Task 14: Secure API Testing & Authorization Validation

🔹 **Objective**

To perform security testing on a REST API to evaluate:

- Authentication mechanisms
- Authorization controls
- Input validation
- Rate limiting
- Response handling
- OWASP API Security risks

🔹 **Tools Used**

- **Primary Tool:** Postman
- **Alternative Tools:** cURL, Insomnia

🔐 **API Security Testing Report**

1️⃣ **Understanding REST APIs**

REST APIs use HTTP methods to interact with backend services:

HTTP Method	Purpose	Security Concern
GET	Retrieve data	Unauthorized data access
POST	Create resource	Input validation
PUT	Update resource	Ownership validation
DELETE	Delete resource	Proper authorization

2️⃣ **Testing Methodology**

Step 1: API Configuration in Postman

- Set endpoint URL (e.g., https://example.com/api/users)
- Added required headers:
    - Authorization: Bearer <token>
    - Content-Type: application/json
- Configured request body according to API documentation.

Step 2: Authentication Testing

✓ Sent request with **valid credentials**
→ Expected: 200 OK

❌ Sent request with **invalid credentials**
→ Expected: 401 Unauthorized

❌ Removed authentication header
→ Checked if API still allowed access

Observation:
If access is granted without authentication → Authentication Bypass vulnerability.

Step 3: Authorization Testing (Broken Authorization)

Modified resource ID:

GET /api/users/101

Changed to:

GET /api/users/102

If a user can access another user's data → **Broken Object Level Authorization (BOLA)**.

Step 4: Input Validation Testing

Sent malformed inputs:

{

"email": "invalid-email",

"age": -50,

"role": "<script>alert(1)</script>"

}

Checked for:

- SQL errors
- Stack traces
- Server crashes
- Unhandled exceptions

Step 5: Rate Limiting Test

- Sent multiple rapid requests using Postman Runner or cURL loop.
- Observed if API blocks excessive requests.

If no limit is enforced → Rate limiting vulnerability.

Step 6: HTTP Response Analysis

Reviewed:

- Status codes (200, 401, 403, 500)
- Detailed error messages
- Exposure of internal system data
- Debug information

🚨 **Identified Vulnerabilities (Mapped to OWASP API Top 10)**

Vulnerability	OWASP API Risk	Impact	Severity
Broken Object Level Authorization	API1	Data exposure	High

Vulnerability	OWASP API Risk	Impact	Severity
Broken Authentication	API2	Account takeover	Critical
Excessive Data Exposure	API3	Sensitive info leak	High
Lack of Rate Limiting	API4	Brute force/DoS	Medium
Security Misconfiguration	API7	Information disclosure	Medium
Improper Input Validation	API8	Injection attacks	High

🛡 **Recommendations**

1. Implement strict role-based access control (RBAC).
2. Validate JWT tokens and enforce expiration.
3. Apply rate limiting (e.g., 100 requests/minute per user).
4. Implement input validation and sanitization.
5. Remove sensitive fields from API responses.
6. Disable debug mode in production.