners. The first field is labeled 'Username' in a light gray font. The second field is labeled 'Password' in a light gray font. Below these fields is a solid blue button with the word 'Submit' written in white text.

Demonstracija

- Sledeći korak napadača je pretpostavka na koji način se ulazni podaci koriste u upitu koji se šalje bazi podataka
- U ovom slučaju napadač može pretpostaviti da upit izgleda na sledeći način:
- `SELECT * FROM users WHERE username='<KORISNICKO_IME>' AND password='<LOZINKA>'`
- Pretpostavka je da se aplikacija ponaša na sledeći način:
 - Ukoliko upit vrati 1 ili više rezultata (redova), korisničko ime i lozinka su ispravni i korisnik je uspešno prijavljen
 - Ukoliko upit nema rezultata, korisnik nije uspešno prijavljen
- Napadač će sada pokušati da unese korisničko ime i lozinku, takve da upit vraća bar jedan rezultat. Koji rezultat je u pitanju nije relevantno, jer aplikacija uzima obzir samo **broj rezultata**.

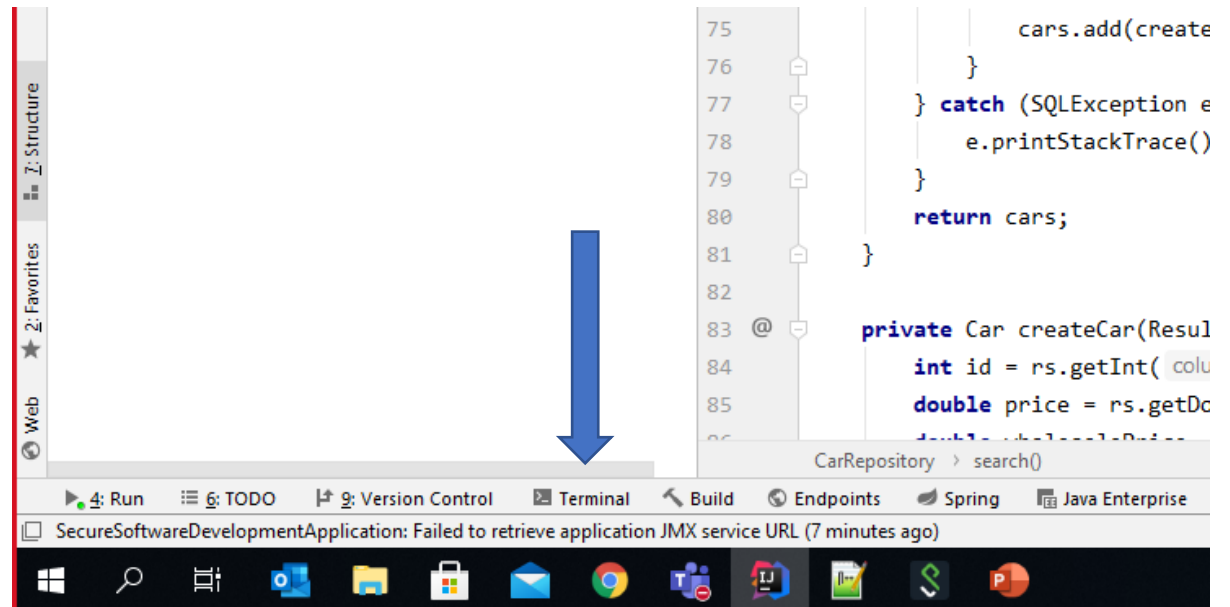


A login form interface. It consists of two text input fields stacked vertically. The first field is labeled 'Username' and the second is labeled 'Password'. Below these fields is a blue rectangular button with the text 'Submit' in white. The entire form is enclosed in a thin black border.

Primer napada

Priprema za samostalni rad

1. Otvorite IntelliJ Idea
2. U terminalu izvršite **git checkout sql-injection**
3. Zatim **git reset --hard**
4. Pokrenite aplikaciju



Samostalni rad

- Opis: Na stranici <http://localhost:8080/cars?id=1> preko forme za dodavanje komentara izvršiti SQL Injection napad koji dodaje novi automobil u tabelu **cars**
- Vreme: **30 min**
- Ukoliko završite ranije, izvršite SQL Injection napad preko pretrage automobila koji dodaje novi automobil u tabelu **cars**

Rešenje zadatka

Demonstracija

Rešenje zadatka

Wholesale Price

Save

Buy Car

Car comments

bruce wayne

Add comment

Create comment

Primer zaštite

Demonstracija

Demonstracija

- Cilj je zaštititi prijavu korisnika od SQL injection napada
- Kod koji proverava da li korisnik postoji u bazi nalazi se u klasi **UserRepository**, u metodi **validCredentials**

Primer zaštite

```
public boolean validCredentials(String username, String password) {  
    String query = "SELECT username FROM users WHERE username = ? AND password = ?";  
    try (Connection connection = dataSource.getConnection();  
        PreparedStatement statement = connection.prepareStatement(query)) {  
        statement.setString(1, username);  
        statement.setString(2, password);  
        ResultSet rs = statement.executeQuery();  
        return rs.next();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return false;  
}
```

Samostalni rad

- Zadatak: Zaštitite dodavanje komentara od SQL injection napada koristeći PreparedStatement
- Vreme: **30 minuta**
- Ukoliko završite ranije, izvršite SQL Injection napad preko pretrage automobila koji dodaje novi automobil u tabelu **cars**

Rešenje zadatka

Demonstracija

Rešenje zadatka

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

Prompt Injection

Prompt Injection

- Napad na Velike jezičke modele – LLM
- Maliciozni unos koji manipuliše GenAI sistemima, čime dolazi do curenja poverljivih informacija, širenja dezinformacija ili gore

Tipovi prompt injection-a

- Direktni prompt injection
- Indirektni prompt injection

Prompt injection vs jailbreaking

- Prompt injection: Koriscenje unosa da se promeni ponasanje
- Jailbreaking: Zaobilazenje zastita

Prevenција i ublažavanje posledica

- Poznate prakse zaštite
- Validacija unosa
- Ograničavanje privilegija
- Čovek kao deo procesa