API Security Risk Analysis Report

API Assessed: JSONPlaceholder
Assessment Type: Read-Only Security Review
Prepared by: Cinchana S
Date: 25/02/2026

Executive Summary

This report presents a read-only API Security Risk Analysis conducted on the public demo API JSONPlaceholder. The objective of the assessment was to evaluate authentication mechanisms, authorization controls, input validation practices, data exposure levels, and protection against resource abuse. Testing was performed using industry-standard API inspection techniques without attempting exploitation, denial-of-service, or unauthorized intrusion.

The assessment identified multiple security weaknesses that would pose significant risk if present in a production SaaS environment. Key findings include the absence of authentication requirements, potential Broken Object Level Authorization (BOLA) risks through ID enumeration, lack of input validation, acceptance of excessively large payloads without restriction, and excessive data exposure in API responses. These issues align with common risks outlined by the OWASP API Security Top 10 and demonstrate how insecure API design can lead to unauthorized data access, data manipulation, service abuse, and increased operational risk.

Although the assessed API is intentionally designed for testing and educational purposes, this analysis highlights critical security controls that must be implemented in real-world SaaS systems, including strong authentication mechanisms, strict authorization checks, schema-based input validation, rate limiting, and data minimization. Implementing these measures significantly reduces the attack surface and strengthens the overall security posture of modern API-driven applications.

Scope & Methodology

Scope

- Public demo API endpoints only
- Read-only testing
- No exploitation attempts
- No denial-of-service testing
- No bypass attempts

Tools Used

- Postman (API request testing)
- Browser DevTools (header inspection)
- Documentation review

Methodology

The following security checks were performed:

- Endpoint enumeration
- Authentication verification
- Authorization assessment
- Response data analysis
- Input validation testing
- Payload size testing
- Header inspection

API Overview

The API assessed in this security analysis is JSONPlaceholder, a publicly available REST API designed for testing and educational purposes. The base URL for the API is https://jsonplaceholder.typicode.com. This API provides multiple endpoints such as `/users`, `/posts`, and `/comments`, which simulate common SaaS application functionalities including user data retrieval and content creation. JSONPlaceholder was selected for this assessment because it is intentionally built for safe testing, requires no authentication setup, and allows read-only experimentation without risk of impacting real users or production systems. Its open design makes it suitable for demonstrating common API security risks such as missing authentication, authorization weaknesses, input validation gaps, and excessive data exposure in a controlled and ethical environment.

API Endpoints Tested

Method	Endpoint	Purpose
GET	/users	Retrieve user list
GET	/users/{id}	Retrieve specific user
POST	/posts	Create new post

Detailed Findings

Finding 1: No Authentication Required

Severity: High
Category: Broken Authentication

Description

The API allows access to endpoints without requiring authentication tokens, API keys, or session validation.

Evidence

GET and POST requests were successfully executed without providing any Authorization header.

Business Impact

If implemented in a production SaaS system, this could allow:

- Unauthorized data access
- Anonymous data manipulation
- Data scraping
- Abuse of backend services

Remediation

- Implement JWT or OAuth 2.0 authentication
- Enforce token validation middleware
- Require authentication for all non-public endpoints

Finding 2: Potential Broken Object Level Authorization (BOLA)

Severity: High
Category: Authorization Failure

Description

User data can be accessed by modifying the user ID in the request path.

Example:
GET /users/1
GET /users/2
GET /users/3

No authorization checks were enforced.

Business Impact

Attackers could enumerate and extract all customer records in a real-world scenario.

Remediation

- Implement server-side authorization checks
- Validate ownership of requested resources
- Apply Role-Based Access Control (RBAC)

Finding 3: Lack of Input Validation

Severity: Medium
Category: Improper Input Validation

Description

The API accepts malformed payloads without validating data types.

Example payload sent:

{

"title": 999,

"body": true

}

The request was processed successfully.

Business Impact

This could lead to:

- Data corruption
- Backend instability
- Increased injection risk
- Business logic abuse

Remediation

- Implement strict schema validation
- Enforce data type verification
- Return HTTP 400 for malformed input
- Use validation libraries (e.g., Joi)

Finding 4: No Payload Size Restrictions

Severity: Medium
Category: Unrestricted Resource Consumption

Description

The API accepts excessively large request bodies without rejecting or limiting size.

Large string inputs were successfully processed.

Business Impact

In production, this could result in:

- Resource exhaustion
- Increased infrastructure costs
- Service slowdown
- Potential Denial-of-Service conditions

Remediation

- Enforce maximum request body size
- Implement field length validation
- Configure server-level request limits
- Apply rate limiting

Finding 5: Excessive Data Exposure

Severity: Medium

Description

The `/users` endpoint returns full user object details including:

- Email
- Phone
- Address
- Company information

Business Impact

In real SaaS applications, excessive data exposure increases:

- Privacy risk
- Information leakage
- Targeted attack surface

Remediation

- Return only required fields
- Implement response filtering
- Apply data minimization principles

Risk Summary Table

Findings	Risk	Severity
1	No Authentication	High
2	BOLA Risk	High
3	Input Validation Failure	Medium
4	No Payload Size Limit	Medium
5	Excessive Data Exposure	Medium

Overall Risk Assessment

If these vulnerabilities existed in a production SaaS application, the overall risk posture would be classified as:

“High Risk”

Immediate implementation of authentication, authorization, validation, and rate limiting controls would be required.

Ethical Statement

This assessment was conducted strictly on a public demo API designed for educational purposes.
No exploitation, denial-of-service testing, or unauthorized access attempts were performed.

Conclusion

The API Security Risk Analysis of JSONPlaceholder revealed several security weaknesses, including missing authentication, potential ID-based access issues, insufficient input validation, lack of payload limits, and excessive data exposure. While acceptable in this demo API, these issues would pose significant risks in a production environment. Implementing proper authentication, authorization, input validation, and data controls can greatly reduce the attack surface and protect sensitive information in real-world API-driven applications.