DVAPI Vulnerability Assessment Report

# API3:2023 – Broken Object Property Level Authorization

## Objective

To demonstrate exploitation of Broken Object Property Level Authorization in DVAPI (API3:2023), where user-supplied data includes privileged fields not properly restricted, leading to unauthorized manipulation of sensitive properties.

## Overview of DVAPI

DVAPI (Damn Vulnerable API) is a test environment replicating vulnerable APIs based on the OWASP API Security Top 10. It allows hands-on experience with real-world attack vectors like property-level authorization flaws, token issues, and more.

## Vulnerability Description

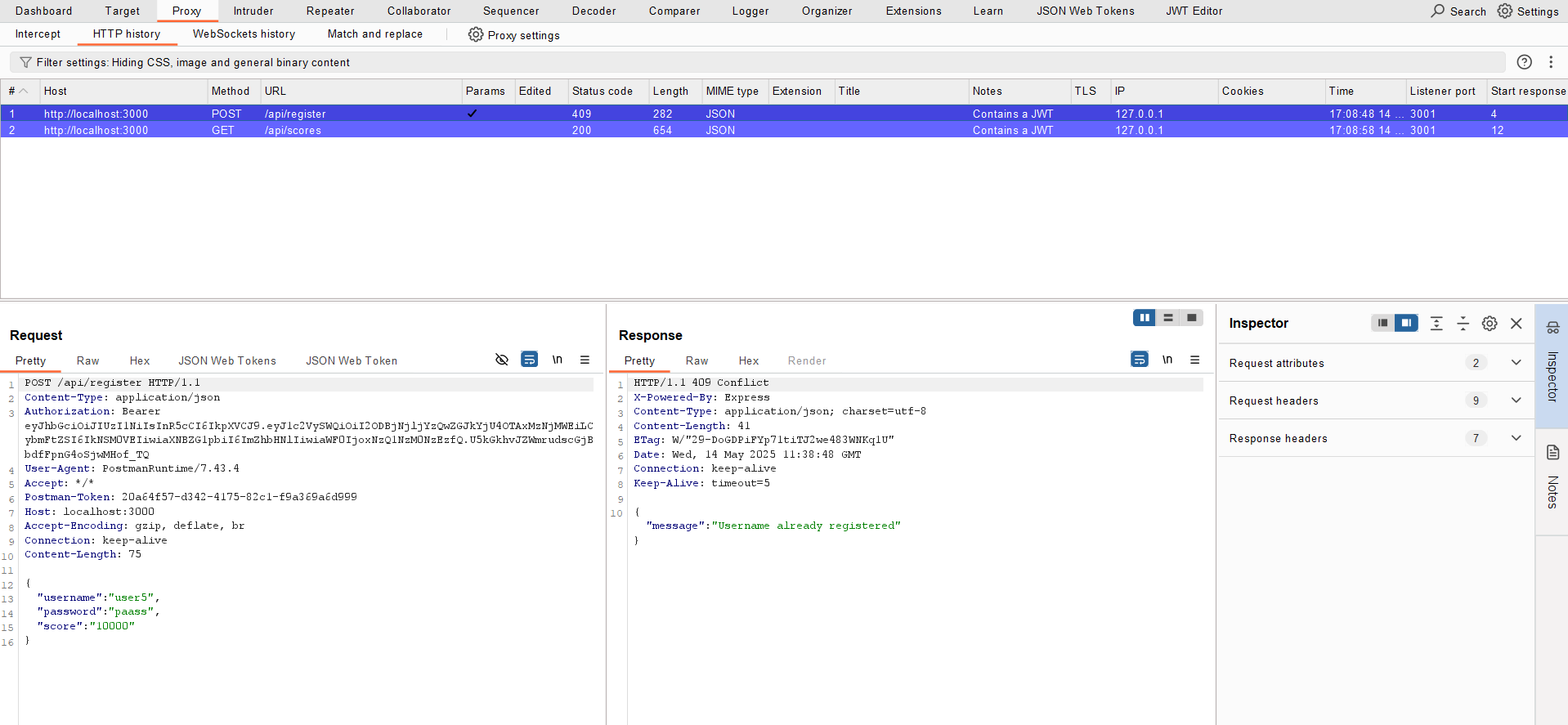
Broken Object Property Level Authorization occurs when APIs allow users to modify or submit object properties they should not have access to. If individual fields (e.g., 'score', 'role') aren't restricted server-side, attackers can alter or inject values that should be immutable or privileged.

## Tools Used

• DVAPI: Local instance hosted at http://localhost:3000  
• Postman: For constructing and sending API requests  
• Burp Suite: For inspecting and manipulating API calls

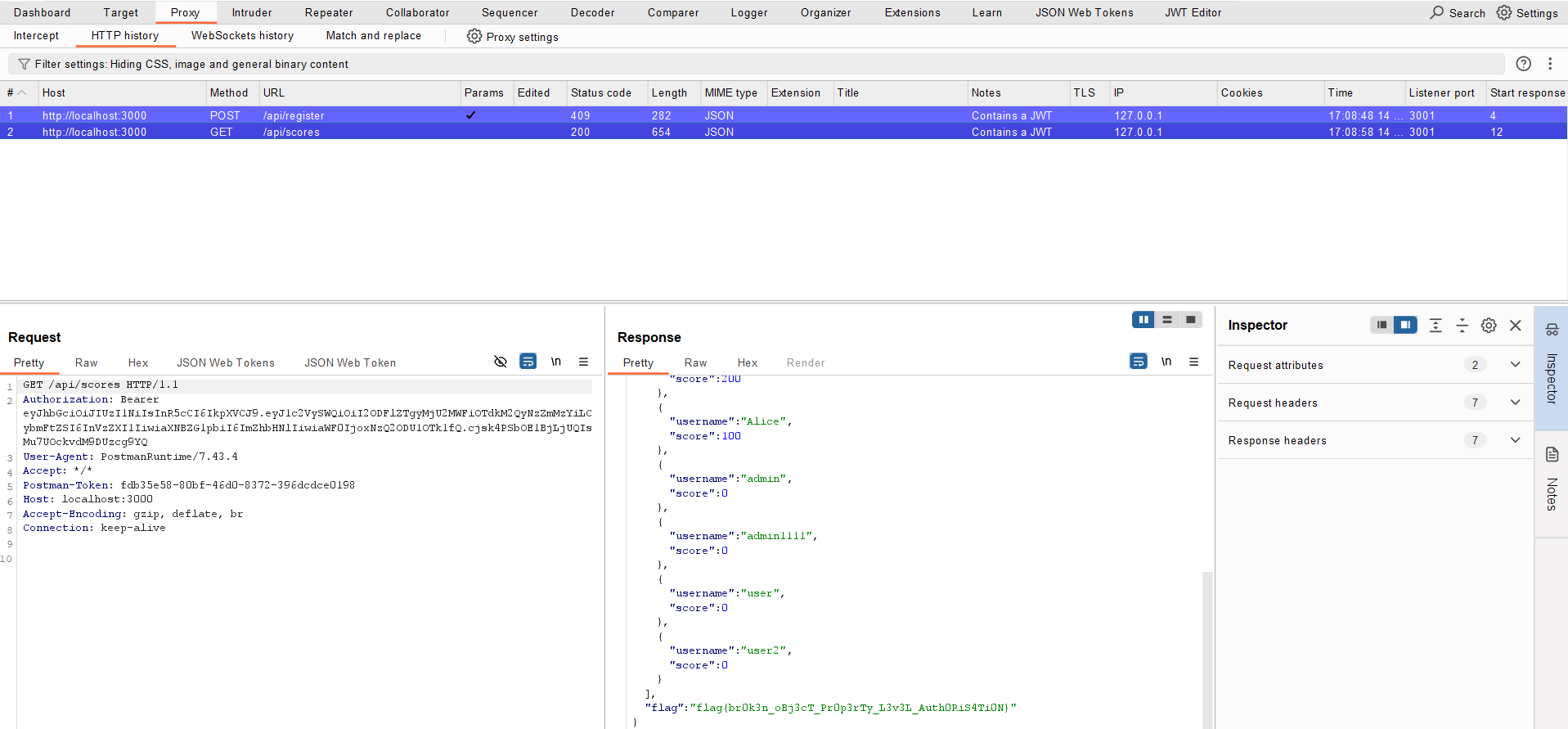
## Steps to Reproduce

1. During user registration, the API request body includes a `score` parameter. By manually injecting a high score (e.g., 10000) into the JSON payload, the backend accepts it without proper authorization control.



2. After registration, a request is made to the `/api/score` endpoint. The inflated score is reflected in the response, and the API reveals the flag:

flag{br0k3n\_oBj3cT\_Pr0p3rTy\_L3v3L\_Auth0RiS4Ti0N}



## Impact

This vulnerability allows users to set privileged or restricted values (like score, isAdmin, etc.) during registration or updates. If such data is used for access control or ranking, it can undermine system integrity, expose sensitive information, or unfairly elevate user status.

## Mitigation Strategy

• Enforce strict server-side filtering and validation of object properties.  
• Never trust client-supplied fields related to permissions, scores, or roles.  
• Implement schema validation and allow-listing of fields per user role.  
• Log and monitor unusual or unauthorized object property changes.