

# Secure API Testing & Authorization Validation

## Introduction

APIs (Application Programming Interfaces) allow communication between client and server using HTTP methods like GET, POST, PUT, and DELETE. Since APIs handle sensitive data and authentication, they are common targets for attacks.

Secure API testing focuses on checking authentication, authorization, input validation, and rate limiting to identify security weaknesses. In this practical, Postman is used to test API security and validate access controls.

## Tools Required

- Kali Linux
- Postman
- cURL (Pre-installed)
- Internet Connection

## Theory

### 3.1 REST API

REST APIs allow communication between client and server using HTTP methods:

- GET – Retrieve data
- POST – Create data
- PUT – Update data
- DELETE – Delete data

### 3.2 Authentication

Authentication verifies the identity of the user using:

- Bearer Tokens
- API Keys
- Username and Password

### 3.3 Authorization

Authorization determines what resources an authenticated user can access.

## 4. Procedure

### Step 1: Install Postman in Kali Linux

```
sudo snap install postman
```

Screenshot 1: Postman Installation

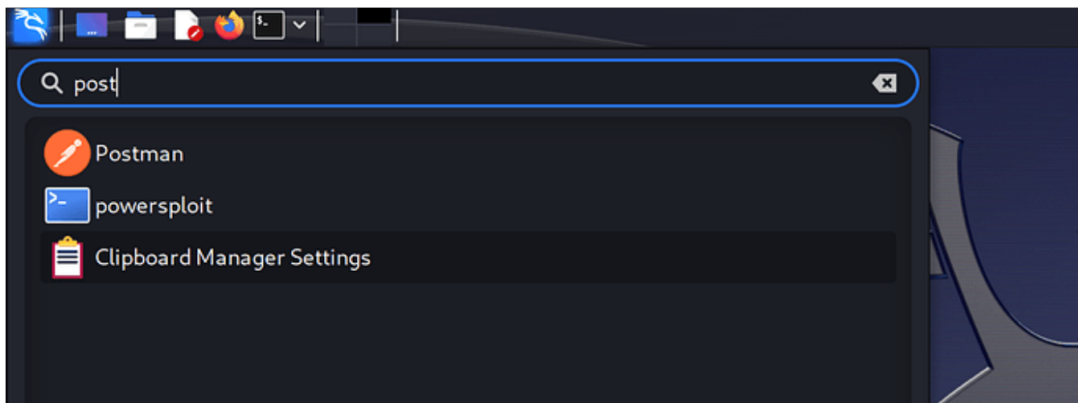


Figure 1: Installing Postman in Kali Linux

### Step 2: Configure API Request

- Open Postman
- Create New HTTP Request
- Enter API Endpoint

```
https://example.com/api/users
```

Add Headers:

```
Content-Type: application/json  
Authorization: Bearer <token>
```

Screenshot 2: API Request Configuration

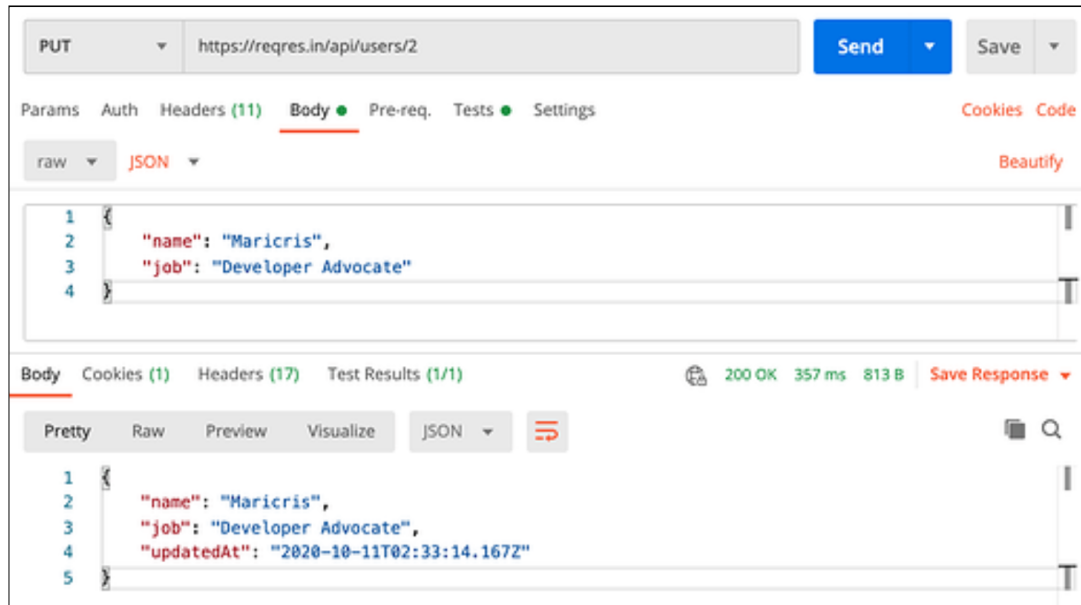


Figure 2: API Request Setup in Postman

## Step 3: Authentication Testing

### Valid Token Test

Expected Response: 200 OK

### Screenshot 3: Valid Token Response

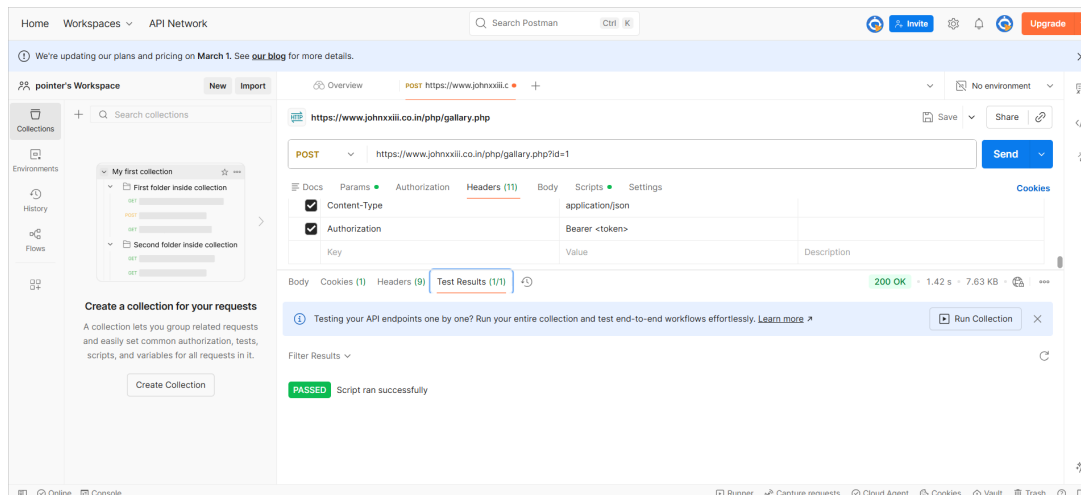


Figure 3: Successful Authentication Response

### Invalid Token Test

Expected Response: 401 Unauthorized

### Screenshot 4: Invalid Token Response

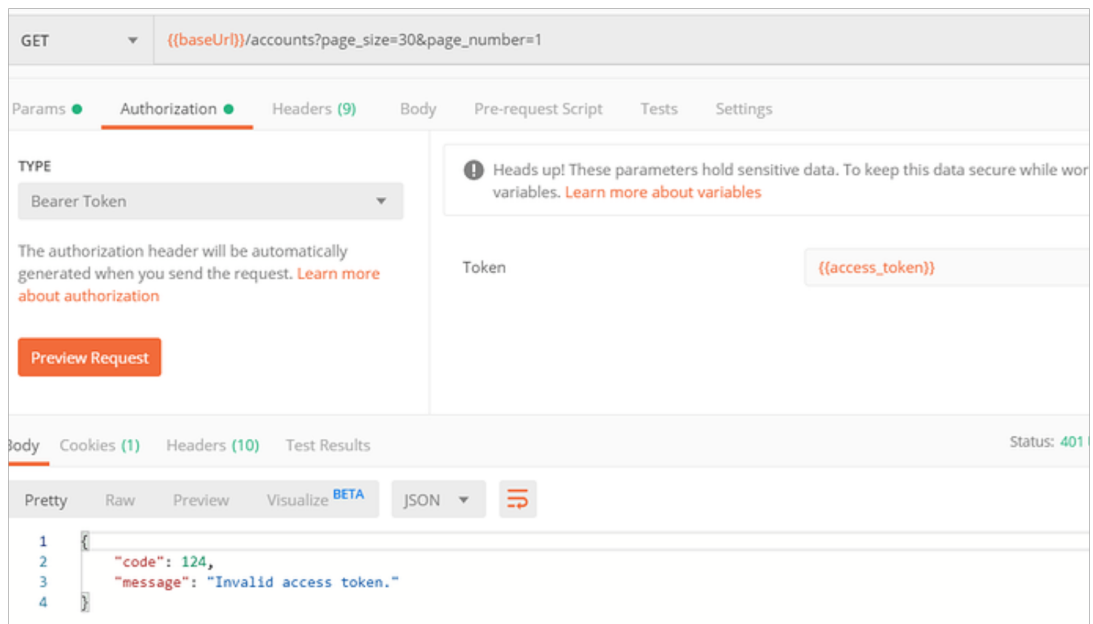


Figure 4: Unauthorized Access Response

## Step 4: Authorization Testing (IDOR)

Modify resource ID:

```
GET /api/users/101
GET /api/users/102
```

If unauthorized data is accessible → Broken Authorization Vulnerability.

### Screenshot 5: IDOR Testing

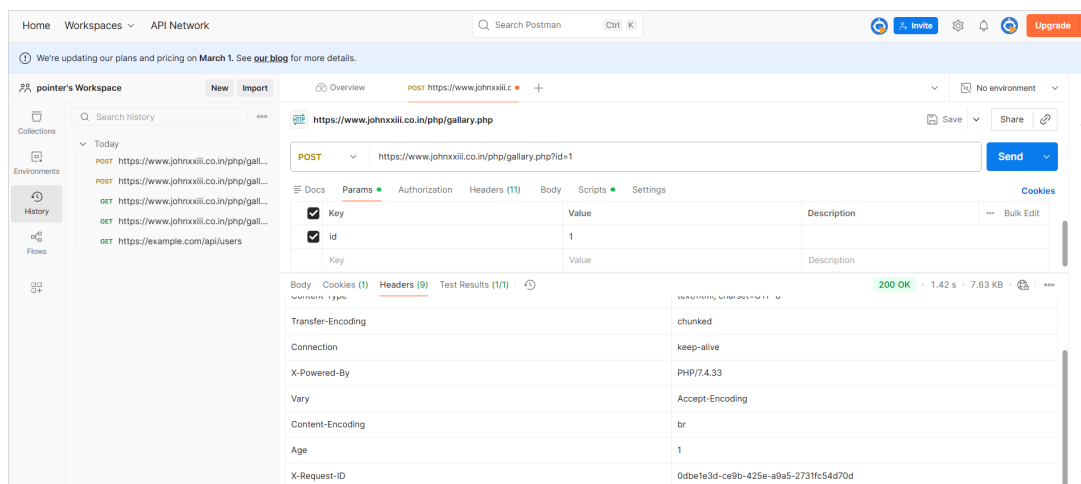


Figure 5: Testing for Broken Authorization

## Step 5: Input Validation Testing

Send malicious payload:

```
{
```

```
"username": "' OR 1=1 --",  
"password": "test"  
}
```

Observe for:

- 500 Internal Server Error
- SQL Error
- Stack Trace

### Screenshot 6: Input Validation Test



Figure 6: Testing Malicious Input Handling

### Step 6: Rate Limiting Test

```
for i in {1..50}; do  
curl -X GET https://example.com/api/users;  
done
```

Check if server returns:

- 429 Too Many Requests

### Screenshot 7: Rate Limiting Test

```
#!/bin/bash

## 1. cURL xargs + seq
seq 1 100 | xargs -n1 -I -P5 curl "http://example.com/"

## 2. cURL --parallel
curl --parallel --parallel-immediate --parallel-max 3 example.com example.com example.com

## 3. cURL --parallel input file
curl --parallel --parallel-immediate --parallel-max 10 --config source.txt

## 4. cURL in for loop
for ((request=1;request<=5;request++))
do
    for `seq 2`
    do
        time curl http://example.com & sleep 2
    done
done
```

Figure 7: Testing Rate Limiting Mechanism

## 5. Observation

- API correctly validates authentication tokens.
- Unauthorized access is restricted.
- Input validation prevents malicious payload execution.
- Rate limiting controls excessive requests.
- No sensitive internal errors are exposed.

## 6. Result

Secure API testing was successfully performed in Kali Linux. Authentication, authorization, input validation, and rate limiting mechanisms were evaluated.

## 7. Conclusion

Secure API testing is essential to identify vulnerabilities such as authentication bypass, broken authorization (IDOR), improper input validation, and missing rate limiting. Implementing proper security controls ensures robust API protection.