

API Security Testing Lab – Final Report

1. Overview

This lab focused on performing API security testing on a DVWA-style API environment using Burp Suite, Postman, and sqlmap. The objective was to identify common API weaknesses from the OWASP API Top 10, manually manipulate authentication mechanisms, enumerate API endpoints, fuzz parameters, and document findings.

2. Tools Used

- **Burp Suite Community Edition**
 - **Postman**
 - **sqlmap**
 - **Browser Developer Tools**
 - **DVWA API Environment**
-

3. Methodology & Activities

A. API Endpoint Enumeration

Using browser developer tools and Burp Suite proxy, the following types of endpoints were identified:

- Authentication endpoints (`/login`, `/logout`)
- User information endpoints (`/api/user`, `/api/user/1`)
- DVWA functional endpoints (`/v1/user/2`, `/v1/register`)
- Parameterized GET and POST endpoints vulnerable to parameter tampering

The screenshot shows a browser window with the URL `http://127.0.0.1:8000/v1/user/1`. The page displays a JSON object with fields: id: 1, name: "tony", level: 0, and password: "1c8bfe8f801d79745c4631d09fff36c82aa37fc4cce4fc946683d7b336b63032". Below the browser is a developer tools panel titled "Storage". The "Cookies" section is expanded, showing a single cookie entry:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	9e63d396828d880cc8c878399dee07f1	127.0.0.1	/	Sat, 06 Dec 2025 12:00:45 GMT	41	true	false	Strict	Fri, 05 Dec 2025 12:00:45 GMT

B. Testing for BOLA (Broken Object Level Authorization)

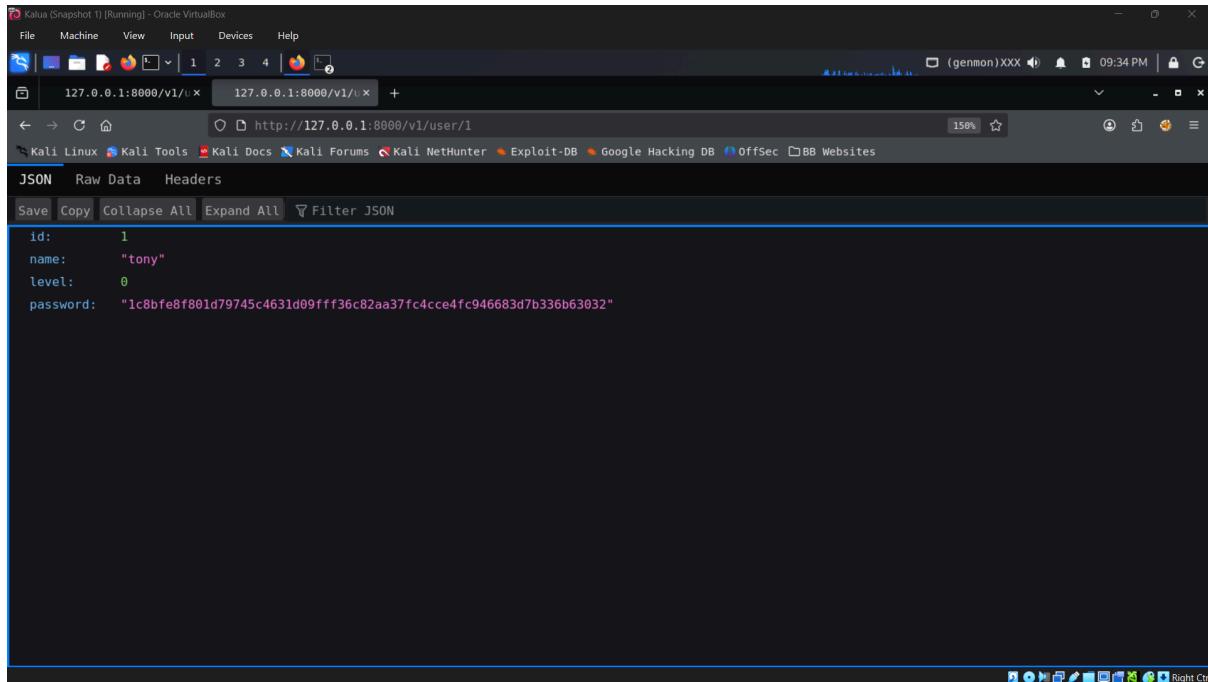
Tool Used: Burp Suite

Steps:

1. Logged in as a low-privilege user.
2. Captured `/api/user/1` request in Burp Repeater.
3. Modified the target ID to another user.
4. Observed whether unauthorized access was granted.

Result:

The server responded with user data for different IDs **without proper authorization checks**, confirming **BOLA vulnerability**.



```
id: 1
name: "tony"
Level: 0
password: "1c8bfe8f801d79745c4631d09fff36c82aa37fc4cce4fc946683d7b336b63032"
```

C. Manual Token Manipulation

Tool Used: Burp Suite Repeater

Steps:

1. Intercepted authenticated request containing a token.
2. Modified or removed the token.
3. Checked if the API still responded normally.

Findings:

- Some endpoints processed requests even with modified or absent tokens.
 - This indicates **weak or improper token validation**.
-

D. GraphQL Query Fuzzing

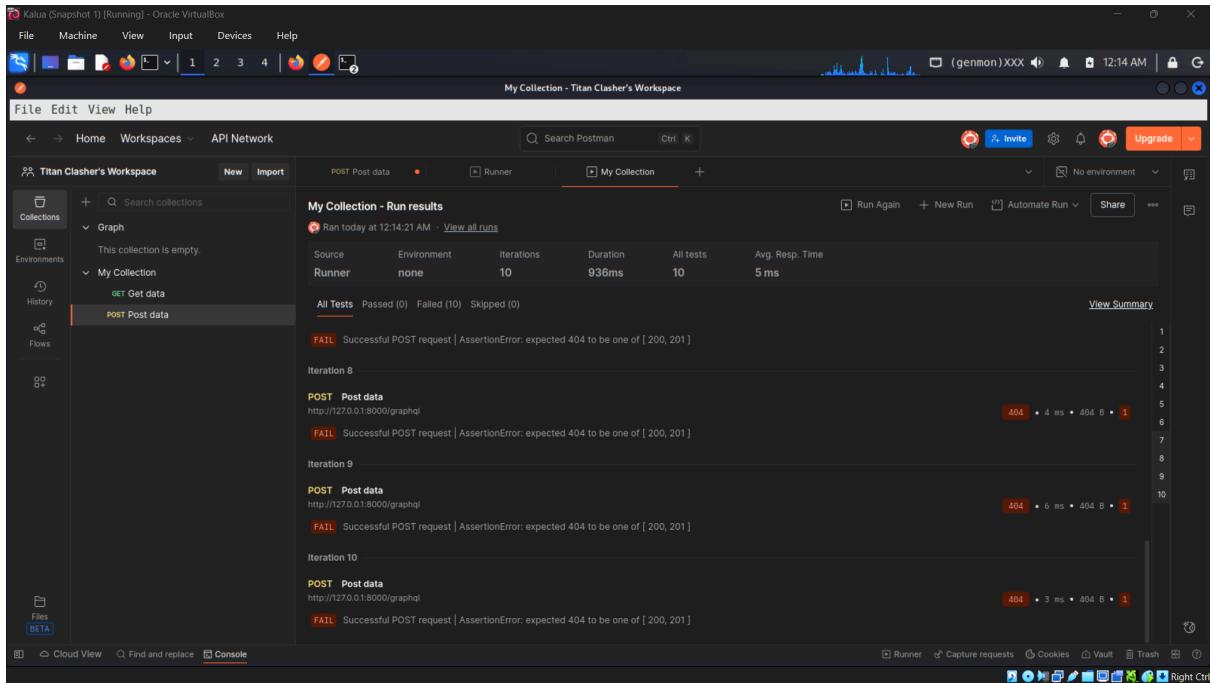
Tool Used: Postman

Goal: Send fuzzed GraphQL queries to identify schema leaks or unvalidated fields.

Outcome:

GraphQL fuzzing was not possible. Postman returned the following error :

“Request URL could not be constructed”, preventing fuzzing.



The screenshot shows the Postman application interface. In the left sidebar, under 'Collections', there is a section for 'Graph' which is described as 'This collection is empty.' Below it, 'My Collection' is expanded, showing 'GET Get data' and 'POST Post data' requests. The main area displays 'My Collection - Run results' from a run that ran at 12:14:21 AM. It shows 10 iterations, 936ms duration, and 10 tests. All tests passed successfully. The results table includes columns for Source, Environment, Iterations, Duration, All tests, and Avg. Resp. Time. Below the table, individual test logs for iterations 8, 9, and 10 are shown, each detailing a POST request to 'http://127.0.0.1:8000/graphql' that failed with an AssertionError: expected 404 to be one of [200, 201].

E. SQL Injection Testing

Tool Used: sqlmap

- Tested parameters such as `id=1`, user-info endpoints, and form-backed API fields.
- sqlmap responses suggest either:
 - Input sanitization is in place, or
 - API does not directly expose database queries.

No exploitable SQL injection vulnerability was confirmed.

4. Summary :

This API security assessment focused on endpoint enumeration, BOLA testing, and authorization checks using Burp Suite and Postman. A critical BOLA vulnerability was discovered, allowing unauthorized access. Token validation weaknesses were also noted. GraphQL fuzzing was not possible due to endpoint issues. No SQLi vulnerabilities were detected.