CYBER SECURITY INTERNSHIP – TASK 13

Secure API Testing & Authorization Validation

---

1. Introduction

Modern applications rely heavily on Application Programming Interfaces (APIs) to exchange data between clients and backend servers. While APIs improve functionality and scalability, improper security controls can lead to serious vulnerabilities such as unauthorized access, data leakage, and abuse.

This task focuses on understanding REST APIs, testing API authentication and authorization, and identifying common API security weaknesses using **Postman**.

---

2. Understanding REST APIs

REST APIs use HTTP methods to perform operations on resources.

Common HTTP Methods

- **GET** – Retrieve data from the server
- **POST** – Send new data to the server
- **PUT** – Update existing data
- **DELETE** – Remove data

APIs typically exchange data in JSON format.

---

3. Tool Used

- **Postman** (Primary tool)
- Alternatives: cURL, Insomnia (conceptual)

Postman was used to send API requests, modify headers, and analyze responses.

---

4. API Configuration in Postman

An API endpoint was configured in Postman by setting:

- Endpoint URL
- HTTP method (GET / POST)
- Request headers
- Request body (if required)

This setup allows controlled testing of API behavior.

---

5. API Authentication Testing

API authentication ensures that only authorized users can access resources.

Testing Performed

- Requests were sent with **valid authentication tokens**
- Requests were sent with **invalid credentials**

Observation

APIs correctly rejected requests with invalid authentication details.

---

6. Unauthenticated Access Testing

Authentication headers were removed and the request was resent.

Observation

The API response was analyzed to check whether unauthenticated access was improperly allowed.

Improper access indicates an authentication misconfiguration.

---

7. Authorization Testing (Broken Authorization)

Authorization ensures users can access only permitted resources.

Testing Performed

- Resource identifiers (such as user ID) were modified in the request
- Access control behavior was observed

Observation

If unauthorized data is accessible, it indicates **Broken Object Level Authorization (BOLA)**.

---

8. Input Validation Testing

Malformed and unexpected input values were sent to the API.

Examples

- Invalid data types
- Missing parameters
- Unexpected values

Observation

Proper input validation prevents injection attacks and server crashes.

---

9. Rate Limiting Testing

Multiple rapid requests were sent to the API to test whether rate limiting was enforced.

Observation

Rate limiting helps prevent brute-force attacks and API abuse.

---

10. HTTP Response Code Analysis

API responses were analyzed using HTTP status codes:

- **200 OK** – Successful request
- **401 Unauthorized** – Authentication failure
- **403 Forbidden** – Authorization failure
- **400 Bad Request** – Input validation error
- **429 Too Many Requests** – Rate limit exceeded

Error messages were reviewed for sensitive information leakage.

---

11. Identified Vulnerabilities Mapped to OWASP API Risks

- Broken Authentication
- Broken Authorization (BOLA)
- Lack of Rate Limiting
- Improper Input Validation
- Excessive Error Information

These vulnerabilities align with **OWASP API Security Top Risks**.

---

12. Security Recommendations

- Enforce strong authentication mechanisms
- Implement proper authorization checks
- Apply strict input validation
- Enable rate limiting
- Avoid exposing sensitive error messages