[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.

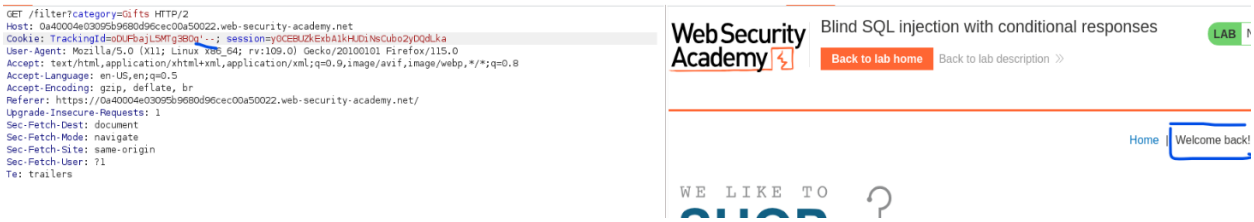
The database contains a different table called `users`, with columns called `username` and `password`. You need to exploit the blind SQL injection vulnerability to find out the password of the `administrator` user.

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 fin this user "***users***" and what is 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

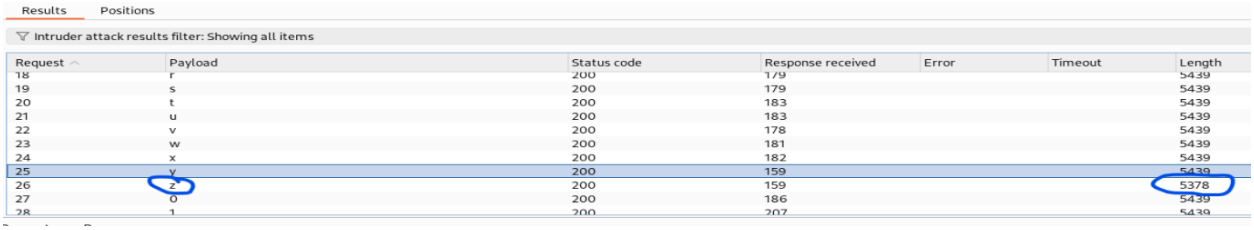
Now we can see that we're working with one column! As the lab said, we have a "***users***" table and we should have a "***password***" column, and the username that we want to query is "***administrator***", with all this information, what I did was just query how long is the "***administrator***" password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

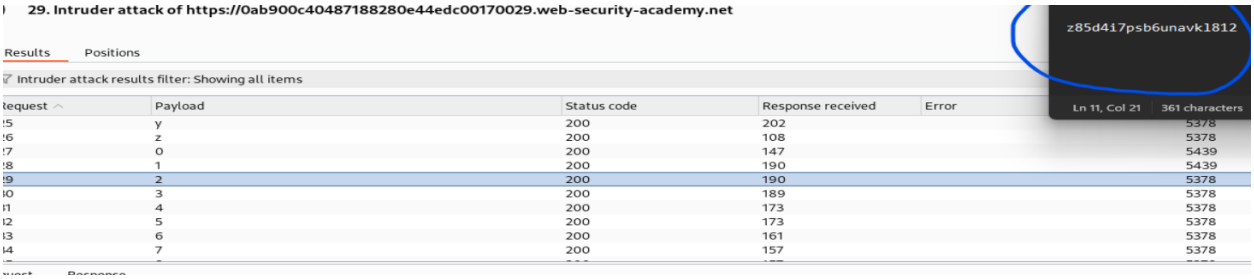
In order to find how long is the account password, I did a trial and error using "***LENGTH***" which can be used to check the length of the string.

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)

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





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