Task 13: Secure API Testing & Authorization Validation

Target Application: http://testphp.vulnweb.com/
Testing Tool: cURL

---

1. Objective

The objective of this task is to perform security testing on a web-based application using cURL in order to identify issues related to authentication, authorization, input validation, rate limiting, and error handling. The identified vulnerabilities are mapped to OWASP API Security risks.

---

2. Tools Used

- **Primary Tool:** cURL
- **Operating Environment:** Linux / Windows (Git Bash)

---

3. Target Description

testphp.vulnweb.com is a deliberately vulnerable web application provided for security testing and learning purposes. It simulates insecure backend behavior commonly found in poorly designed APIs and web services.

---

4. Methodology

The following security checks were performed:

- Authentication testing
- Authorization testing
- Input validation testing
- Rate limiting testing
- HTTP response and error handling review

5. Testing & Observations

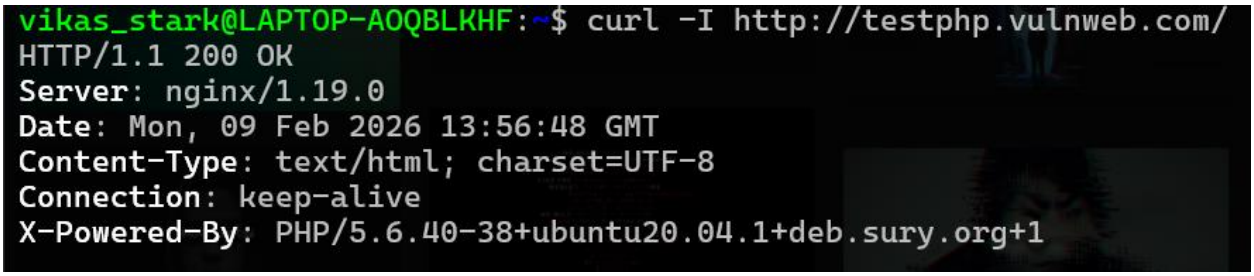
5.1 Connectivity & Reconnaissance

Command:

curl -I http://testphp.vulnweb.com/

Observation:
The server responds with HTTP headers indicating that the application is accessible.

1: HTTP header response of target application



5.2 Authentication Testing

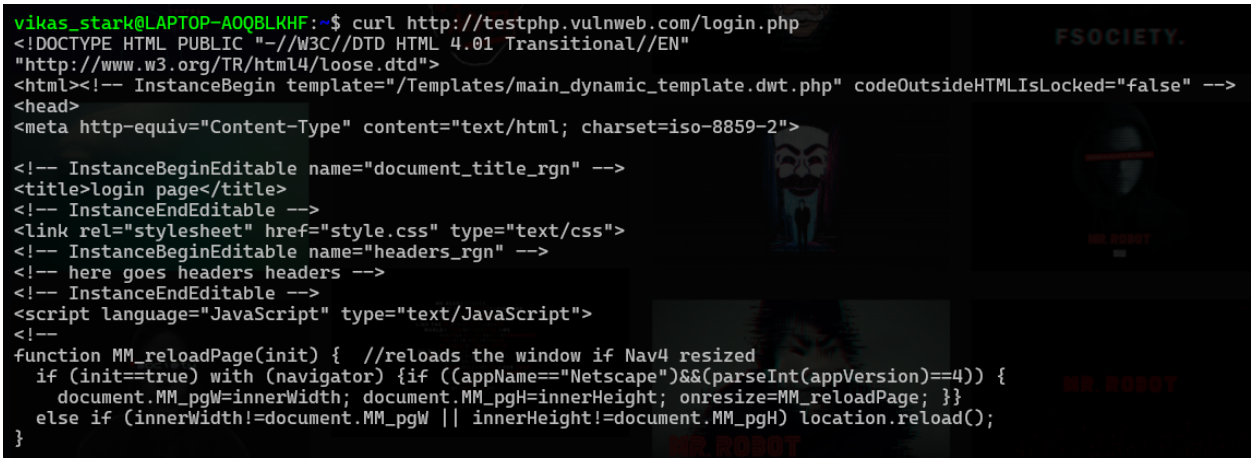
5.2.1 Access Login Page Without Authentication

Command:

curl http://testphp.vulnweb.com/login.php

Observation:
Login page is accessible without any authentication token or session.

2: Login page accessible without authentication



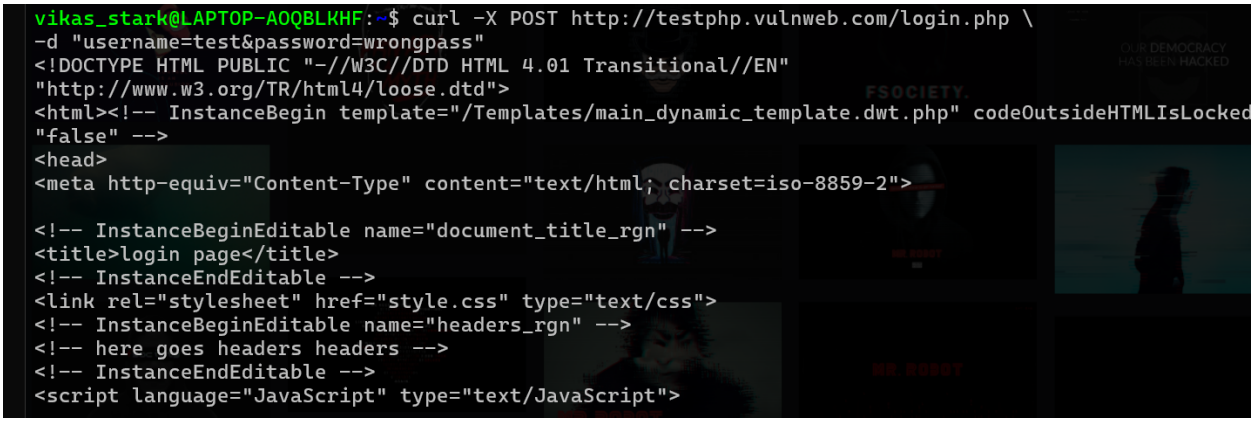
5.2.2 Attempt Login with Invalid Credentials

Command:

curl -X POST http://testphp.vulnweb.com/login.php \

-d "username=test&password=wrongpass"

Observation:
Application processes the request but does not implement secure token-based authentication.



5.3 Authorization Testing (Broken Authorization)

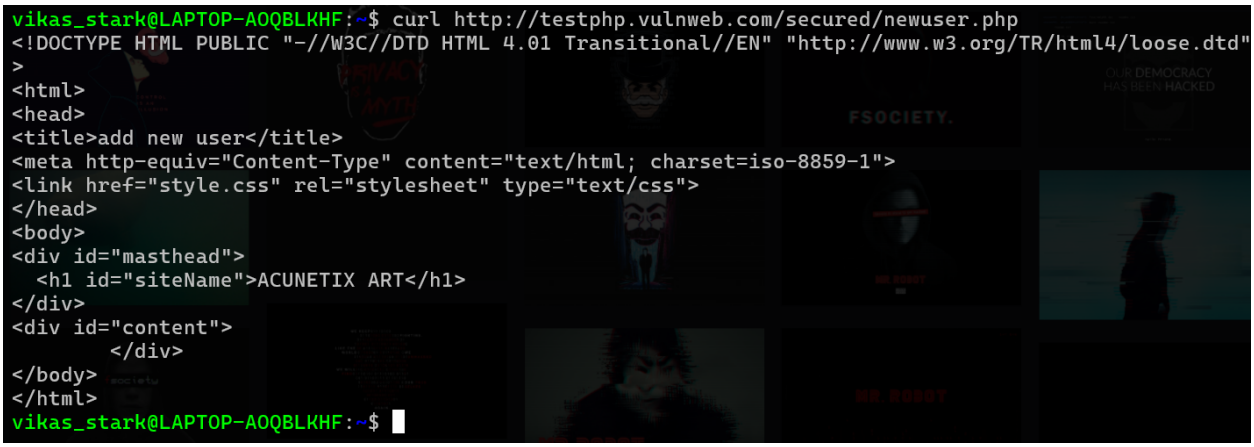
5.3.1 Access Potentially Restricted Page Without Login

Command:

curl http://testphp.vulnweb.com/secured/newuser.php

Observation:
Restricted functionality is accessible without verifying user authorization.

4: Unauthorized access to restricted page



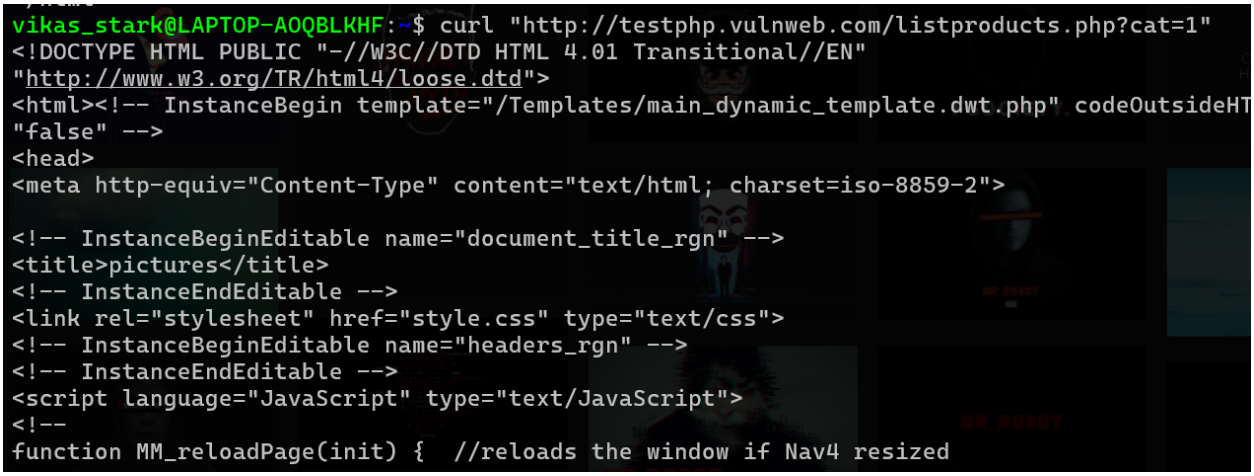
5.4 Input Validation Testing

5.4.1 Normal Request

Command:

curl "http://testphp.vulnweb.com/listproducts.php?cat=1"

5: Normal application response



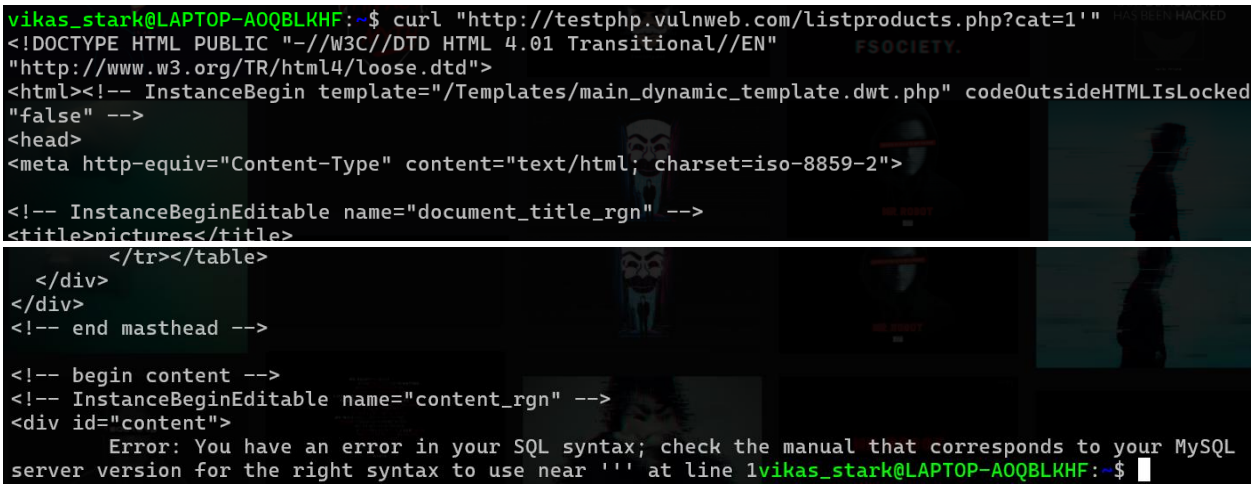
5.4.2 Malformed Input (SQL Injection Test)

Command:

curl "http://testphp.vulnweb.com/listproducts.php?cat=1'"

Observation:
Application behavior changes when malformed input is supplied, indicating lack of input validation and possible SQL injection vulnerability.

6: Application response to malformed input



5.5 Rate Limiting Testing

Command:

for i in {1..20}; do

curl -s -o /dev/null -w "%{http_code}\n" \

http://testphp.vulnweb.com/

done

Observation:
All requests return HTTP 200. No rate limiting or throttling is enforced.

7: Multiple rapid requests showing no rate limiting

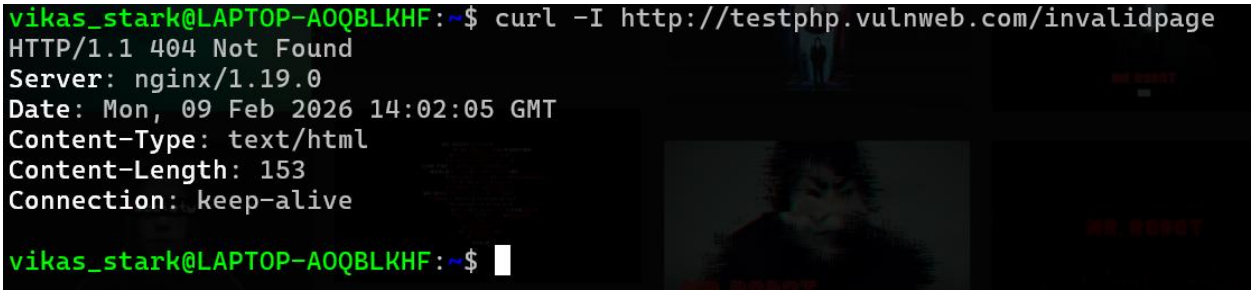
5.6 Error Handling & HTTP Response Review

Command:

curl -I http://testphp.vulnweb.com/invalidpage

Observation:
Server returns error responses that may disclose implementation details.

8: Error response and status code



6. Identified Vulnerabilities & OWASP Mapping

Vulnerability	Description	OWASP API Risk
Missing Authentication	No token/session enforcement	API2: Broken Authentication
Broken Authorization	Restricted pages accessible	API1: Broken Object Level Authorization
SQL Injection	Unsanitized input accepted	API1: Injection
No Rate Limiting	Unlimited requests allowed	API4: Unrestricted Resource Consumption
Poor Error Handling	Verbose responses	API8: Security Misconfiguration

Mapped according to **OWASP API Security Top 10**.

---

7. Final Outcome

- Successfully performed security testing using cURL
- Identified authentication and authorization flaws
- Detected input validation weaknesses
- Confirmed absence of rate limiting controls
- Mapped vulnerabilities to OWASP API Security risks