API SECURITY RISK ANALYSIS Report

Prepared by: Olaniyan Sodiq Abiodun
Date: 17-Febuary-2026
For: Future-interns / Internal Training

Executive Summary

We assessed the /users and /posts endpoints of the public API. Our audit identified three high-severity risks, including a complete lack of authentication and broken access control, which could allow an attacker to access all data. We recommend implementing strict authorization checks and rate limiting.

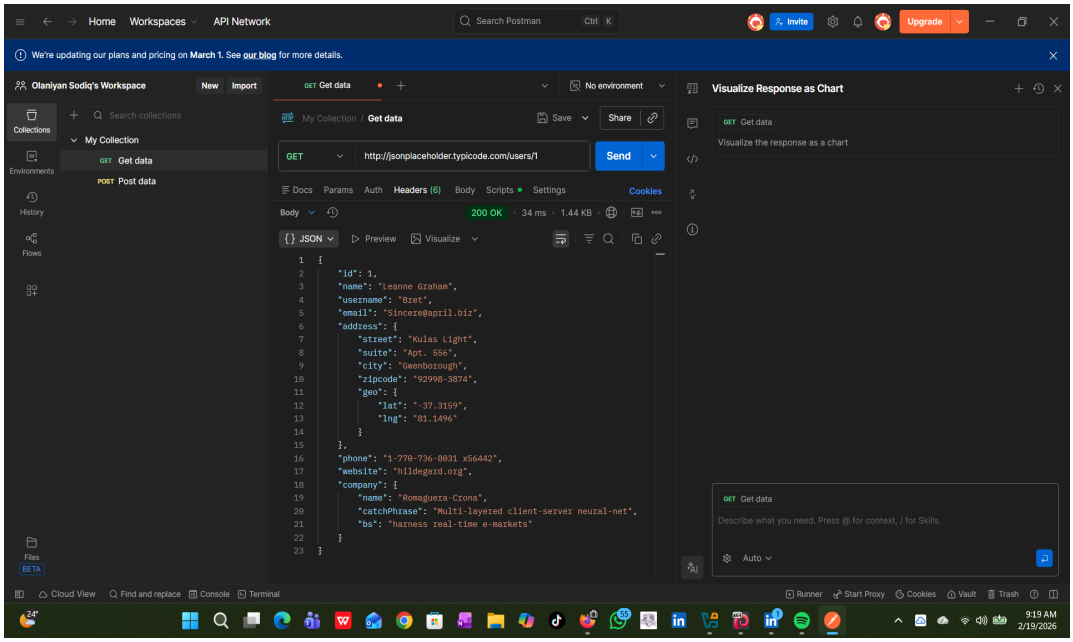
Scope

- Target: JSONPlaceholder API (typicode.com).
- Endpoints Tested: /posts, /users, /comments.
- Methodology: Read-only security analysis based on OWASP API Top 10.
- Tools Used: Postman, Browser DevTools, Google Docs.

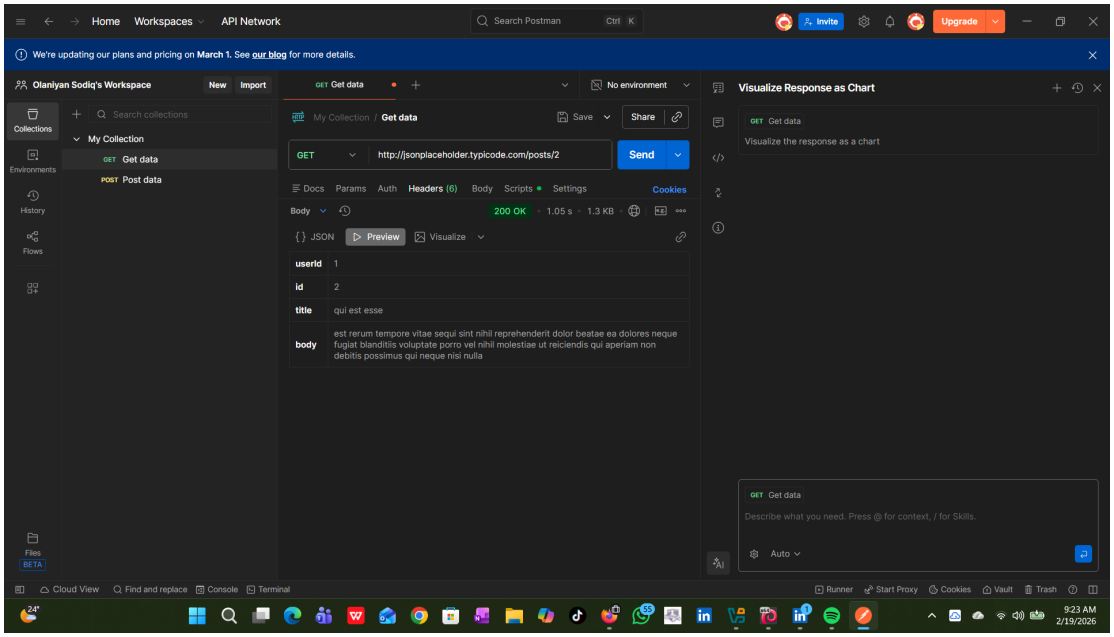
Findings

Risk Category	HTTP METHOD and Endpoint Tested	Observation	Business Impact	Severity	Remediation
Broken Authentication	GET /users/1	API accessible without any API key or token.	Unauthorized access to all user data, leading to data breaches and loss of customer trust.	High	Implement OAuth 2.0 or API Key authentication. Require valid tokens for every request.
Excessive Data Exposure	GET /users/1	Response includes lat and lng (geo-coordinates) when	Exposes user location data unnecessaril	Medium	Ensure the API only returns fields explicitly

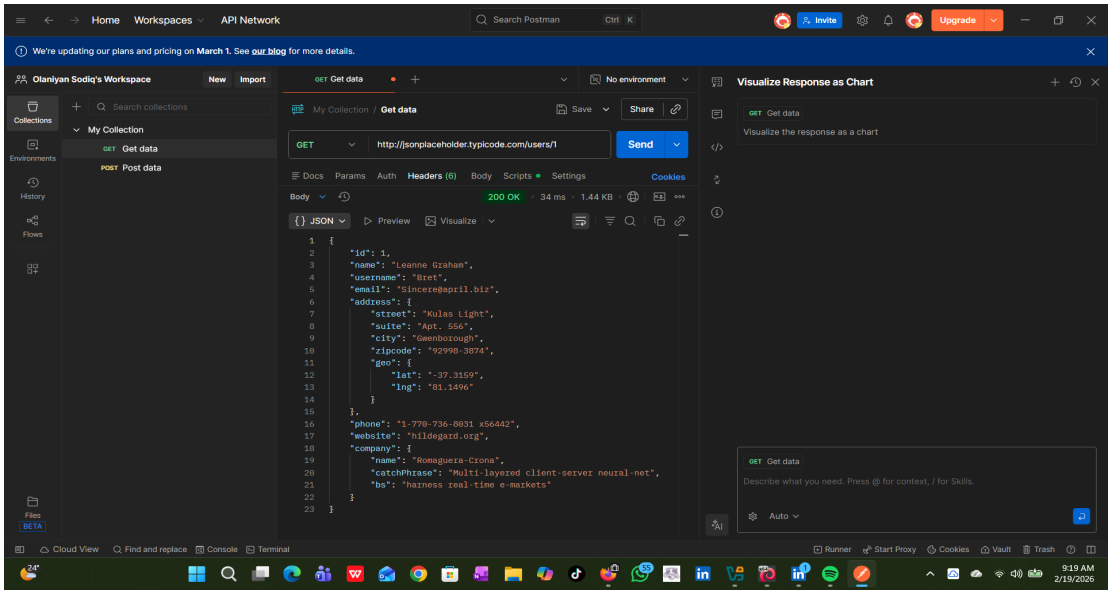
		only name/email is needed.	y, increasing the impact of a data breach.		needed by the client. Use schema-based responses.
Broken Object Level Authorization (BOLA)	GET /posts/2	Changing the ID in the URL from 1 to 2 allows access to another user's post.	Users can view/edit other users' private data. Could lead to major privacy scandals	High	Implement server-side access control checks. Verify the authenticated user owns the resource ID before returning data.
Missing Rate Limiting	GET /posts	Sent 30 rapid requests; all were successful.	System vulnerable to Denial of Service (DoS) and brute-force attacks, causing potential downtime.	Medium	Implement rate limiting (e.g., 100 requests per minute per IP). Return 429 Too Many Requests status code.
Input Validation	POST /posts	API accepted a script tag (<script>alert('XSS')</script>) in the request body.	If this data were stored and displayed, it could lead to Cross-Site Scripting (XSS) attacks against other users.	High	Implement strict input validation and output encoding. Sanitize all user inputs on the server side.



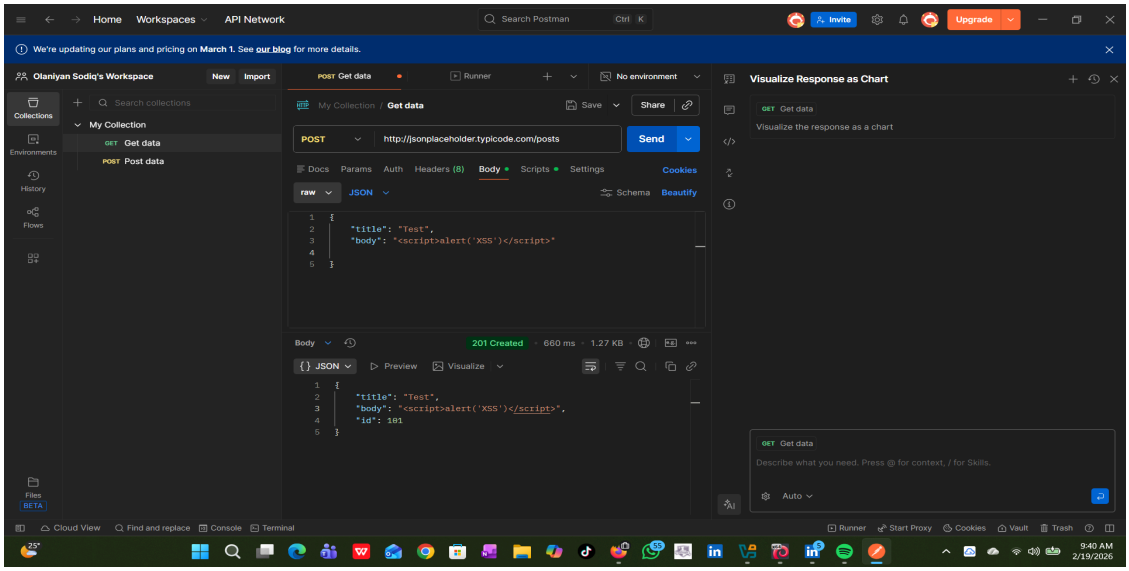
Auth-test.png



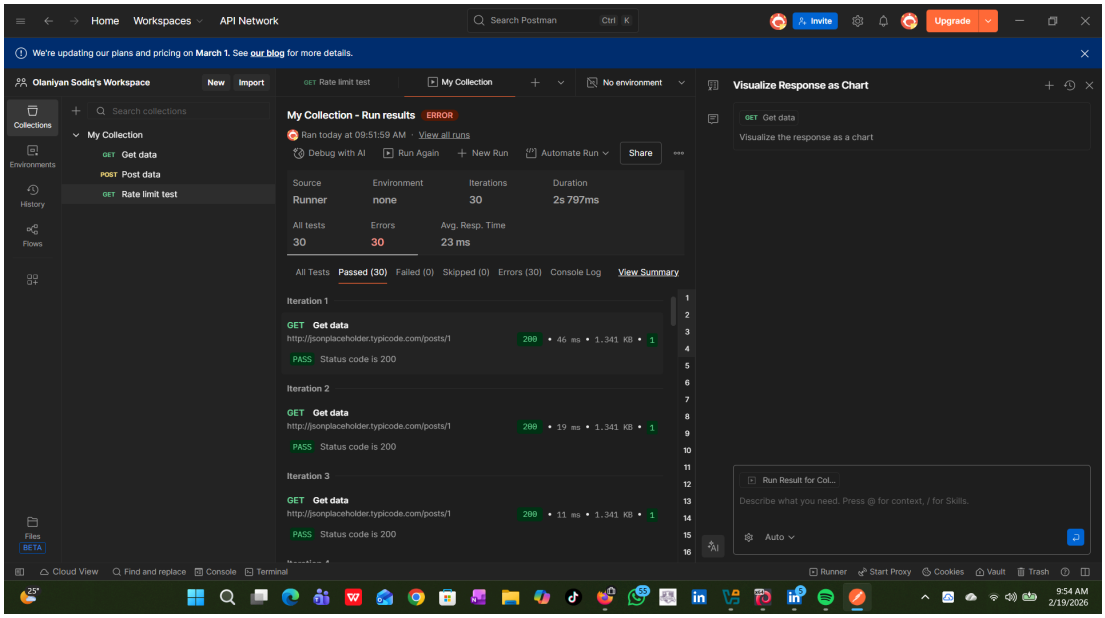
BOLA-test.png



Data-Exposure.png



Input-Validation.png



Rate-Limit.png