Task 13: API Security Testing Report

Name: Mohammed Fadil PK **Date:** February 6, 2026

1. Introduction

APIs (Application Programming Interfaces) are the backbone of modern web applications. In this task, I used Postman to test a REST API for common vulnerabilities, including Broken Object Level Authorisation (BOLA) and Lack of Input Validation.

2. Setup & Tools

- **Tool Used:** Postman (v10) on Kali Linux.
- **Target API:** jsonplaceholder.typicode.com (Simulated environment).

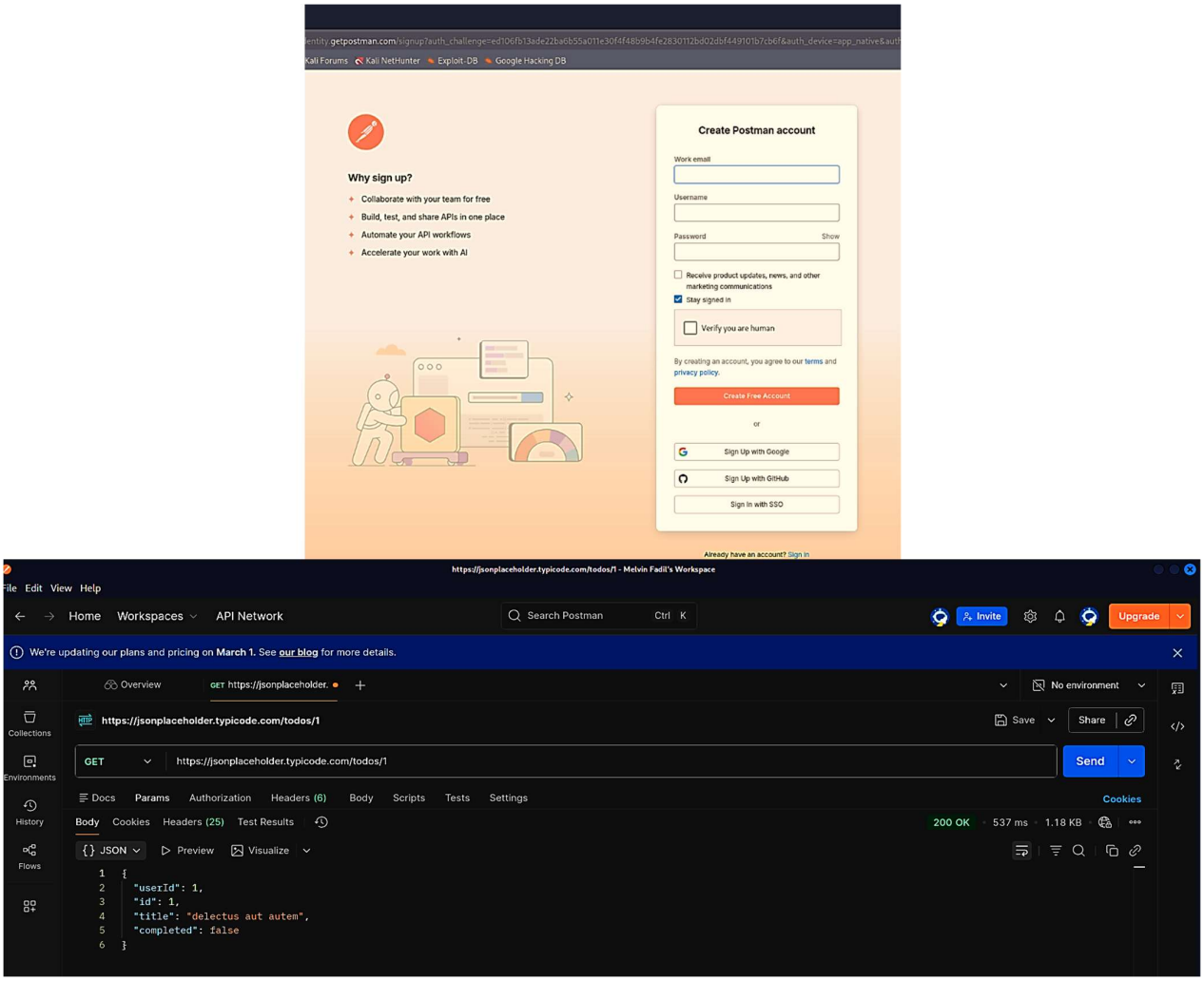

Fig 1: Postman Installation / Dashboard

3. Test 1: Broken Object Level Authorisation (BOLA)

I tested whether the API allows access to arbitrary user data by manipulating the ID parameter in the URL.

- **Request:** GET https://jsonplaceholder.typicode.com/todos/1
- **Attack:** Changed ID 1 to 5 to access another user's data.
- **Result:** The API returned the data for User 5 without any authorisation check (Status 200 OK).

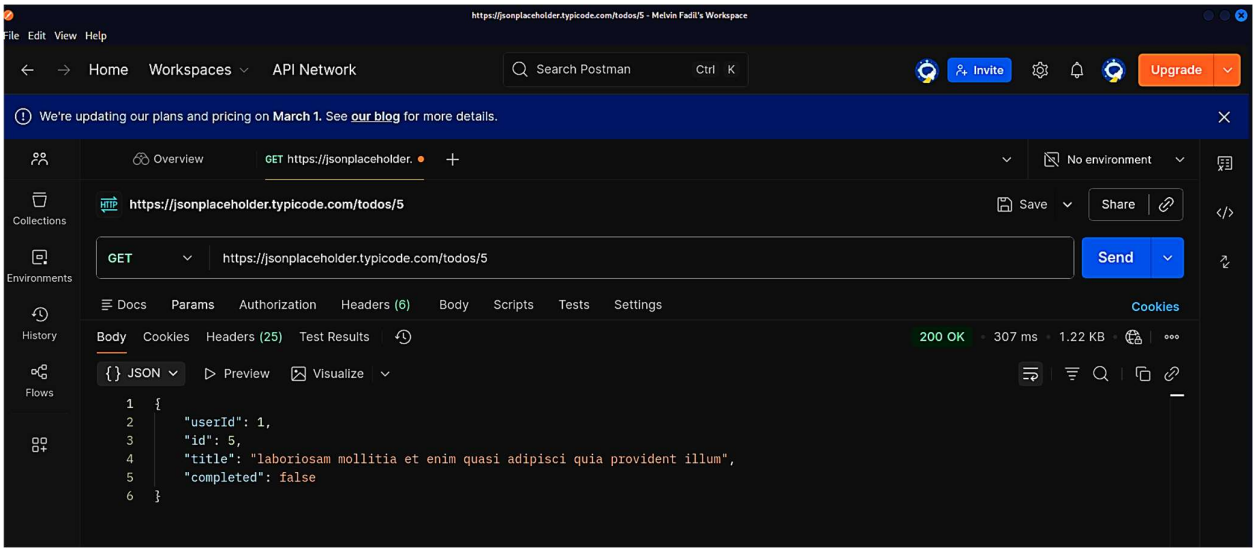

Fig 2: GET Request showing User Data

4. Test 2: Input Validation Testing (Data Injection)

I attempted to inject unauthorised content into the database using a POST request.

- **Method:** POST
- **Payload:**

JSON

```
{
  "title": "HACKED by Fadil",
  "body": "Security Testing in progress",
  "userId": 1
}
```

- **Result:** The server accepted the malicious payload and returned **Status 201 Created** with a new ID (101).

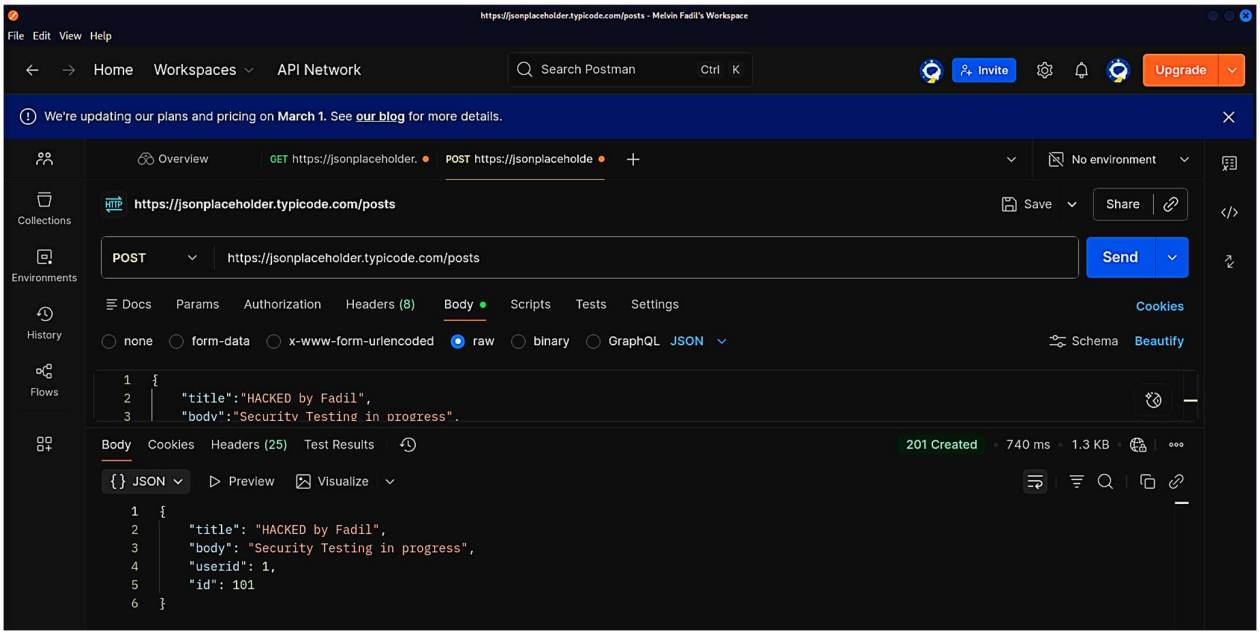

Fig 3: POST Request showing "201 Created"

5. Interview Questions

- **What is API Authentication?** It is the process of verifying the identity of the user (e.g., using API Keys or JWT Tokens) before allowing access.
- **What is Broken Authorisation (BOLA)?** A vulnerability where an attacker can access another user's data just by changing an ID number (e.g., changing /user/100 to /user/101).
- **Difference between GET and POST?**
    - **GET:** Used to **retrieve** data (e.g., reading a profile).
    - **POST:** Used to **send/create** data (e.g., posting a tweet).
- **Why is Rate Limiting important?** It prevents attackers from overloading the server (DDoS) or brute-forcing passwords by limiting how many requests they can send per minute.

6. Conclusion

This assessment demonstrated that without strict input validation and authorisation checks, APIs are vulnerable to data leakage and unauthorised data creation.