# Restful-Booker API SECURITY SCENARIO

### Information:

Project Name: **Restful-Booker**

Created by: Hitesh Kantariya

Creation Date: 08/04/2025

Description: This document contains all scenarios and steps to improve the security of Restful-Booker API

Reference: Postman Practical’s, Test cases, Bug Report

## Points to Consider:

1. Secure Call with HTTPS (Do not accept HTTP format requests)
2. Avoid basic Authentication with simple Username and password, Use JWT OR OAuth 1.0/ 2.0 methods.

* EX. BUG\_S02: Those API uses Simple Authentication

1. Always use Auth tokens with expiry date.
2. User privilege escalation test (normal user can not able to access admin privileges).
3. Validate User Input. Ex. Special chars, Length, Format (Mail, mobile number, etc)
4. Use of Quotas and throttling to avoid DDOS and Brute force attack

* Throttling: Limits how many requests a user /IP can make per second/minute.
* Quota: Sets a max limit (ex. 1000 API calls/day) per user/app
* Throttling can be per-user, per-IP OR per-endpoint
* Combine with CAPTCHA or Account lockout for brute-force defence.

1. Detect insecure API calls with sniffers

|  |  |
| --- | --- |
|  |  |

* If user sending password in simple text, Sensitive data in URL, Unencrypted token all problems solved and security increased

1. Use HSTS headers with SSL to avoid SSL strip attack
2. Validate content type on request accept header
3. Don’t use sensitive data in URL but use standard authorization header.

## Data Type Validation – Checks if values are of correct type (e.g., string, number, Boolean).

* Ex. BUG\_S01: In those API’s “first\_name” field accept number.
* For this validation check all fields properly.
* Required Fields – Ensures all mandatory fields are present. Ex. First\_name required.

## Structure Validation – Confirms the JSON structure matches the defined schema.

* User cannot enter any data except than declared in JSON schema.
* In those API users can not add {“salary”: “40,000/-”} pair in request body and if user add than ignored by the API.
* Format Validation – Validates formats like email, date-time, URI (if defined in schema).
* Length Validation – Check if text or number fields meet length/range limits.
* Value Validation – Ensure fields have valid/expected values (e.g., status = "active").
* Ex. For those API’s “payment status” accept only “True” or “False”.

1. Mask confidential fields (Ex. password)
2. Block methods are not used (Ex. Block PUT and DELETE methods if not required)
3. Content-type validation ex. If user selected JSON than accept only JSON otherwise return error (406/415 – unsupported headers)
4. Session management (token expiry, logout invalidates token, no reuse of expired tokens)
5. SQL injection and XSS attack testing