API Security Risk Analysis Report

INTERNSHIP TASK 3 – CYBER SECURITY

Laiken Naidoo
Cyber Security Track (CS)
Future Interns Internship
February 2026

1. Executive Summary

This report presents a security risk analysis of a public REST API (JSONPlaceholder) conducted as part of Internship Task 3 in the Cyber Security track. The objective of this assessment was to evaluate authentication mechanisms, authorization controls, data exposure risks, and overall API security posture using ethical and read-only testing methods.

Testing was performed using Postman to analyze API endpoints, response data, and headers. The assessment identified several security weaknesses including lack of authentication enforcement, absence of authorization controls, excessive data exposure, and missing rate-limiting mechanisms.

Although the selected API is designed for demonstration purposes, these findings represent critical vulnerabilities that would significantly impact a real-world SaaS application if implemented in production.

2. API Selection and Documentation Review

2.1. API Selection

The API selected for this security assessment was **JSONPlaceholder**, a publicly available REST API designed for testing and educational purposes.

API Name: JSONPlaceholder
Base URL: https://jsonplaceholder.typicode.com
API Type: REST
Authentication Requirement: None observed

JSONPlaceholder was selected because it allows safe and ethical testing of API endpoints without requiring credentials or access to live production systems.

2.2. Documentation Review

Before conducting testing, the official API documentation was reviewed to understand:

- Available endpoints
- Supported HTTP methods (GET, POST)
- Request and response structures
- Expected data formats (JSON)

Based on the documentation review, the following endpoints were selected for testing:

- GET /posts
- GET /posts/1
- GET /comments
- POST /posts

The documentation confirmed that the API does not implement authentication or authorization controls, making it suitable for demonstrating common API security risks.

3. Scope and Methodology

3.1. Scope

The scope of this API security assessment was limited to controlled and ethical testing of publicly available endpoints. The following boundaries were strictly maintained:

- Testing was performed only on a public demo API.
- Only read-only GET requests and safe POST requests were executed.
- No exploitation attempts were made.
- No authentication bypass attempts were performed.
- No denial-of-service (DoS) or load testing was conducted.
- No modification of backend data or infrastructure was attempted.

The assessment focused purely on identifying potential security risks through observation and analysis.

3.2. Tools Used

The following tools were used during testing:

- **Postman –** For sending HTTP requests and analyzing responses.
- **Browser Developer Tools –** For inspecting response behavior and API interactions.
- **Manual Security Analysis –** For evaluating authentication, authorization, and data exposure risks.

3.3. Methodology

The methodology followed during this assessment included:

1. Reviewing API documentation to understand endpoint structure.
2. Sending GET requests to retrieve resources.
3. Inspecting response payloads for sensitive data exposure.
4. Reviewing response headers for authentication and rate-limiting controls.
5. Testing a safe POST request to observe API behavior.
6. Identifying potential security weaknesses.
7. Classifying risks based on severity (Low, Medium, High).
8. Recommending remediation strategies aligned with secure API development practices.

4. Authentication, Header and Response Inspection

This section evaluates how the API handles authentication, authorization, request headers, and response data security.
The API tested:
JSONPlaceholder (https://jsonplaceholder.typicode.com)
Tested endpoints:

- GET /posts
- POST /posts

Testing tool:

- Postman

4.1. Authentication Analysis

The API was tested to determine whether authentication mechanisms were enforced.
During testing:

- No API key was required.
- No Bearer token was required.
- No OAuth mechanism was required.
- No login or session validation was required.
- Requests were successfully executed without providing any credentials.

Both GET and POST endpoints were accessible without authentication.
This confirms that the API does not implement authentication controls.

4.2. Authorization Analysis

Authorization testing was conducted to determine whether access restrictions were enforced per user.
Observations:

- The GET /posts endpoint returned 100 records in a single request.
- Records belonging to multiple userId values (1–10) were returned.
- No ownership validation was enforced.
- No access restrictions were implemented.

This indicates the absence of role-based access control (RBAC) or object-level authorization.

4.3. Request Header Inspection

The request headers generated by Postman included:

- Content-Type: application/json
- User-Agent: PostmanRuntime
- Accept: */*
- Host
- Connection

No security-related headers were required by the server.
No Authorization header was required to complete the request.

4.4. Response Header Inspection

Response headers were inspected for security controls.
Observed headers included:

- Content-Type: application/json
- Cache-Control
- Connection
- Content-Encoding
- ETag

However, the following security headers were not present:

- Authorization headers
- WWW-Authenticate
- X-RateLimit-Limit
- X-RateLimit-Remaining
- Strict-Transport-Security
- Content-Security-Policy

This indicates the absence of authentication enforcement and rate-limiting mechanisms.

4.5. Response Data Analysis

The API returned structured JSON data containing:

- userId
- id
- title
- body

The API returned 100 records in a single request.
No pagination enforcement or record limitation was observed.
All user data was publicly accessible without restrictions.
In a production SaaS application, unrestricted access to large datasets without authentication or pagination controls could result in data scraping, privacy violations, and increased exposure to automated attacks.

5. Identified Security Risks

The following security risks were identified during testing of the JSONPlaceholder API. Each risk is classified based on severity (Low, Medium, High) and evaluated in terms of potential business impact if deployed in a real-world production environment.

5.1. Lack of Authentication Enforcement

The API does not require authentication credentials such as API keys, bearer tokens, OAuth tokens, or session cookies. All endpoints were accessible without identity verification.
Severity: High
Business Impact:
In a production SaaS environment, the absence of authentication would allow unauthorized users to access sensitive resources. This could result in:

- Data breaches
- Exposure of confidential information
- Regulatory violations (e.g., POPIA/GDPR)
- Reputational damage
- Financial penalties

Remediation Recommendation:
Implement strong authentication mechanisms such as:

- OAuth 2.0
- JWT-based authentication
- API keys with secure validation
- Enforced login/session management

All endpoints should require valid authentication before processing requests.

5.2. Lack of Authorization Controls (Broken Access Control)

The API returned data belonging to multiple userId values in a single request without enforcing ownership validation or role-based access restrictions.
Severity: High
Business Impact:
Without proper authorization controls, users may gain access to data belonging to other users. This could result in:

- Privacy violations
- Horizontal privilege escalation
- Unauthorized data exposure
- Legal liability

Remediation Recommendation:
Implement:

- Role-Based Access Control (RBAC)
- Object-level access validation
- User ownership verification before returning resources

Each request should validate whether the authenticated user has permission to access the requested resource.

5.3. Excessive Data Exposure

The GET /posts endpoint returned 100 records in a single response. No pagination, filtering, or data minimization controls were enforced.
Severity: Medium
Business Impact:
Returning large datasets without restriction increases the risk of:

- Automated scraping
- Bulk data harvesting
- Increased attack surface
- Performance degradation under abuse

Remediation Recommendation:

- Implement pagination (limit/offset parameters)
- Enforce maximum record limits per request
- Return only necessary fields (data minimization principle)

5.4. Absence of Rate Limiting Controls

Response headers did not include rate-limiting controls such as X-RateLimit-Limit or X-RateLimit-Remaining. No request throttling was observed.
Severity: Medium
Business Impact:
Without rate limiting, attackers could:

- Execute automated attacks
- Perform brute-force attempts
- Launch denial-of-service (DoS) attacks
- Overload backend systems

Remediation Recommendation:

- Implement rate limiting per IP and per user
- Apply request throttling policies
- Monitor abnormal traffic behavior
- Use API gateways with built-in rate control

5.5. Missing Security Headers

The API responses did not include security-related headers such as:

- Strict-Transport-Security
- Content-Security-Policy
- X-Content-Type-Options

Severity: Low to Medium
Business Impact:
While less critical for API-only services, missing security headers may increase exposure to:

- Man-in-the-middle risks
- Content-type sniffing attacks
- Reduced transport security enforcement

Remediation Recommendation:

- Enforce HTTPS with HSTS
- Configure secure response headers
- Implement proper transport-layer protection

The identified risks align with common issues described in the OWASP API Security Top 10 framework, particularly broken authentication and broken object-level authorization.

6. Risk Severity Summary

The following table summarizes the identified risks, their severity levels, and their potential business impact:

Risk ID	Risk Title	Severity	Business Impact
5.1	Lack of Authentication Enforcement	High	Unauthorized access to sensitive data and potential data breaches
5.2	Lack of Authorization Controls	High	Horizontal privilege escalation and user data exposure
5.3	Excessive Data Exposure	Medium	Increased risk of data scraping and bulk harvesting
5.4	Absence of Rate Limiting	Medium	Potential API abuse, automated attacks, and service overload
5.5	Missing Security Headers	Low–Medium	Reduced transport-layer and response-level security protections

7. Recommendations Overview

To improve API security posture, the following measures are recommended:

1. Implement strong authentication mechanisms such as JWT or OAuth 2.0.
2. Enforce role-based access control (RBAC) and object-level authorization.
3. Apply pagination and data minimization principles to reduce excessive data exposure.
4. Configure API rate limiting to prevent abuse and automated attacks.
5. Enforce HTTPS and configure appropriate security headers.
6. Deploy API gateway protections for centralized monitoring and traffic control.

Implementing these controls would significantly strengthen API security and reduce exposure to common SaaS-related vulnerabilities.

8. Conclusion

This assessment demonstrated how API security weaknesses can be identified through structured testing and analysis. While JSONPlaceholder is a demonstration API, the identified issues represent critical vulnerabilities if implemented in a real-world production SaaS environment.
Modern applications rely heavily on APIs, making authentication, authorization, rate limiting, and data protection essential components of secure architecture. Organizations must prioritize API security to protect sensitive data, maintain regulatory compliance, and preserve customer trust.