SQL Injection

SQL Injection (SQLi) nastaje kada aplikacija direktno spaja korisnički unos u SQL upit. Napadač tada ubacuje ili menja upit i navodi bazu da izvrši neočekivane komande.
Varijante: union-based, error-based, boolean/time-based blind, second-order, OOB.

Uticaj

- Čitanje poverljivih podataka (korisnici, lozinke, skriveni proizvodi).
- Izmena ili brisanje podataka.
- Bypass logovanja (OR 1=1).
- Eskalacija privilegija ili čak izvršavanje OS komandi.
- Pravne/finansijske posledice (GDPR, PCI DSS).

Omogućavajuće ranjivosti

- String konkatenacija umesto parametrizacije.
- Slaba validacija ulaza.
- Detaljne poruke o grešci (error leakage).
- Preširoke DB privilegije.
- Dozvoljeni multi-statements.
- Nedostatak sigurnosnog testiranja.

Kontramere

- **Parametrizovani upiti (Prepared Statements)** u svim jezicima/ORM-ovima.
- Validacija i allow-list vrednosti (npr. category ∈ {Gifts, Lifestyle}).
- Minimizacija DB privilegija (princip *least privilege*).
- Generičke poruke o grešci (detalji samo u log).
- Onemogućiti multi-statement gde nije potreban.
- Sekundarne zaštite: WAF, RASP, rate limiting.
- Sigurnosni proces: SAST/DAST, code review, patch-ovanje.

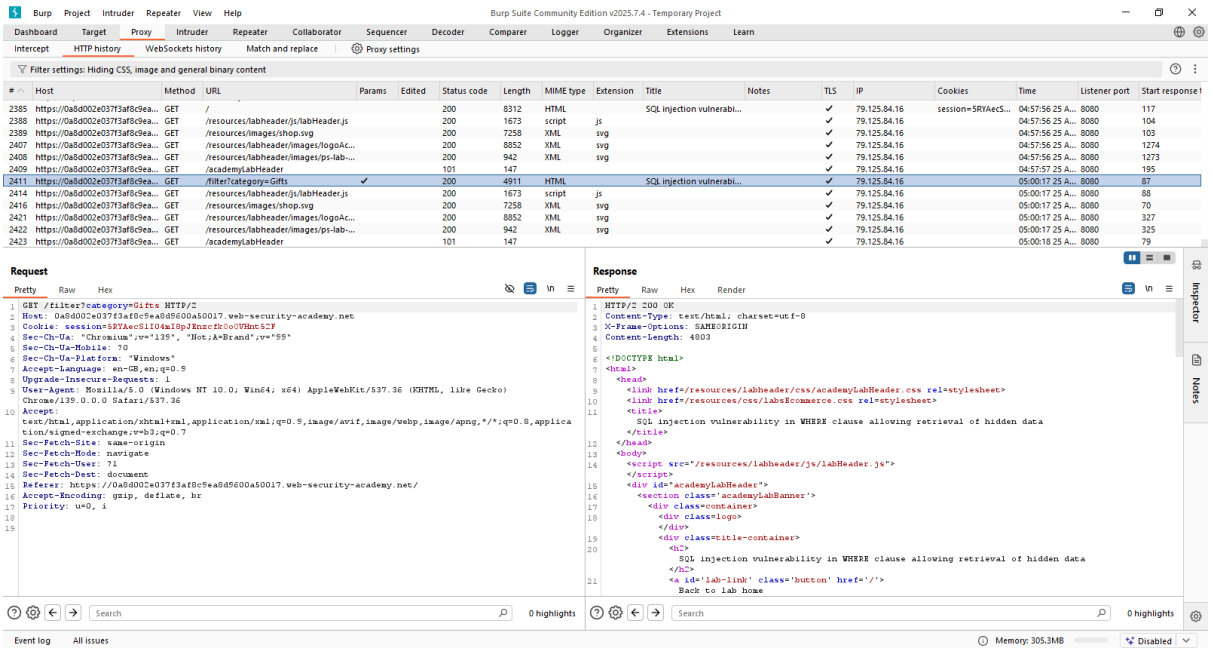
Zadatak 1:

➔ SQL injection vulnerability in WHERE clause allowing retrieval of hidden data
➔ Cilj napada: Eksploatacija ranjivosti u SQL upitu kako bi se prikazali proizvodi koji su označeni kao „unreleased“ (skriveni od običnih korisnika).
➔ Tačka unosa:GET /filter?category=...

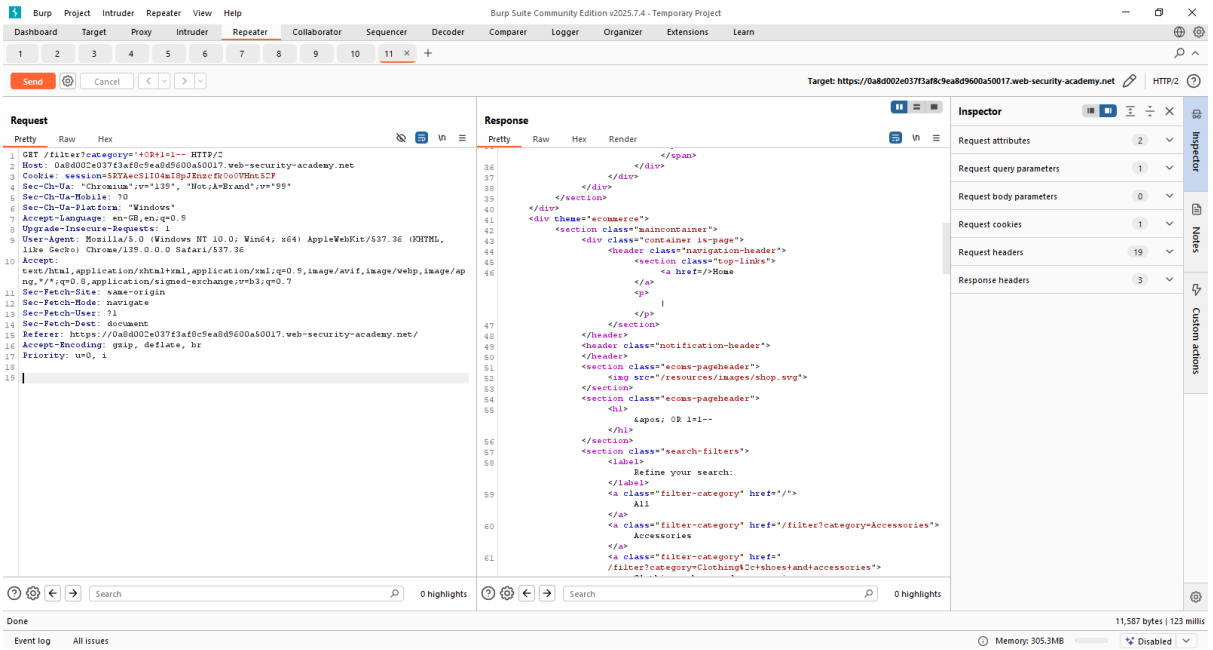
Payload (URL-encoded u praksi): '+OR+1=1--

Koraci (Burp):

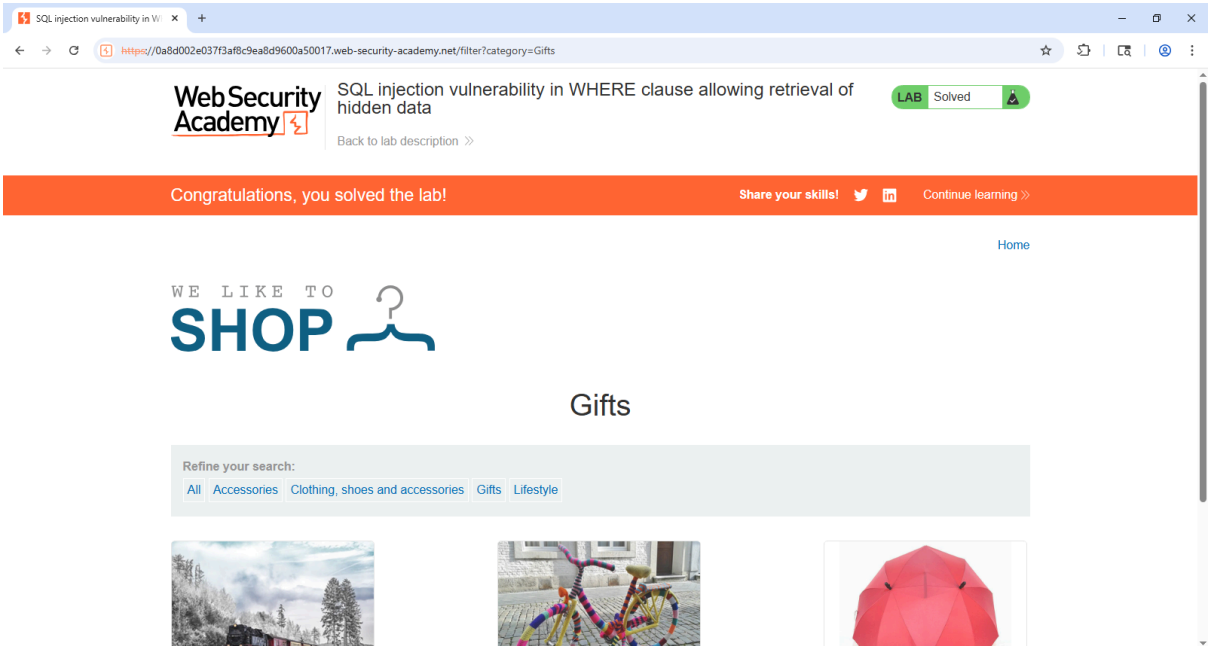
1. Proxy → HTTP history → `GET /filter?category=Gifts` → **Send to Repeater**.



2. U Repeater-u zameniti putanju u: `GET /filter?category='+OR+1=1-- HTTP/2` → **Send**.



3. U response-u se prikazuju i „unreleased“ stavke. Lab solved.

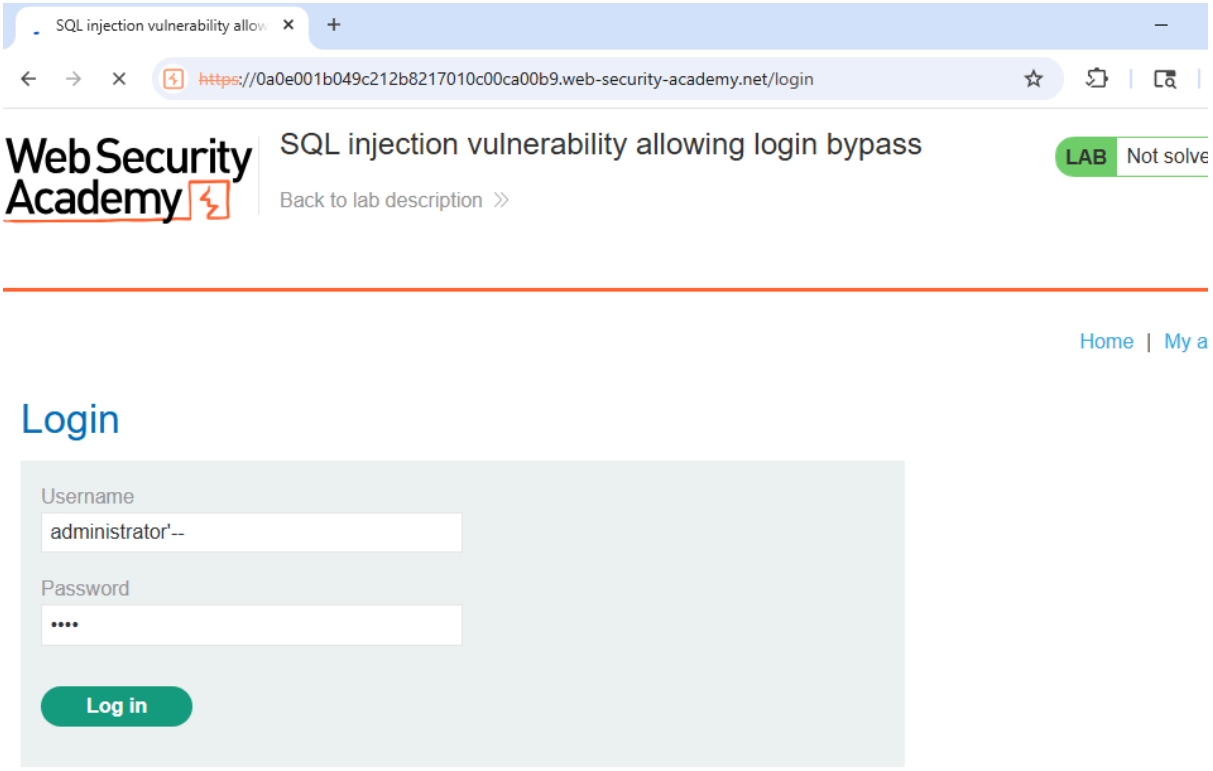


Zadatak 2:

Lab vežba je imala SQL injection ranjivost u login funkciji.
Cilj je bio da se iskoristi SQL injection kako bi se zaobišla provera lozinke i izvršila prijava kao **administrator** korisnik.

Postupak rešavanja

1. **Pokretanje Burp Suite-a i otvaranje lab aplikacije**
    - Pokrenula sam Burp Suite i otvorila lab kroz njegov integrisani browser kako bi sav saobraćaj mogao da se presretne.
2. **Presretanje HTTP zahteva**
    - Na stranici za login unela sam sledeće podatke:
        - **Username:** `administrator'--`
        - **Password:** bilo šta (npr. `test`)
    - Klikom na **Login** zahtev je presretnut u Burp Proxy-ju.



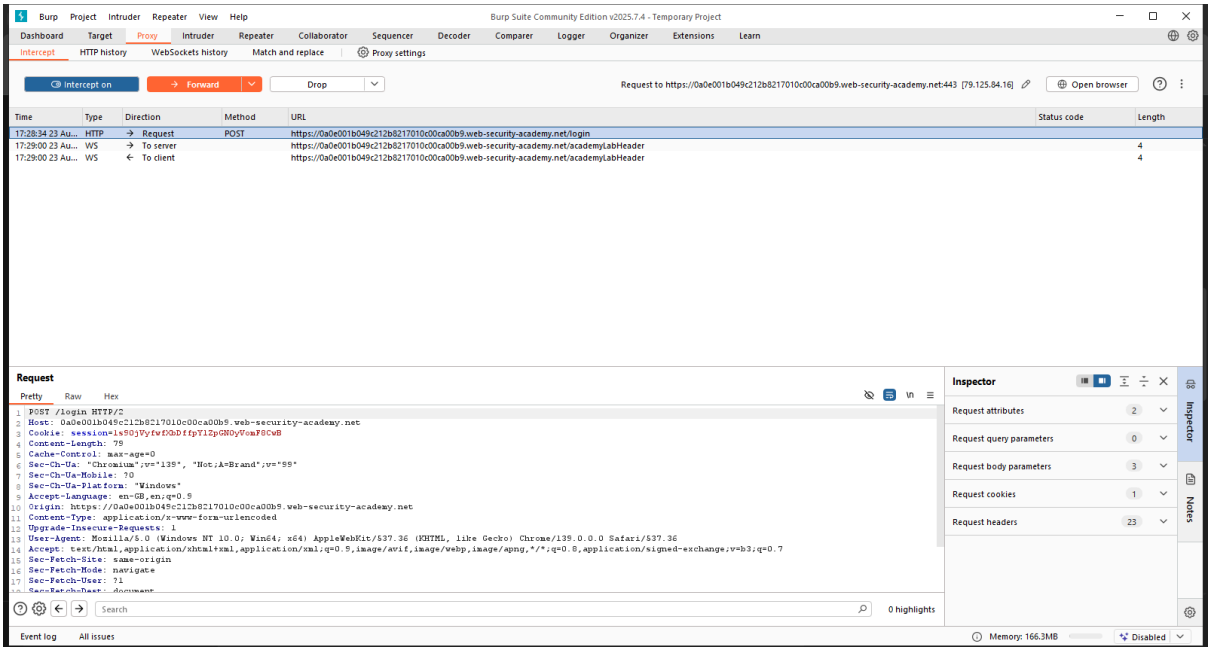
3. **Analiza i izmena zahteva**

U telu POST zahteva nalazilo se:
`username=administrator'--&password=test`

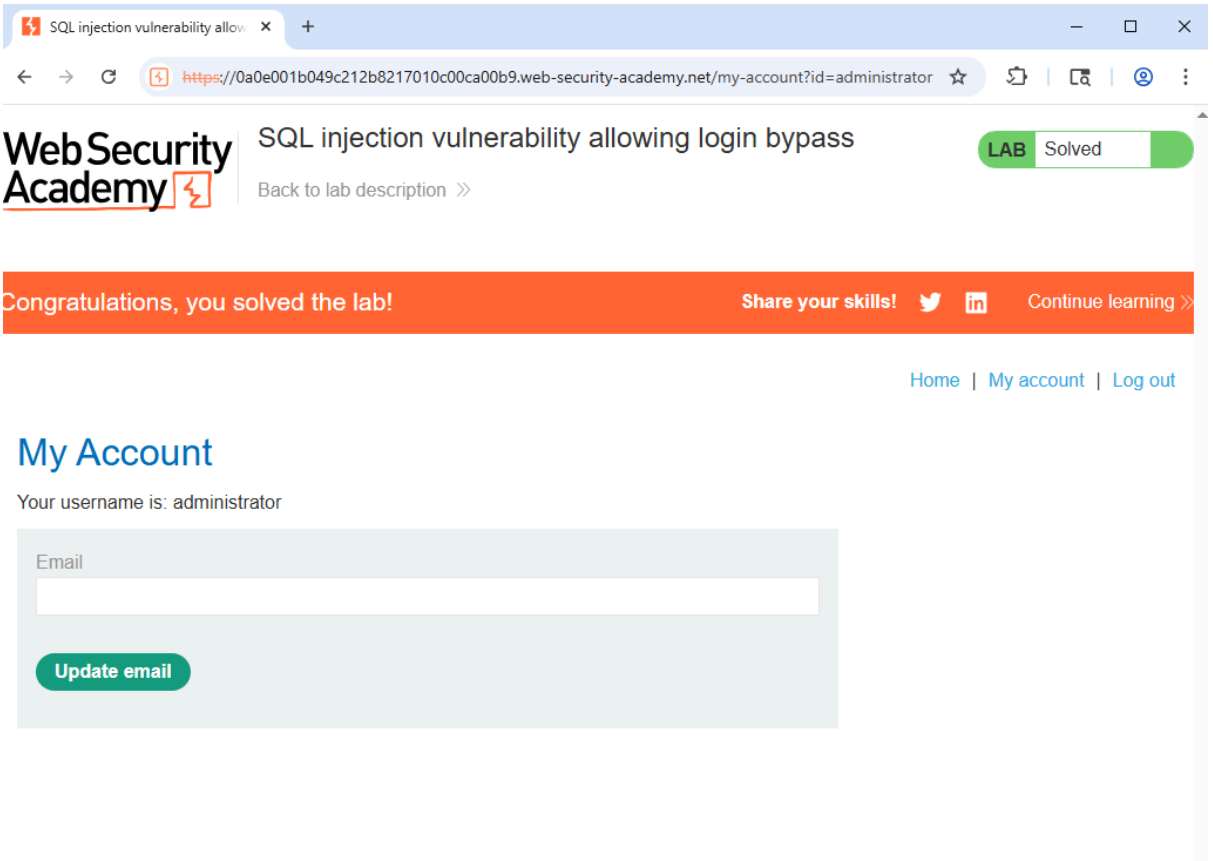
Ovaj unos zatvara string za korisničko ime i koristi SQL komentar `--` kako bi se zanemario ostatak upita koji se odnosi na lozinku.

4. **Slanje zahteva i izvršavanje injekcije**

    - Prosledila sam zahtev (Forward) i nakon toga je aplikacija obradila injekciju.



Pošto je ostatak uslova ignorisan, upit je proverio samo da li postoji korisnik `administrator`, što je dovelo do uspešne prijave.



Tipičan SQL upit za login izgleda ovako:
`SELECT * FROM users WHERE username = '<unos>' AND password = '<unos>';`

Kada se ubaci payload `administrator'--`, upit postaje:

```
SELECT * FROM users WHERE username = 'administrator'-- ' AND password = 'test';
```

Zahvaljujući komentaru `--`, deo sa lozinkom se ignoriše.
Na ovaj način je moguće ući u sistem kao administrator bez potrebe za lozinkom.

Rezultat

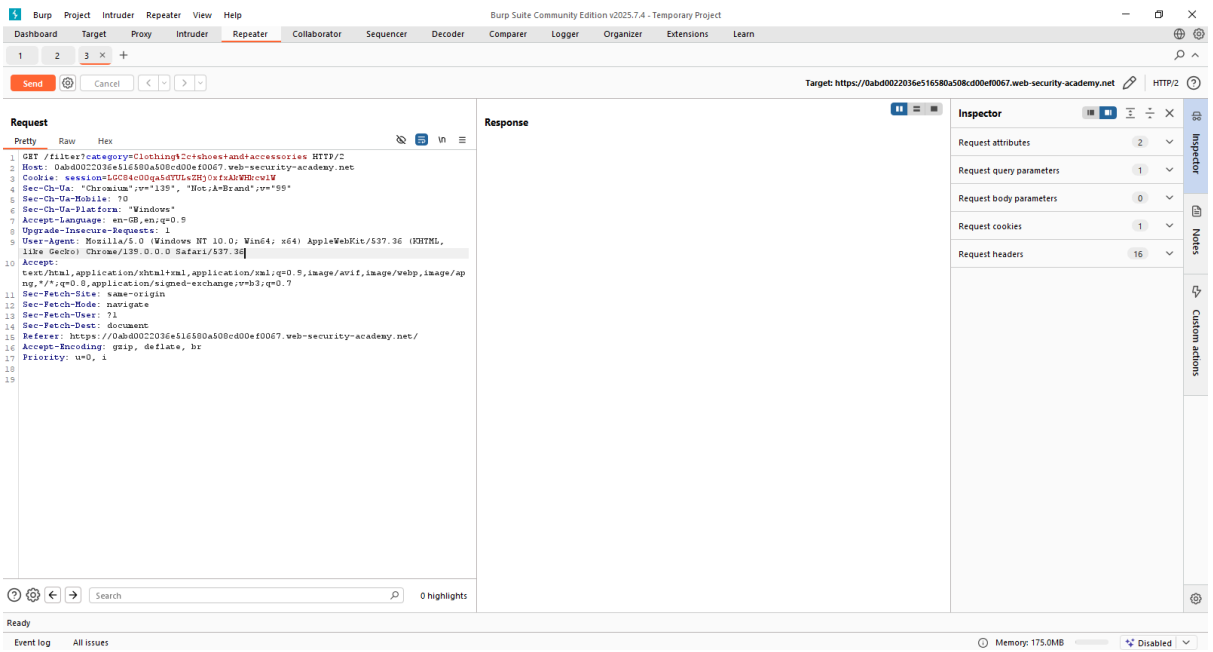
Korisnik je uspešno prijavljen kao **administrator**, čime je zadatak rešen.
Lab je prepoznao uspešno iskorišćenu ranjivost i obeležio vežbu kao *Solved*.

Zadatak 3

Lab vežba sadrži SQL injection ranjivost u filteru kategorija proizvoda.
Cilj je bio da se koristeći **UNION SQL injection** prikaže verzija baze podataka Oracle.

Postupak:

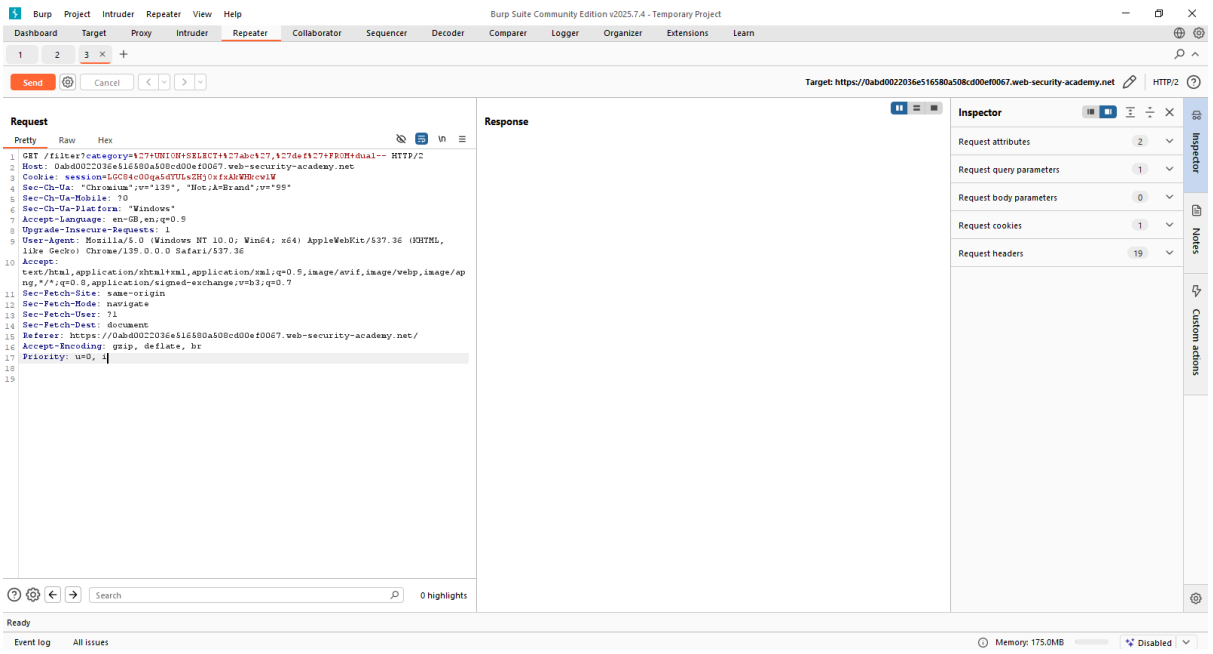
1. **Pokretanje Burp Suite-a i presretanje zahteva**
    - Otvorila sam Burp Suite i lab u njegovom ugrađenom browseru.
    - Klikom na bilo koju kategoriju proizvoda, HTTP zahtev za filtriranje je presretnut u Proxy → Intercept.
    - Zahtev je prosleđen u Repeater za dalju manipulaciju.



2. **Potvrda ranjivosti i broj kolona**

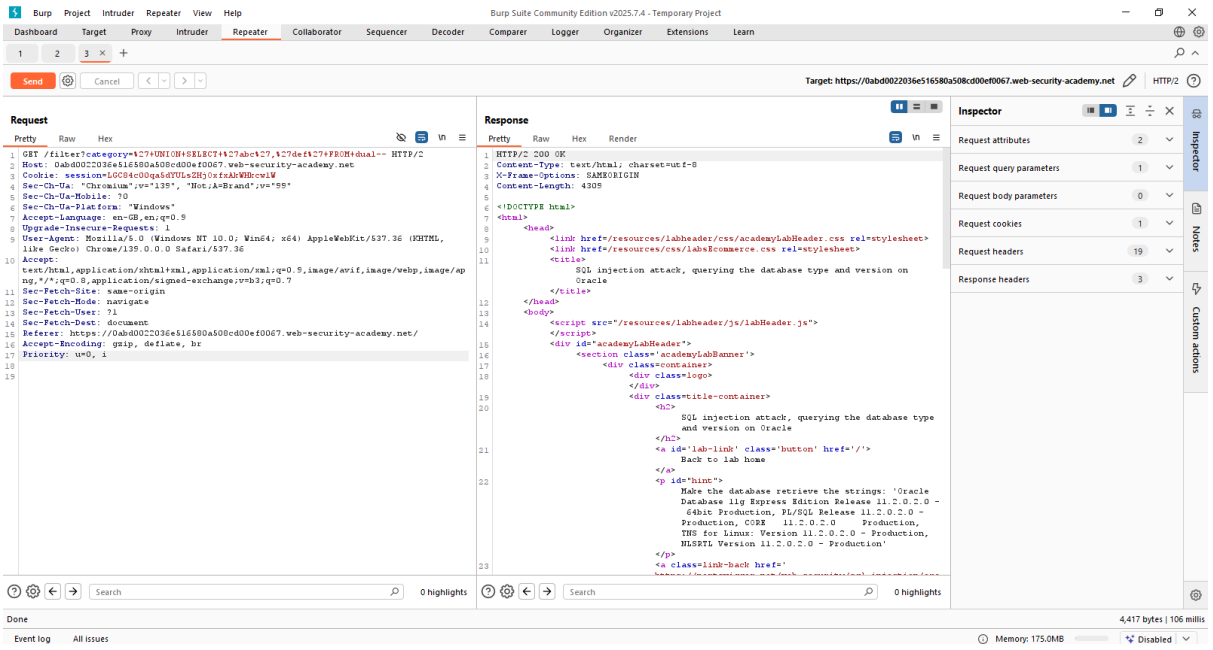
U Repeater-u izmenila sam parametar `category` koristeći payload:

```
'+UNION+SELECT+'abc','def'+FROM+dual--
```



Po slanju zahteva, stranica je prikazala dve vrednosti “abc” i “def”.
Time je potvrđeno da:

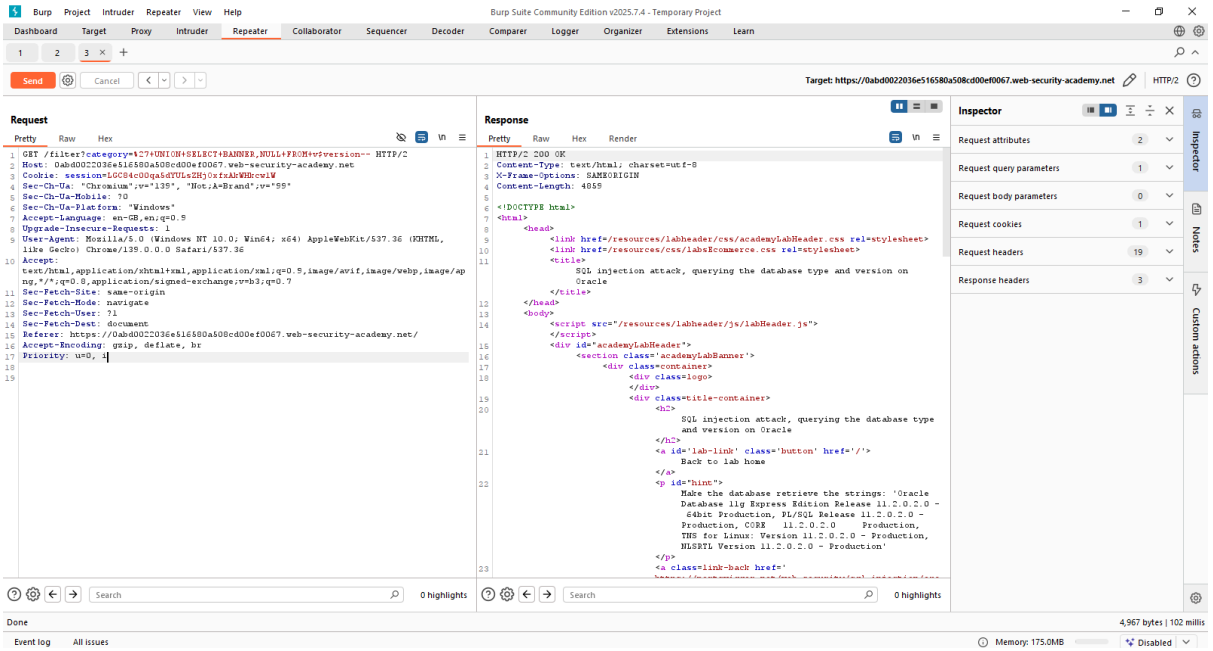
- aplikacija prihvata SQL injection,
- upit vraća **dve kolone**, obe sposobne za prikaz tekstualnih podataka.



3. **Prikaz verzije baze podataka**

Nakon toga, u istom Repeater-u sam koristila payload:

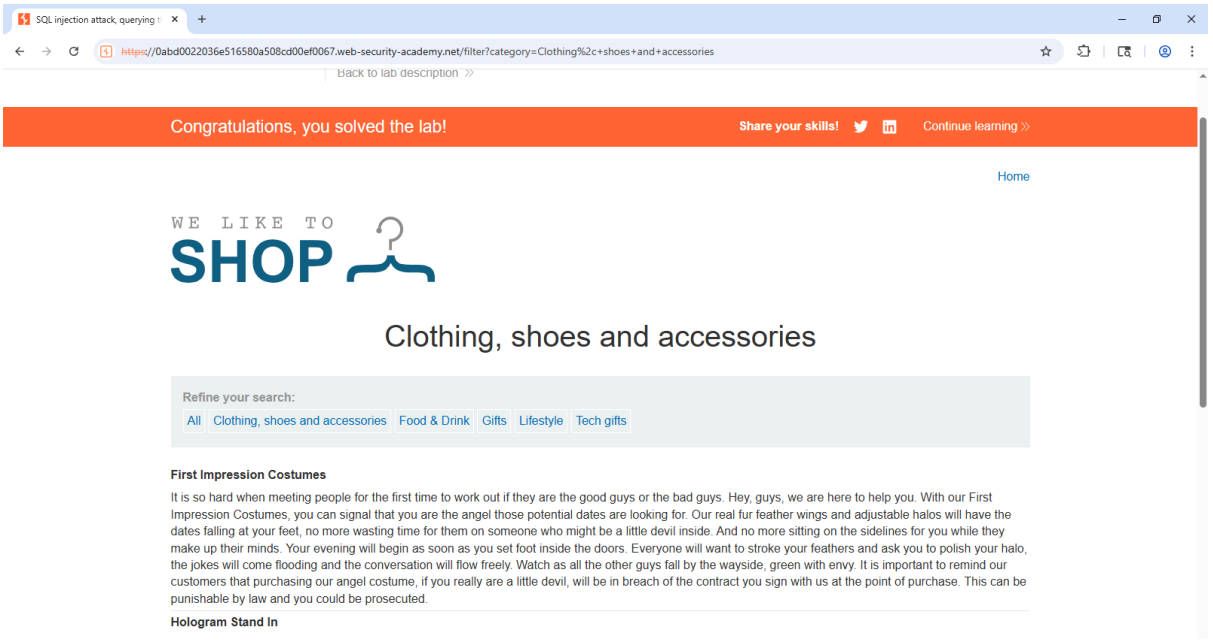
```
'+UNION+SELECT+BANNER,NULL+FROM+v$version--
```



Klikom na **Send**, stranica je prikazala verziju baze ((11g Express Edition Release 11.2.0.2.0 - 64bit Production).

```
                    Tech gifts
                </a>
            </section>
            <table class="is-table-longdescription">
                <tbody>
                    <tr>
                        <th>
                            CORE 11.2.0.2.0     Production
                        </th>
                    </tr>
                    <tr>
                        <th>
                            NLSRTL Version 11.2.0.2.0 - Production
                        </th>
                    </tr>
                    <tr>
                        <th>
                            Oracle Database 11g Express Edition Release
                            11.2.0.2.0 - 64bit Production
                        </th>
                    </tr>
                    <tr>
                        <th>
                            PL/SQL Release 11.2.0.2.0 - Production
                        </th>
                    </tr>
                    <tr>
                        <th>
                            TNS for Linux: Version 11.2.0.2.0 - Production
                        </th>
                    </tr>
                </tbody>
            </table>
        </div>
    </section>
```

Lab je prepoznao uspešno iskorišćenu ranjivost i označio vežbu kao *Solved*.



Rezultat

Korišćenjem SQL injection-a putem **UNION SELECT**, uspešno je prikazano:

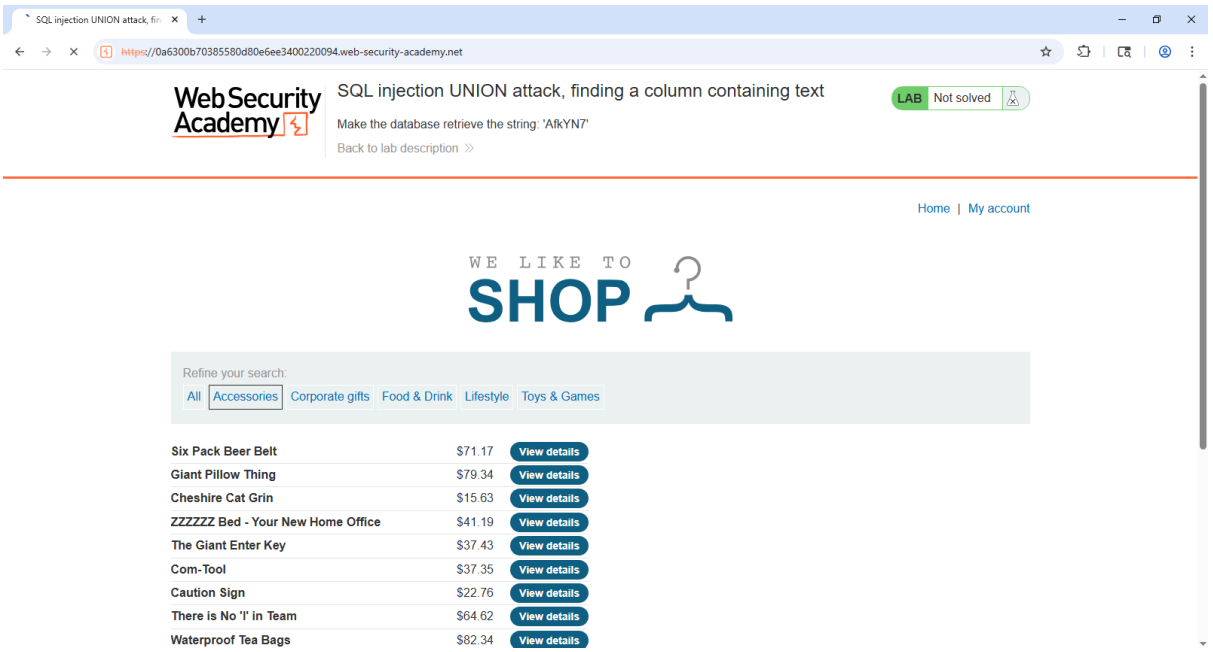
- tekstualni test (`abc/def`) za identifikaciju kolona,
- verzija Oracle baze podataka, što je zadatak uspešno rešilo.

Zadatak 4

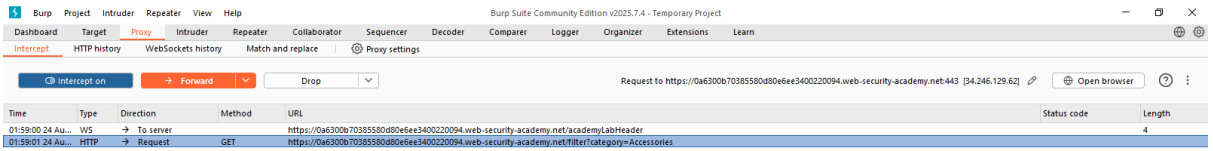
Lab vežba sadrži SQL injection ranjivost u filteru kategorija proizvoda.
Cilj je bio da se koristeći **UNION SQL injection** ubaci dodatni red sa **vrednošću koju lab pruža**, čime se potvrđuje koje kolone mogu prikazivati tekstualne podatke.

Postupak rešavanja

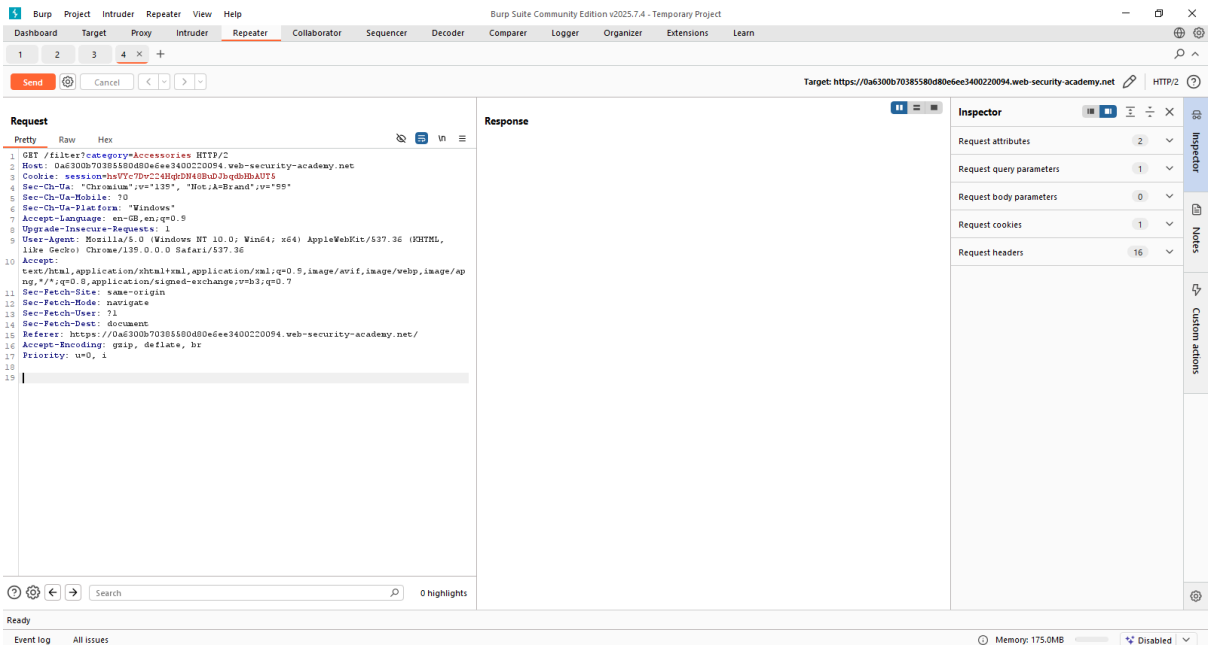
1. **Pokretanje Burp Suite-a i presretanje zahteva**
    - Otvorila sam lab u ugrađenom browseru Burp Suite-a.



    - Klikom na kategoriju proizvoda, HTTP zahtev za filtriranje je presretnut u Proxy → Intercept.



- Zahtev je prosleđen u **Repeater** za dalju manipulaciju.

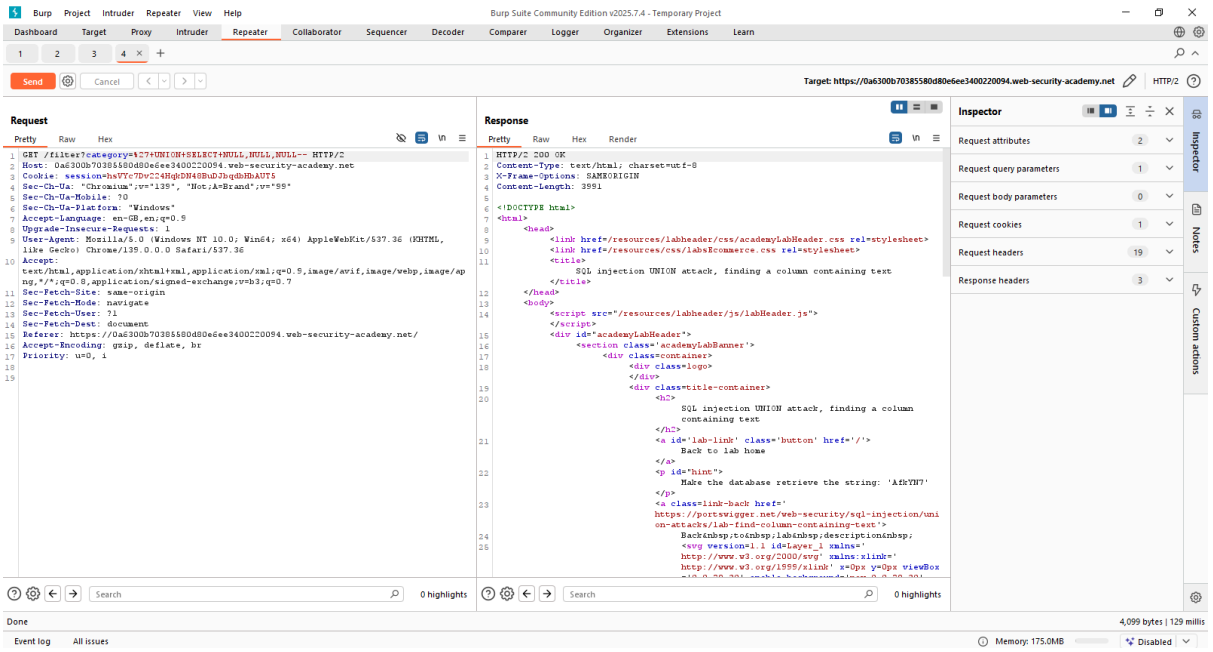


2. Određivanje broja kolona

U Repeater-u sam parametar `category` izmenila payloadom:

```
'+UNION+SELECT+NULL,NULL,NULL--
```

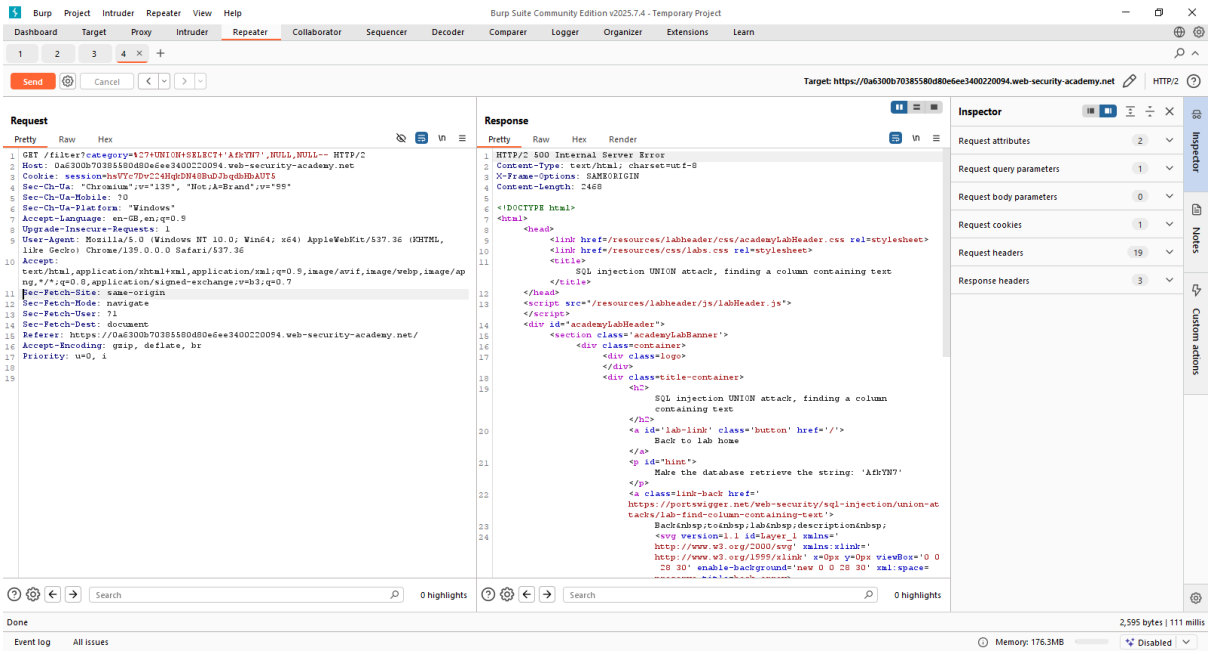
Po slanju zahteva, upit je prošao bez greške, što je potvrdilo da upit vraća **3 kolone**.



3. **Identifikacija kolone kompatibilne sa tekstom**

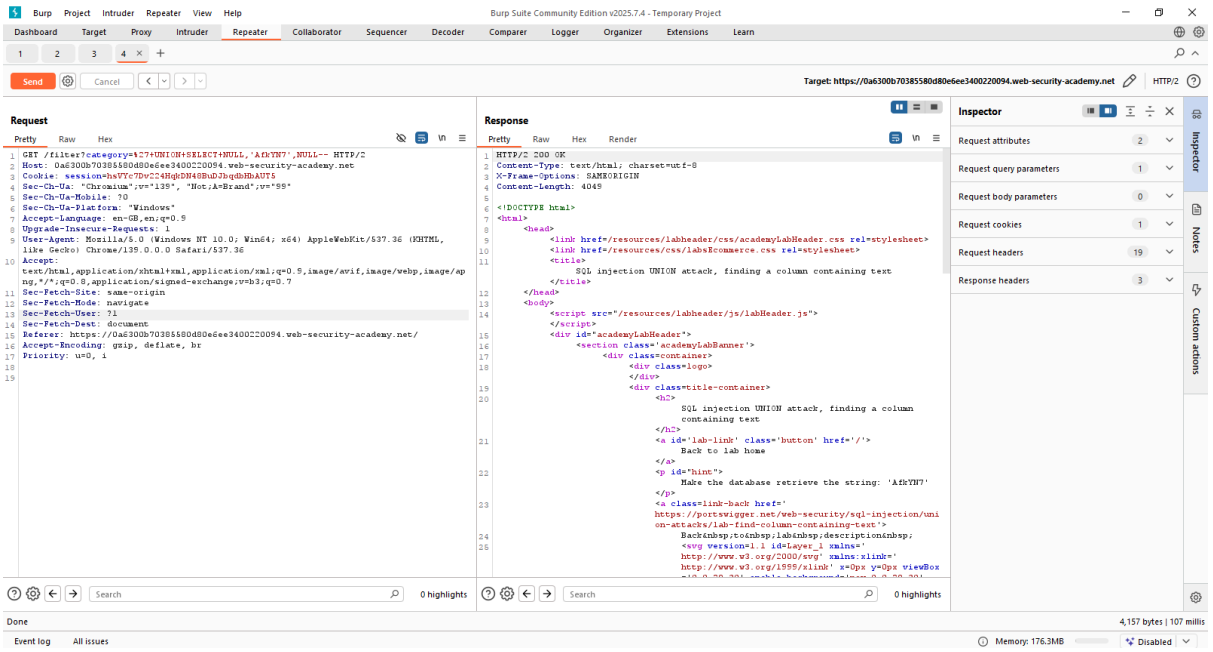
Random vrednost koju lab pruža (npr. `abcdef`) sam probala u prvoj koloni:

```
'+UNION+SELECT+'abcdef',NULL,NULL--
```



Po slanju zahteva, dobila sam grešku → vrednost sam probala u drugoj koloni:

```
'+UNION+SELECT+NULL,'abcdef',NULL--
```

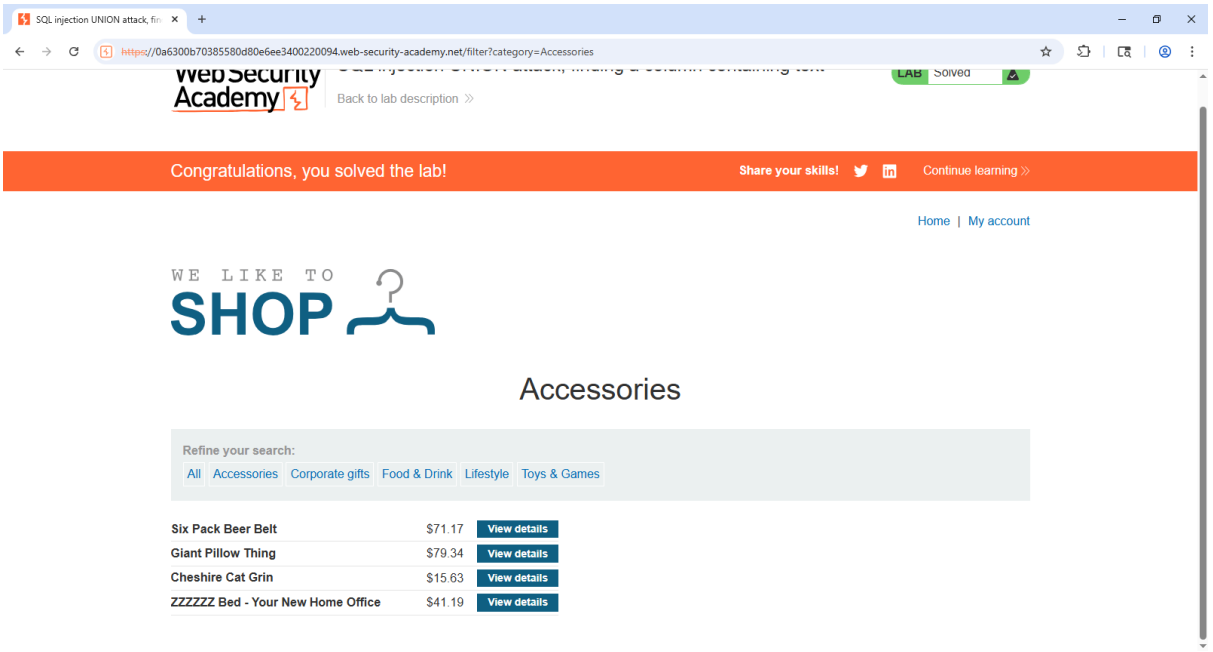


- Ovaj put vrednost se pojavila u rezultatu HTML stranice, što je pokazalo da je **druga kolona kompatibilna sa tekstom**.

4. **Ubacivanje vrednosti i rešavanje lab-a**

- Kada je identifikovana prava kolona, upit je uspešno ubacio dodatni red sa lab-om datom vrednošću.

- Lab je prepoznao uspešno iskorišćenu ranjivost i označio vežbu kao *Solved*.



Korišćenjem **UNION SQL injection**:

- uspešno je identifikovan broj kolona (3),
- pronađena je kolona kompatibilna sa tekstom,
- dodatni red sa lab-om datom vrednošću je ubačen, čime je zadatak uspešno rešen.

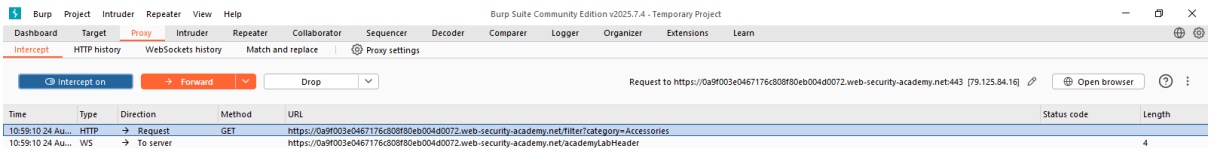
Zadatak 5: SQL Injection – prikazivanje verzije baze (UNION attack)

Lab opis:
Ovaj lab sadrži ranjivost na **SQL injection** u filteru kategorija proizvoda. Cilj je bio izvršiti **UNION-based SQL injection** kako bi se prikazala verzija baze podataka.

1. Identifikacija ranjivosti

- Otvorila sam stranicu i pregledala filter za kategorije proizvoda.
- Parametar `category` u URL-u (`/filter?category=Accessories`) potencijalno može biti ranjiv.
- Pošaljala sam GET zahtev kroz Burp Suite Proxy da ga presretnem.

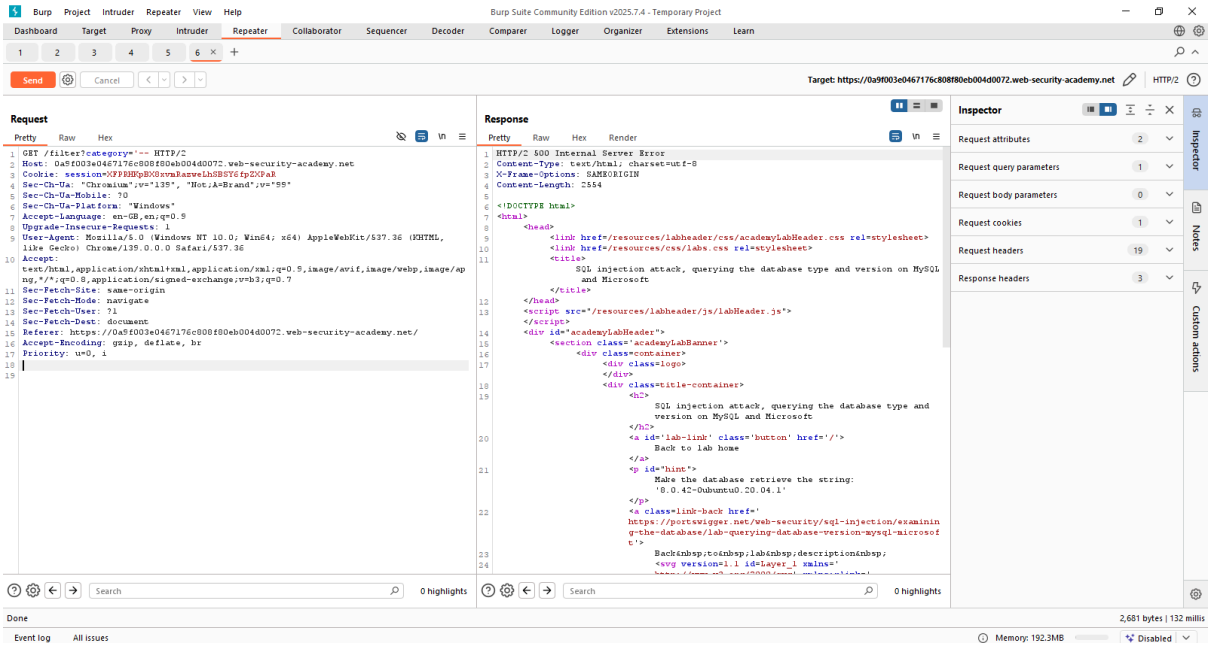
Presretanje zahteva u Burp Proxy-u.



2. Testiranje ranjivosti

- Testirala sam prosti payload za SQL injection: `' OR '1'='1`

- Stranica je prikazala sve proizvode, što je potvrdilo ranjivost parametra.

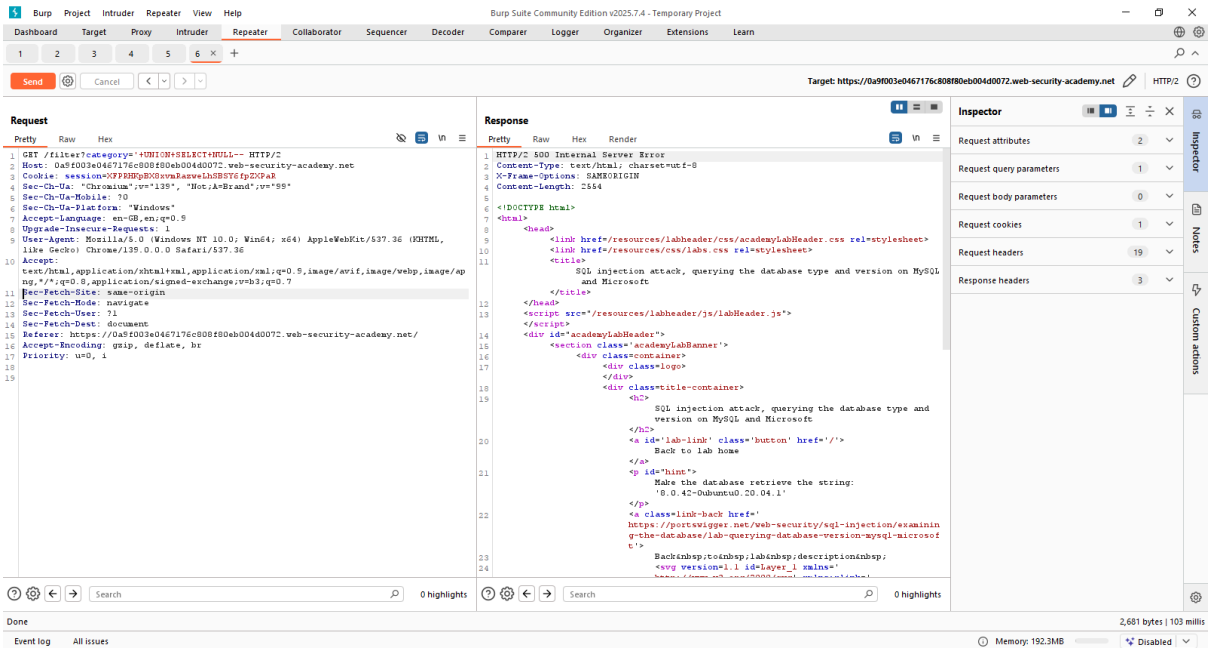


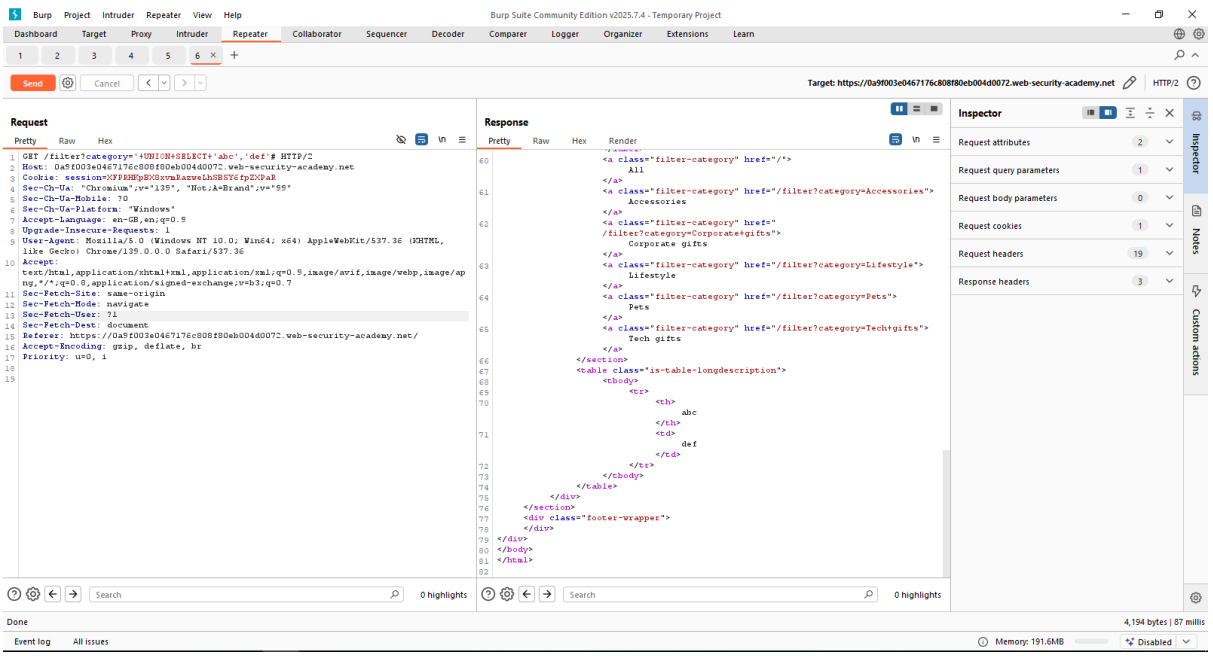
3. Određivanje broja kolona

Koristila sam **UNION SELECT test** da bih proverila koliko kolona vraća upit.

`'+UNION+SELECT+'abc','def'#`

- Stranica je prikazala `'abc'` i `'def'` u rezultatima.
- Zaključak: upit vraća **2 kolone**, obe su tekstualnog tipa.





4. Prikaz verzije baze

S obzirom na broj kolona, injektovala sam payload za prikaz verzije baze: `'+UNION+SELECT+@@version,+NULL#`

Stranica je prikazala verziju baze: `8.0.42-0ubuntu0.20.04.1`

```
                        Pets
                    </a>
65                  <a class="filter-category" href="/filter?category=Tech+gifts">
                        Tech gifts
                    </a>
66              </section>
67              <table class="is-table-longdescription">
68                  <tbody>
69                      <tr>
70                          <th>
                                8.0.42-0ubuntu0.20.04.1
                            </th>
71                      </tr>
72                  </tbody>
73              </table>
74          </div>
75      </section>
76      <div class="footer-wrapper">
77      </div>
78  </div>
79  </body>
80  </html>
81
```

Parametar `category` je ranjiv na **UNION-based SQL injection**. Uspešno sam prikazala verziju baze MySQL: `8.0.42-0ubuntu0.20.04.1`. Lab je uspešno rešen.