

Task 13: Secure API Testing & Authorization Validation

◆ 1. Understanding REST APIs (Short Explanation)

A REST API allows communication between client and server using HTTP methods:

Method Purpose

GET Retrieve data

POST Create new data

PUT Update existing data

DELETE Remove data

Example:

GET /api/users

POST /api/login

APIs typically use:

- JSON format
- Authentication tokens (JWT, API Keys, OAuth)
- HTTP status codes (200, 401, 403, 500 etc.)

◆ 2. API Configuration in Postman

Steps:

1. Open Postman
2. Select method (GET/POST/etc.)
3. Enter API endpoint (e.g., `http://example.com/api/login`)
4. Add Headers:
 - Content-Type: `application/json`
 - Authorization: `Bearer <token>`

5. Add Body (if POST/PUT):

- Select raw → JSON
- Example:

```
{  
  "username": "admin",  
  "password": "admin123"  
}
```

6. Click Send

◆ **3. Authentication Testing**

✓ **Valid Credentials Test**

- Send correct login credentials
- Expect: 200 OK + token returned

✗ **Invalid Credentials Test**

- Send wrong password
- Expect: 401 Unauthorized

🔍 Vulnerability if:

- API returns 200 with wrong password
 - Detailed error reveals internal logic
-

◆ **4. Unauthenticated Access Test**

Step:

1. Remove Authorization header
2. Send request to protected endpoint

Expected:

- 401 Unauthorized or 403 Forbidden

⚠ Vulnerability if:

- API still returns sensitive data without authentication

OWASP Mapping:

→ API2: Broken Authentication

◆ 5. Broken Authorization (IDOR Testing)

Modify resource identifiers:

Example:

GET /api/users/101

Change to:

GET /api/users/102

If user 101 can access user 102's data → Vulnerability found.

This is called:

● Broken Object Level Authorization (BOLA)

OWASP Mapping:

→ API1: Broken Object Level Authorization

◆ 6. Input Validation Testing

Send malformed inputs:

Examples:

```
{  
  "email": "' OR 1=1 --"  
}
```

Or:

```
{  
  "age": -9999  
}
```

Check for:

- SQL errors
- Stack traces
- Server crash
- Improper validation

OWASP Mapping:

→ API8: Injection

→ API4: Lack of Resources & Rate Limiting (if server crashes)

◆ 7. Rate Limiting Test

Using Postman Runner or cURL:

Send 100+ rapid login requests.

Expected:

- 429 Too Many Requests
- Temporary block

⚠ Vulnerability if:

- Unlimited attempts allowed
- No lockout mechanism

OWASP Mapping:

→ API4: Unrestricted Resource Consumption

◆ 8. Response Code Analysis

Code Meaning

200 Success

201 Created

400 Bad Request

Code Meaning

401 Unauthorized

403 Forbidden

404 Not Found

429 Too Many Requests

500 Internal Server Error

⚠ Security issue if:

- 500 errors expose stack trace
- Detailed database errors returned

API Security Testing Report (Submission Format)

1 Objective

To assess API endpoints for authentication, authorization, and input validation weaknesses.

2 Tools Used

- Postman
 - cURL
 - Browser DevTools
-

3 Scope

Tested endpoints:

- /api/login
- /api/users
- /api/users/{id}

Findings

Finding 1: Broken Object Level Authorization

- Endpoint: GET /api/users/{id}
- Issue: User ID manipulation allowed access to other user data.
- Risk Level: High
- OWASP Mapping: API1 – Broken Object Level Authorization

Finding 2: No Rate Limiting

- Endpoint: POST /api/login
- Issue: Unlimited login attempts allowed.
- Risk Level: Medium
- OWASP Mapping: API4 – Unrestricted Resource Consumption

Finding 3: Detailed Error Messages

- Endpoint: /api/register
- Issue: Server returned stack trace.
- Risk Level: Low
- OWASP Mapping: API8 – Security Misconfiguration

Recommendations

- Implement role-based access control (RBAC)
- Enforce JWT validation on all protected endpoints
- Implement rate limiting (5 attempts per minute)
- Disable verbose error messages in production
- Validate all user inputs server-side

