DVAPI Vulnerability Assessment Report

# API5:2023 – Broken Function Level Authorization

## Objective

To demonstrate how a Broken Function Level Authorization vulnerability in DVAPI (API5:2023) was exploited to escalate privileges and access unauthorized functionality, specifically by manipulating HTTP methods.

## Overview of DVAPI

DVAPI (Damn Vulnerable API) is a purposely insecure RESTful API designed for practicing API security. It includes real-world API vulnerabilities mapped to the OWASP API Security Top 10, including broken function-level authorization issues.

## Vulnerability Description

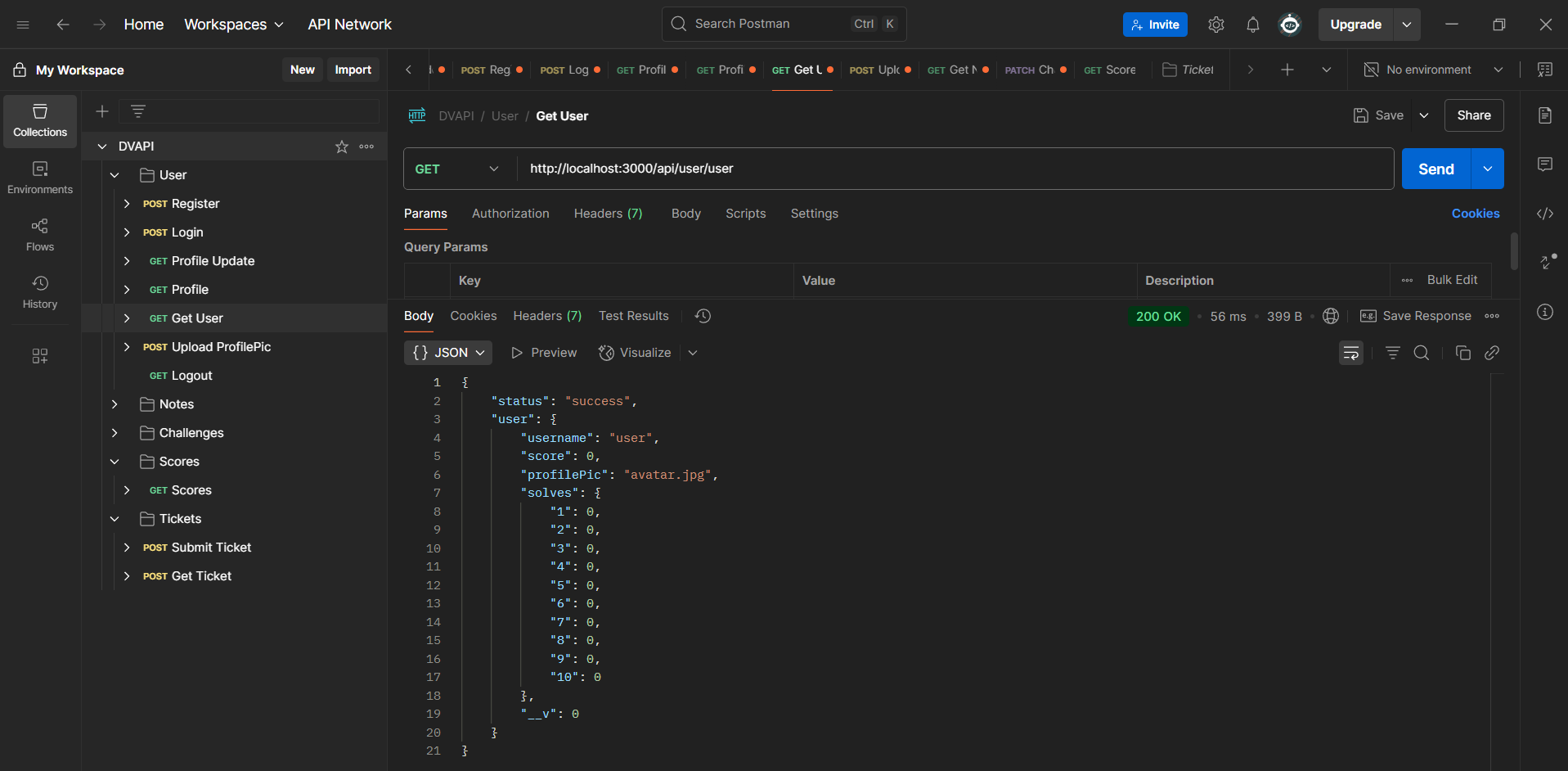
Broken Function Level Authorization occurs when the server fails to properly verify if a user is authorized to perform a specific action (e.g., GET, POST, DELETE) on a given endpoint. Even if UI-level controls restrict these actions, lack of server-side enforcement allows attackers to change the HTTP method and invoke unauthorized functionality.

## Tools Used

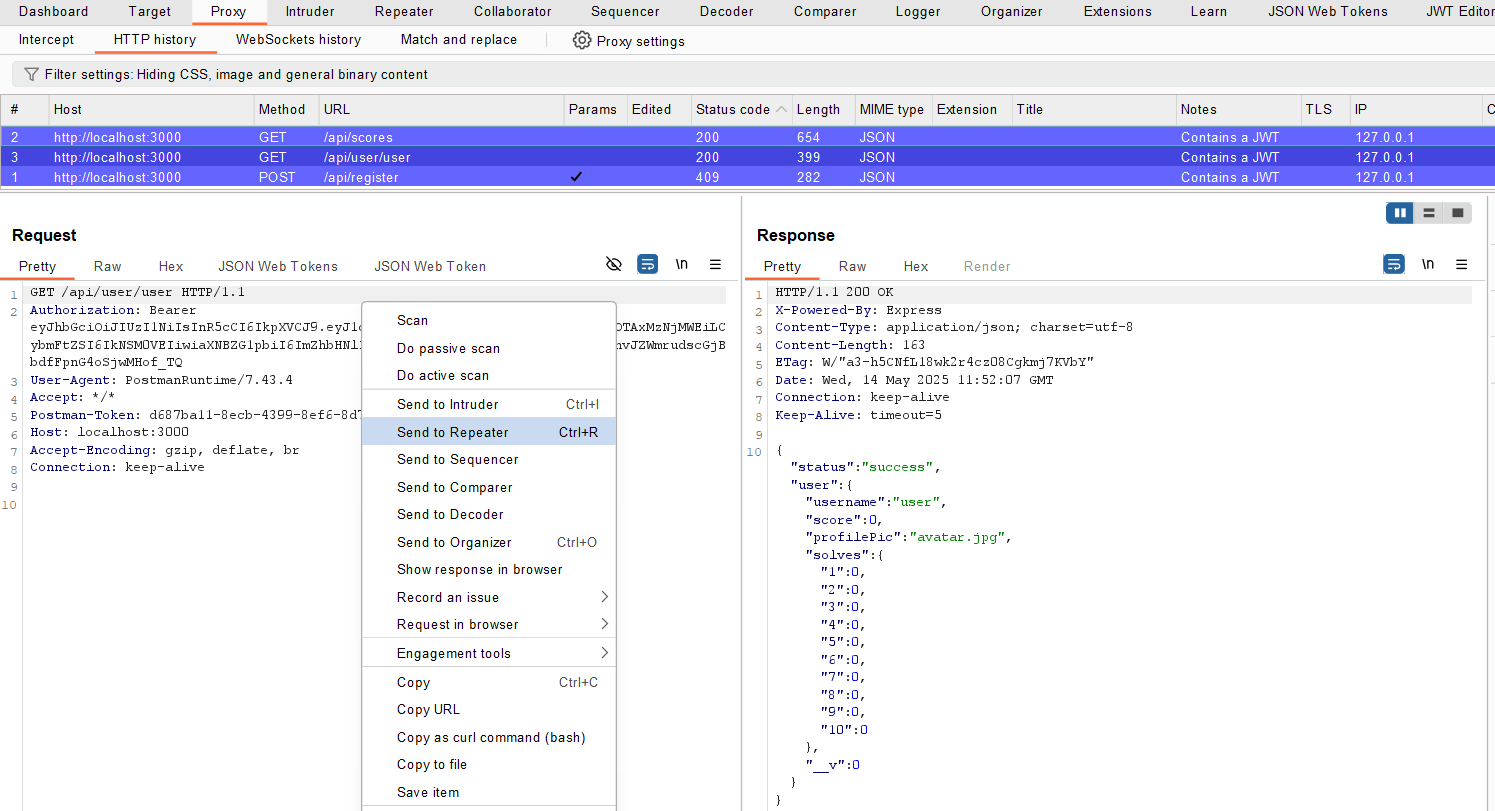
• DVAPI: Hosted locally at http://localhost:3000  
• Postman: Used to send initial API requests and inspect endpoint behavior  
• Burp Suite: Used to intercept, replay, and modify HTTP request methods

## Steps to Reproduce

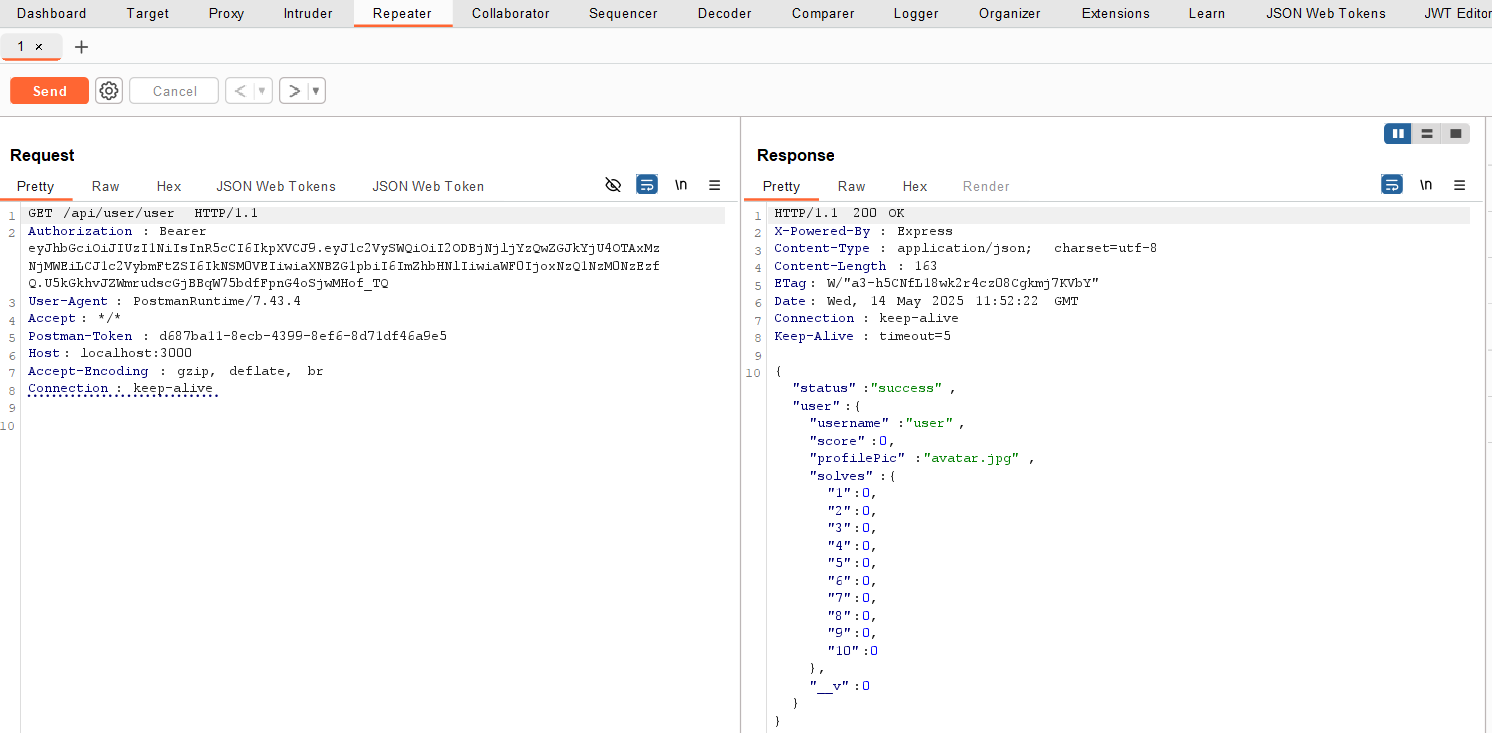
1. A request was sent via Postman to the `/api/getUser` endpoint to retrieve basic user information.



2. The request was intercepted in Burp Suite and forwarded to the Repeater for testing.

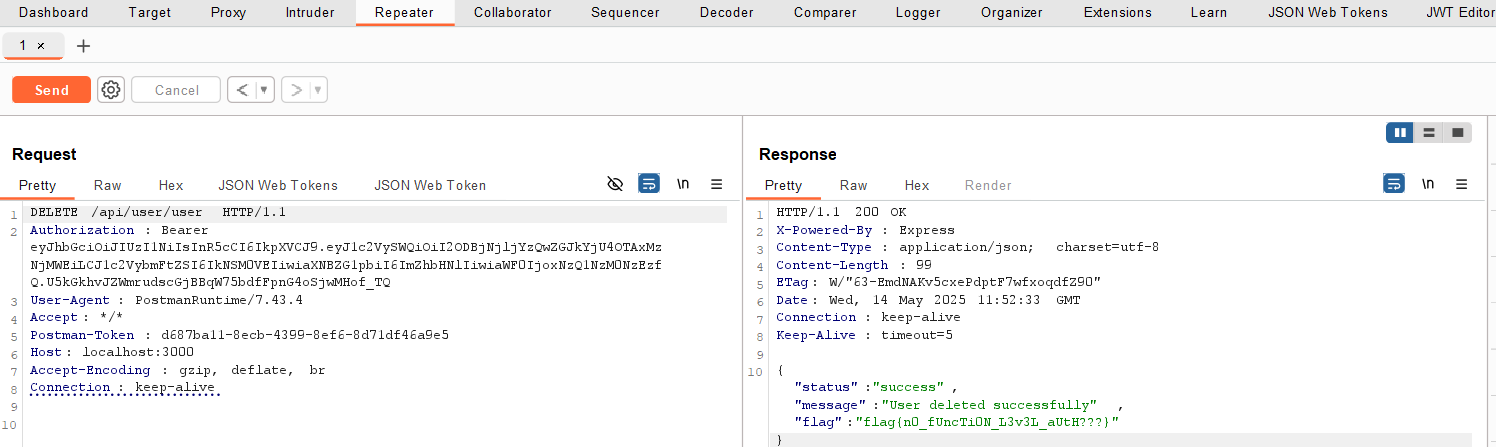


3. In Repeater, the original GET request was inspected to understand its structure and response behavior.



4. The HTTP method was then changed from GET to DELETE in the Repeater. Upon sending the modified request, the API responded with a flag, confirming unauthorized access to privileged functionality.

flag{br0k3n\_funCt10n\_l3v3l\_4uth\_0w4sp}



## Impact

This vulnerability allows users to perform functions they are not authorized to use, such as deleting or modifying resources. Attackers can exploit this to delete accounts, change data, or access privileged operations that should be restricted to admins.

## Mitigation Strategy

• Enforce role-based authorization for every endpoint and HTTP method on the server side.  
• Never rely solely on client-side controls to restrict actions.  
• Validate user permissions explicitly before processing any sensitive function.  
• Implement proper access control checks based on user roles and the action being performed.