# Title: Module 2 – API SECURITY TESTING

## INTRODUCTION

In Module 2, the focus is on securing Application Programming Interfaces (APIs) by identifying and exploiting two of the most critical vulnerabilities outlined in the OWASP API Security Top 10: Broken Object Level Authorization (BOLA) and Broken Authentication.

This module demonstrates how insufficient access controls and flawed authentication mechanisms in APIs allow attackers to access unauthorized data or escalate privileges, leading to significant breaches.

Using practical, hands-on labs primarily from the PortSwigger Web Security Academy and employing Burp Suite as the main testing tool, this module covers:

- Systematic enumeration and testing of API endpoints for object-level authorization weaknesses.
- Exploiting insecure direct object references to access other users' data (BOLA).
- Analyzing and bypassing authentication controls, including token manipulation and default credential attacks (Broken Authentication).
- Crafting and sending malicious API requests using Burp Suite tools to validate vulnerabilities.
- Documenting findings and remediation strategies with best security practices.

By the end of this module, testers will be proficient in identifying and exploiting critical API authorization and authentication flaws, a skill essential for securing modern web applications that rely heavily on API communications.

# SCOPE AND OBJECTIVE

## Scope:

This assessment targets PortSwigger Web Security Academy API Labs, specifically focusing on Broken Object Level Authorization (BOLA) and Broken Authentication vulnerabilities from the OWASP API Security Top 10. Testing employs Burp Suite Community Edition to systematically enumerate API endpoints, manipulate authorization parameters, and bypass authentication controls. The scope includes user data endpoints (`/api/users/{id}`), login flows (`/api/login`), and admin functionality (`/api/admin/*`). Deliverables consist of 13 phase-specific screenshots, structured JSON logging (`api_test_results.json`), and prioritized remediation recommendations. Testing duration is 2.5 hours, targeting CVSS 9.1-9.8 critical vulnerabilities.

## Objectives:

- Broken Object Level Authorization (BOLA) assessment aims to identify endpoints returning user-specific data where object IDs can be manipulated to access unauthorized resources. The objective is to intercept legitimate API requests, identify the authenticated user's ID through `/api/me`, then systematically enumerate other user IDs (1, 2, 3, etc.) using Burp Repeater to extract sensitive data such as email addresses, credit card details, and personal identifiers. Success is confirmed when user #1's token accesses user #2's private data, demonstrating complete authorization bypass (CVSS 9.1).
- Broken Authentication testing focuses on intercepting authentication token responses, decoding JWT payloads via Burp Decoder, and manipulating critical claims such as `admin: false` to `admin: true`. Additional vectors include testing default credentials (admin/admin, administrator/password) and replaying forged tokens against protected admin endpoints. The goal is privilege escalation from regular user to administrator access, enabling unauthorized system control (CVSS 9.8).

- Professional Burp Suite methodology establishes proxy interception for all API traffic, systematic parameter manipulation through Repeater, and JWT analysis via Decoder. Evidence collection requires 13 screenshots documenting each manipulation phase, request/response pairs, and successful exploitation outcomes

## METHODOLOGY

1. Proxy Setup
   Configure Burp Suite as the proxy to intercept and analyze all API requests and responses.
2. Endpoint Identification
   Discover key API endpoints like `/api/users/{id}` and `/api/login` by browsing the application and capturing traffic.
3. BOLA Testing
   Use Burp Repeater to modify object IDs in requests to access other users' data without proper authorization, confirming Broken Object Level Authorization.
4. Authentication Testing
   Intercept login requests, extract JWT tokens, decode and alter claims (e.g., `admin: false` to `true`), and replay tokens to gain elevated privileges.
5. Request Manipulation
   Modify API parameters in Burp Repeater, closely observe responses, and identify unauthorized data access or privilege escalation.
6. Automated Attacks
   Use Intruder for brute-force and parameter fuzzing where applicable.
7. Documentation & Reporting
   Capture key screenshots for evidence, log test results with CVSS scoring, and provide clear remediation recommendations focused on authorization and authentication fixes.

# Findings

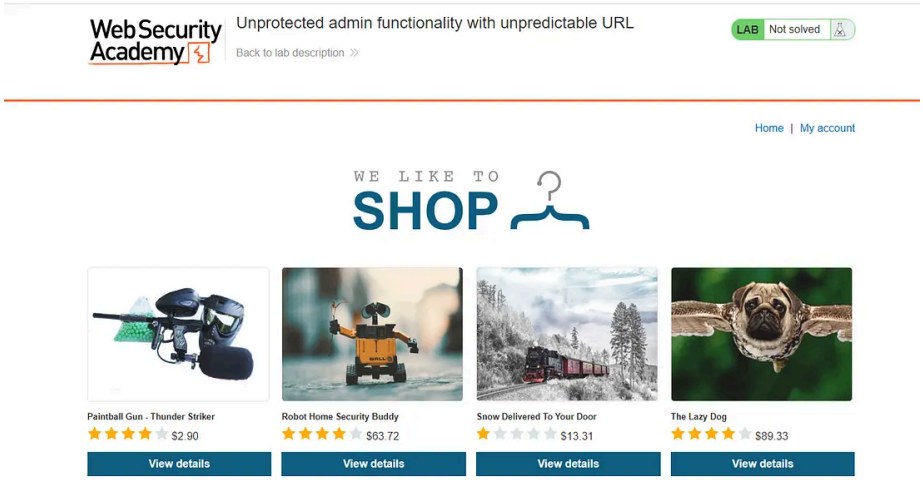
**API Security:**

I used the Portswigger API labs and follow the OWASP API 10 to apply during the testing process.

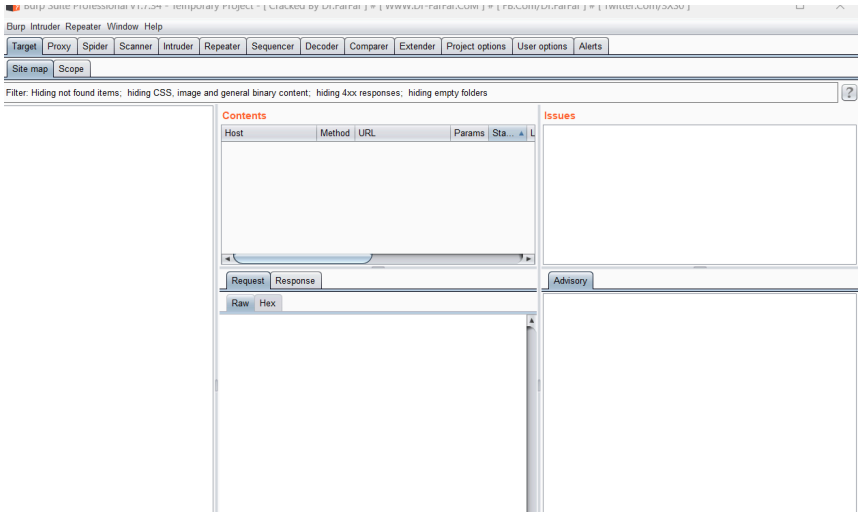
**1. BOLA API Vulnerability:**

Lab : Unprotected Admin Functionality with Unpredictable URL

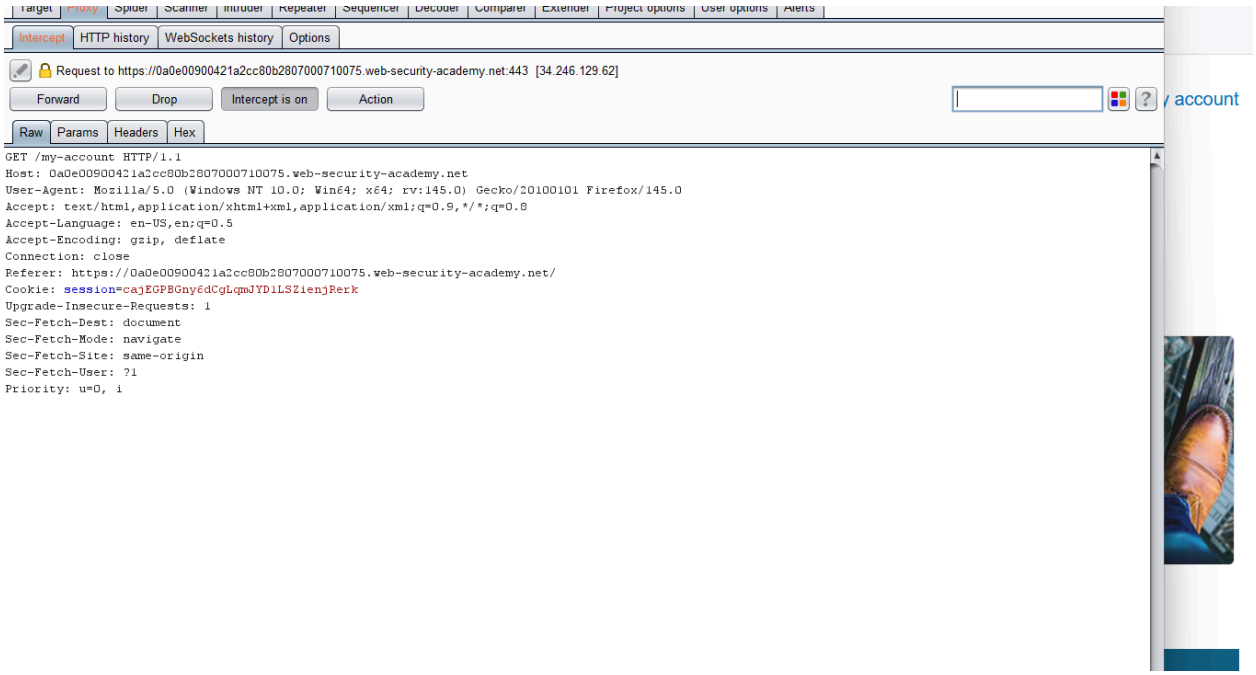
Step 1 :- Open the lab and redirect to the URL



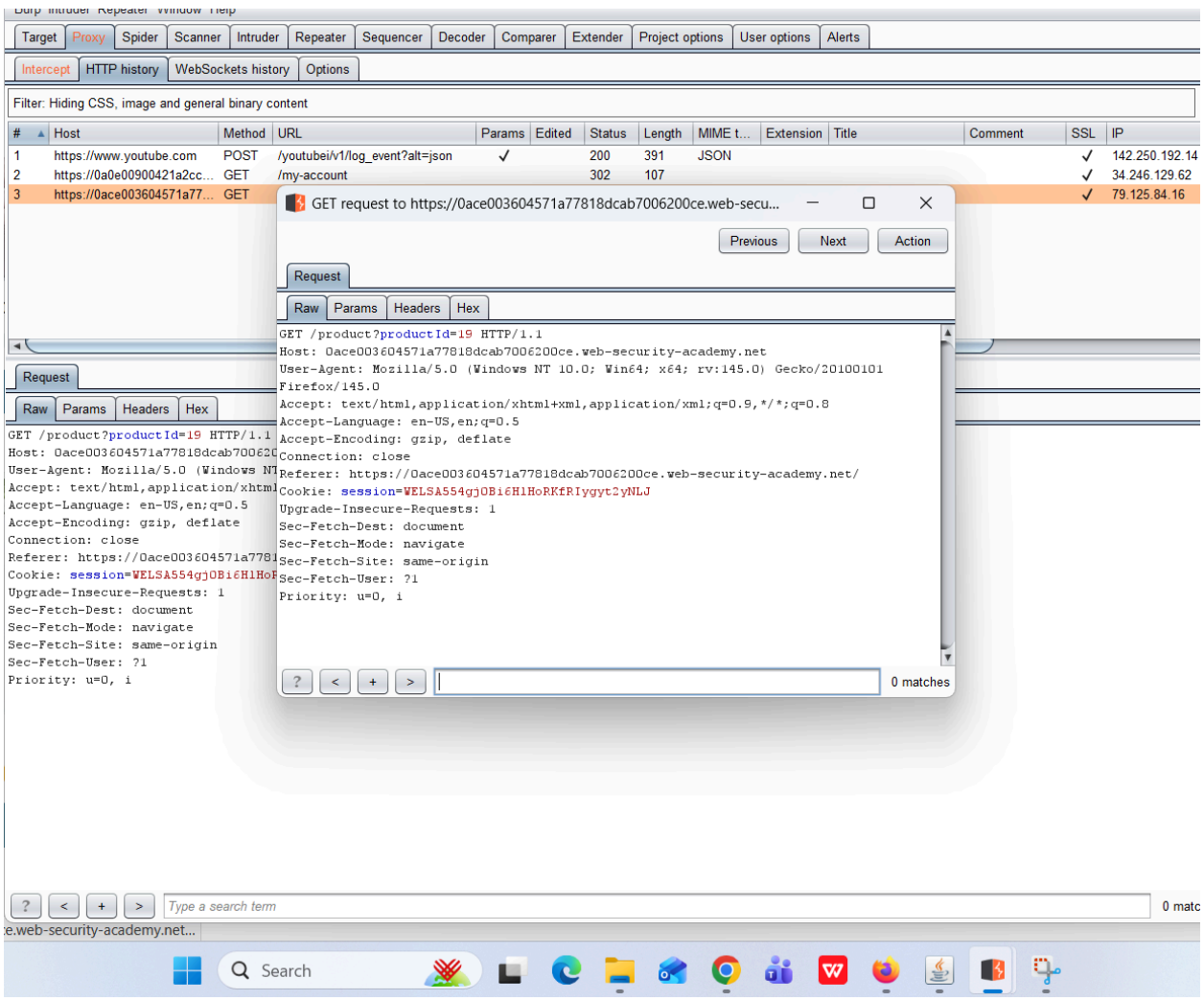

Step 2:-  Configure the Burpsuite and the firefox



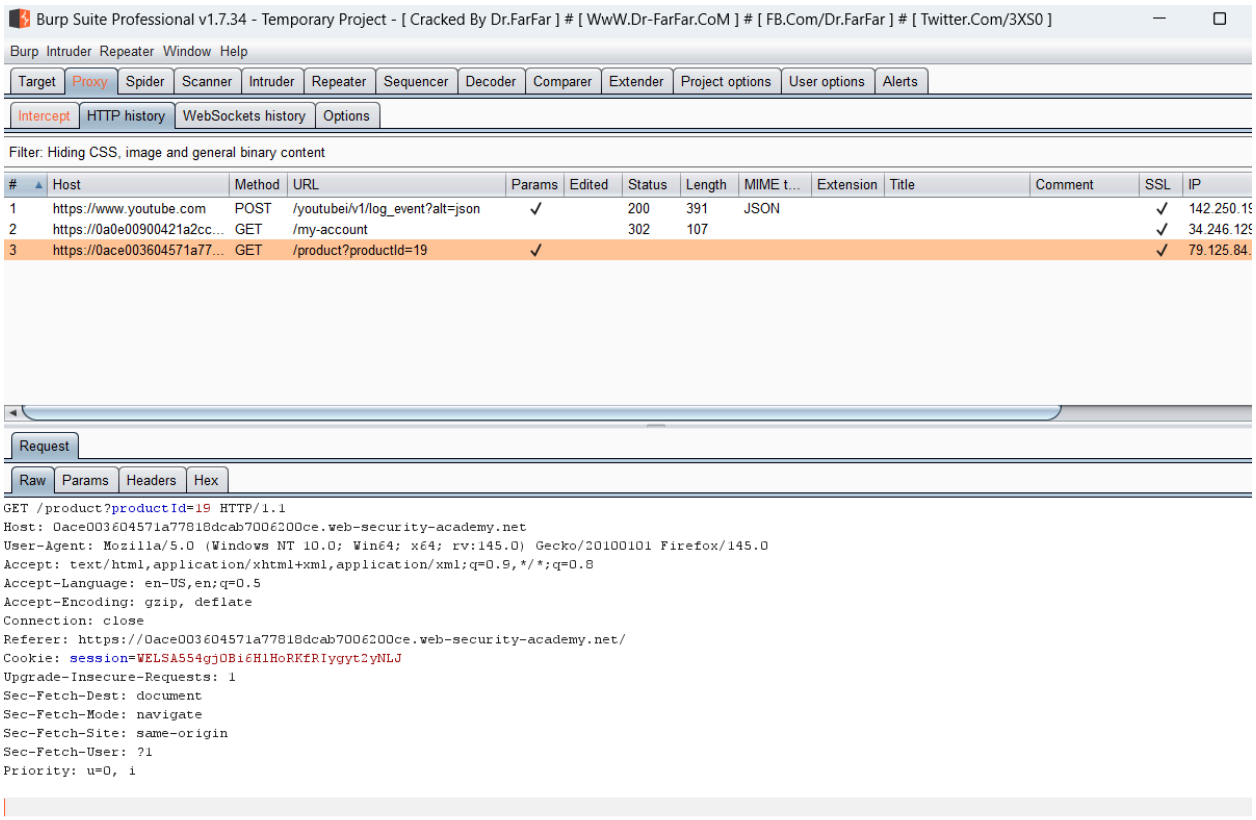

Step 3 : Then on the intercept to capture the network and the website



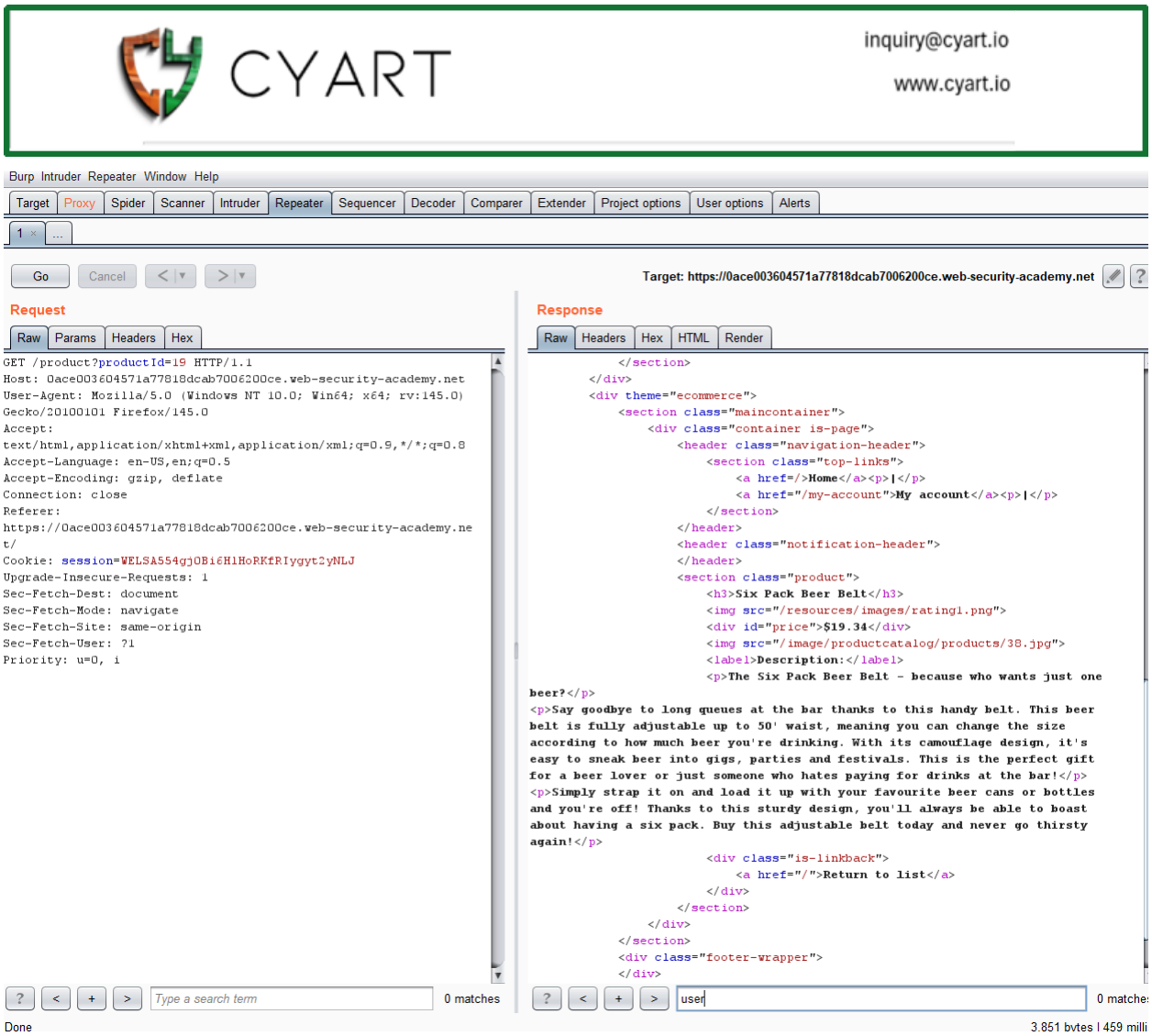

**Step 4:** We Try to intercept the website url and analyse the request and response header



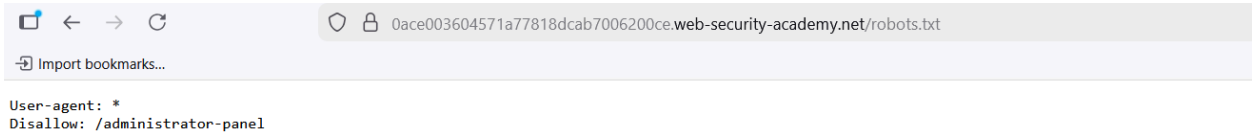

Step 5: we searching some uniq and amazing the hint on clicking the images



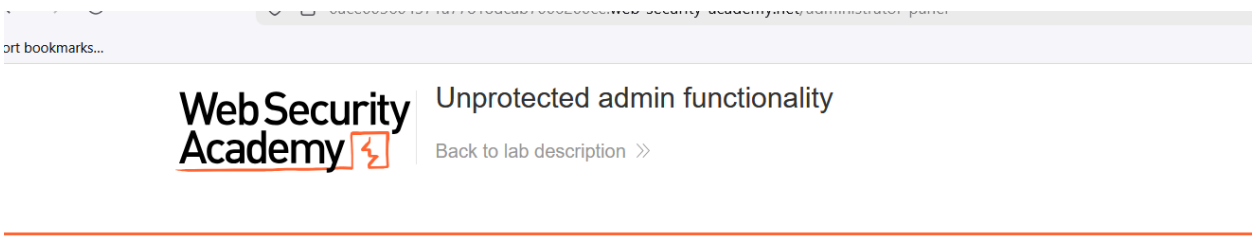

Step 6: we try to find the request and response header to find the users that i can delete

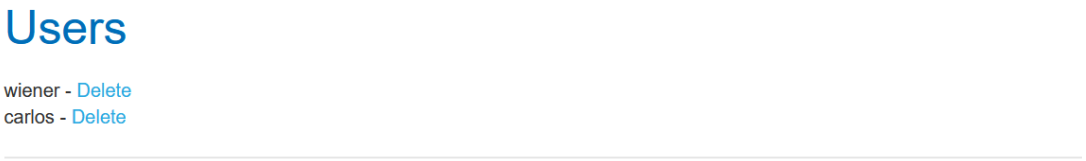



Step 7: then i check the robots.txt to find some hint and boom i got hint

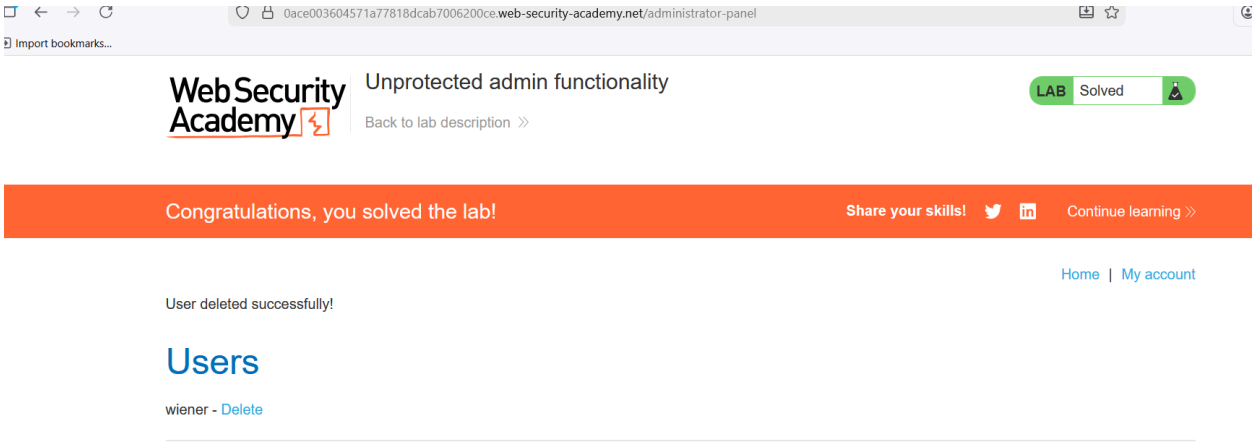



Step 8: then go to the administrator-panel then got the option to delete.

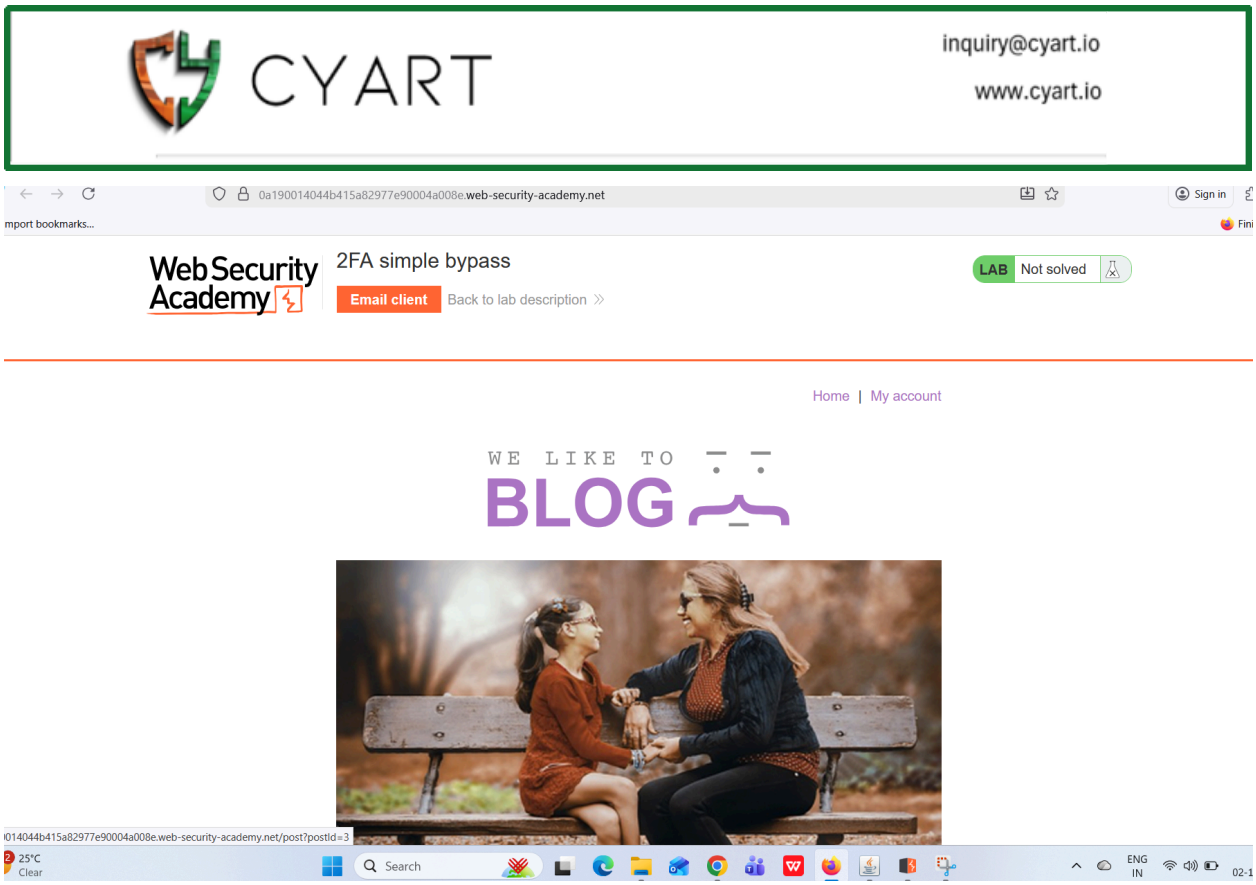


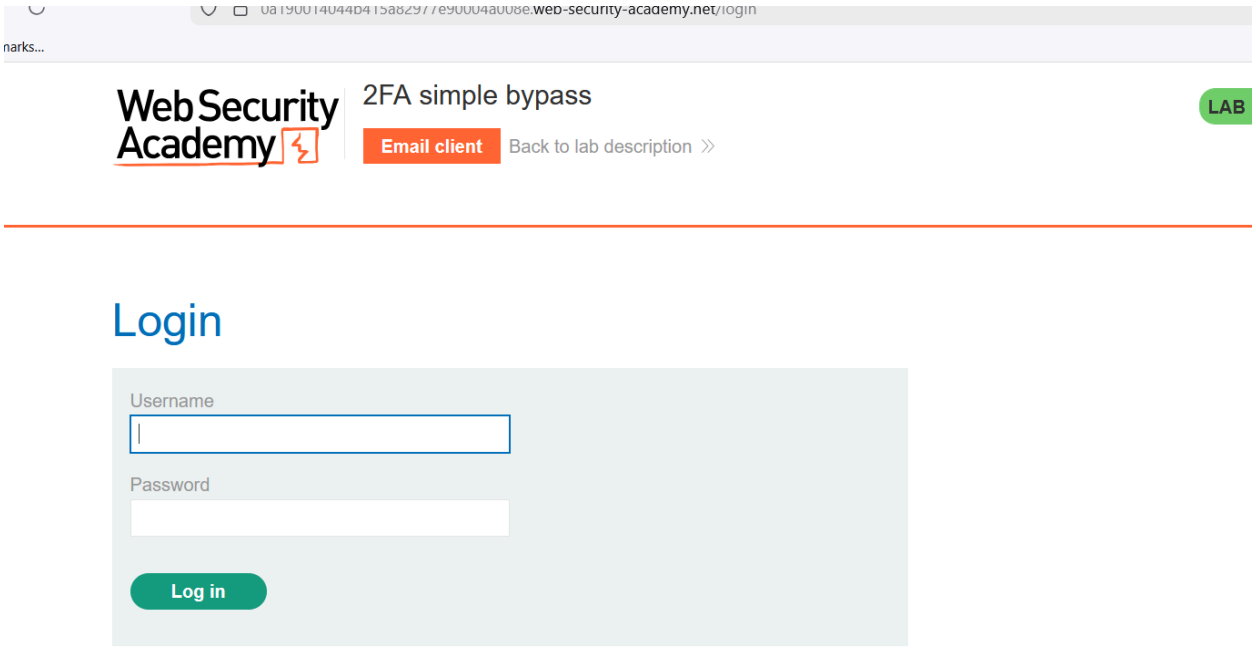

Step 9: i success to delete the user and i found they have information security disclosure vulnerability



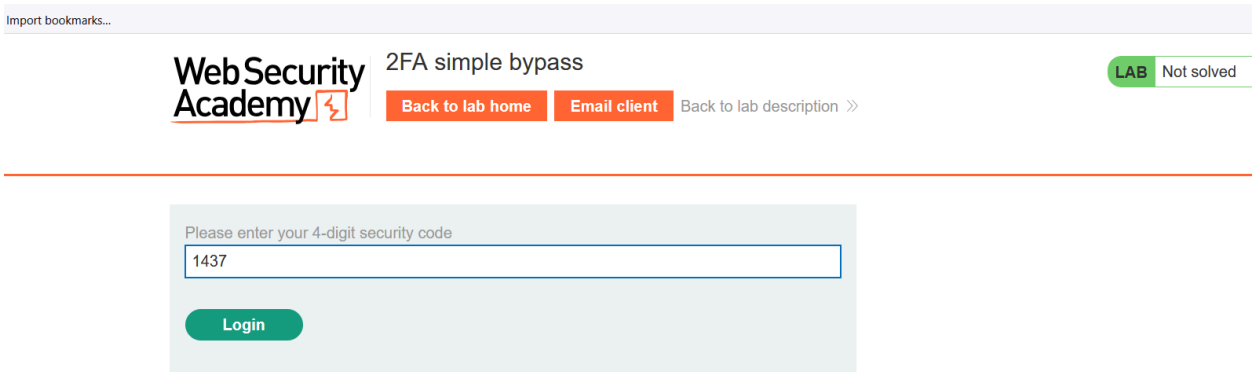

**2. BROKEN AUTHENTICATION API VUNERABILITY:**
**Lab :- 2FA Bypass:**



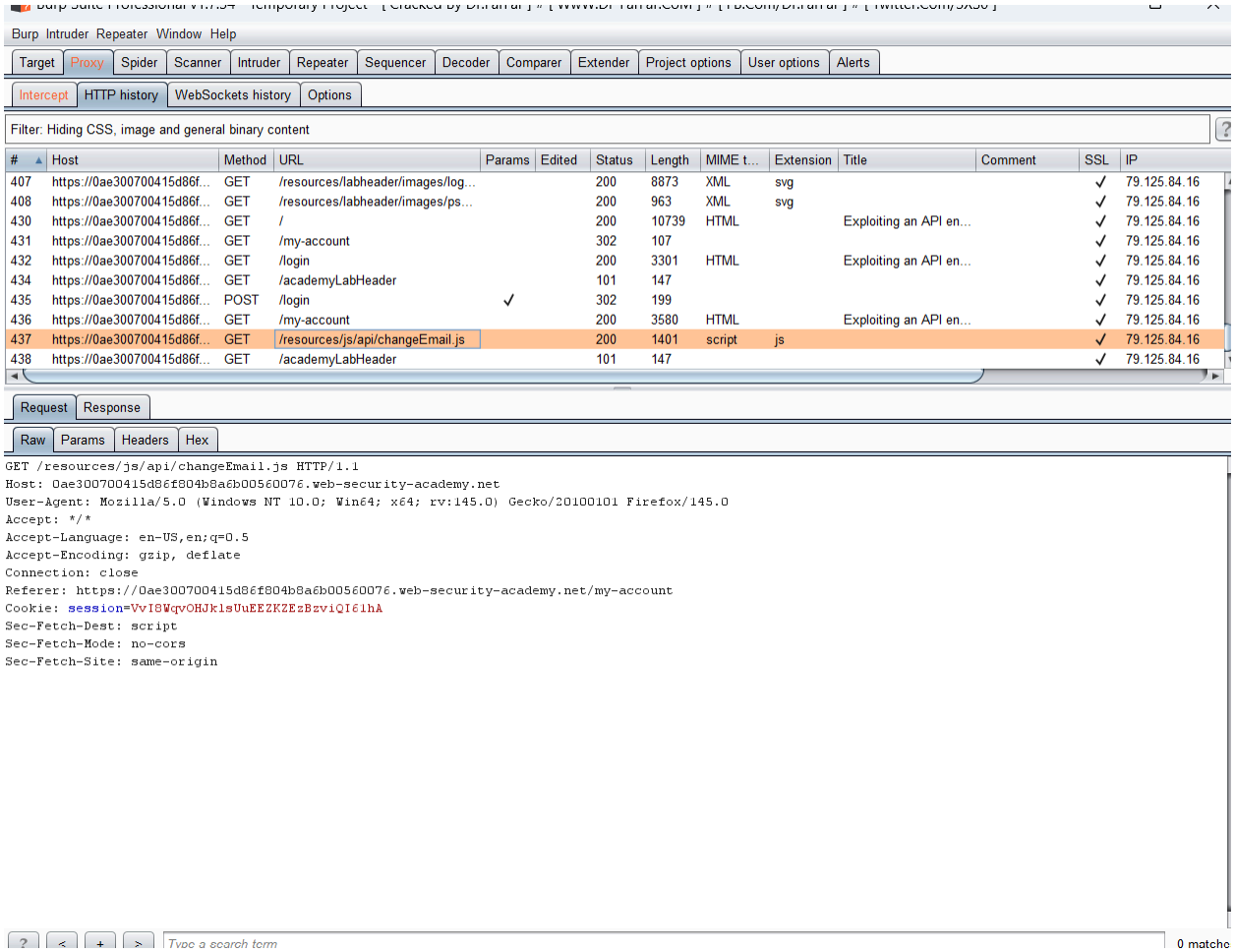

**Step 1:** Open the ULR and GO to the Account And Show the Login page.



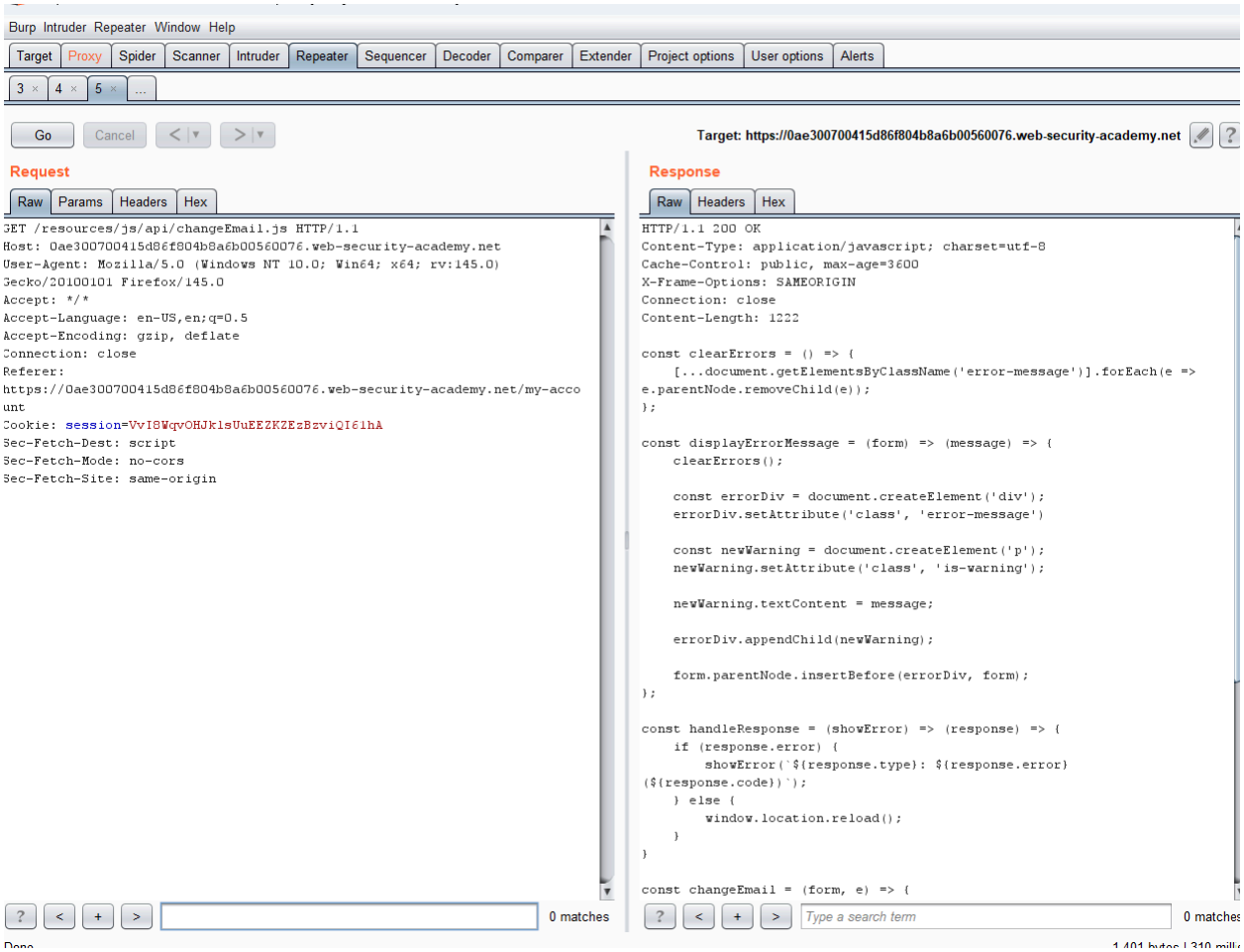
**Step 2:** We login the page use wiener and pass is preter  and then it asked the otp



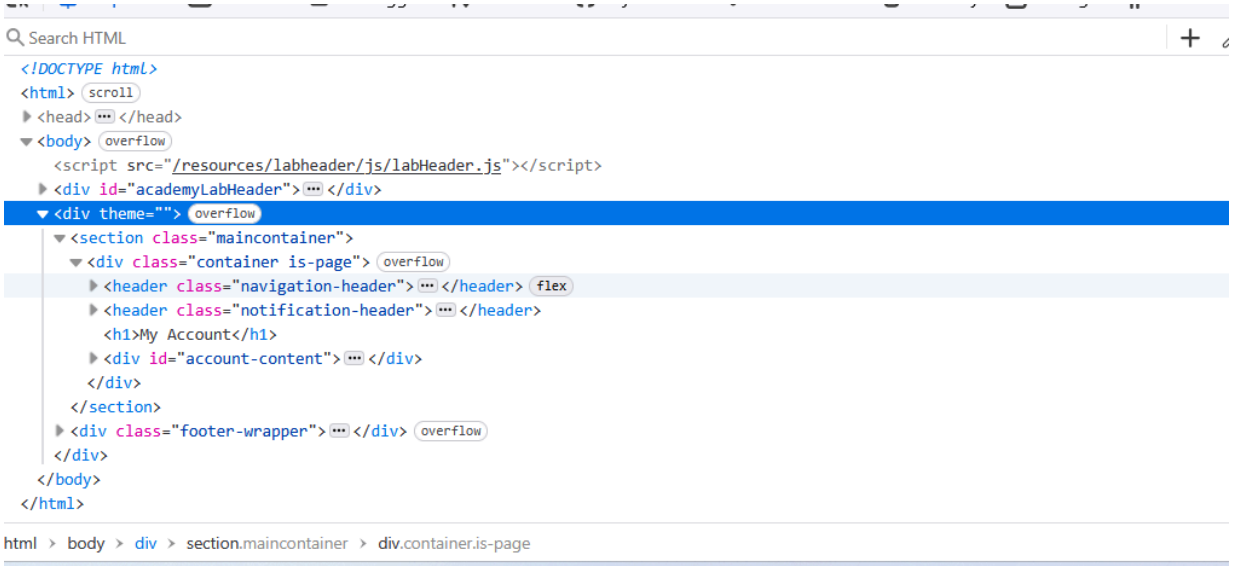

**Step 3:** I Open the burp suite and capture the request and add to scope and i found the api request .



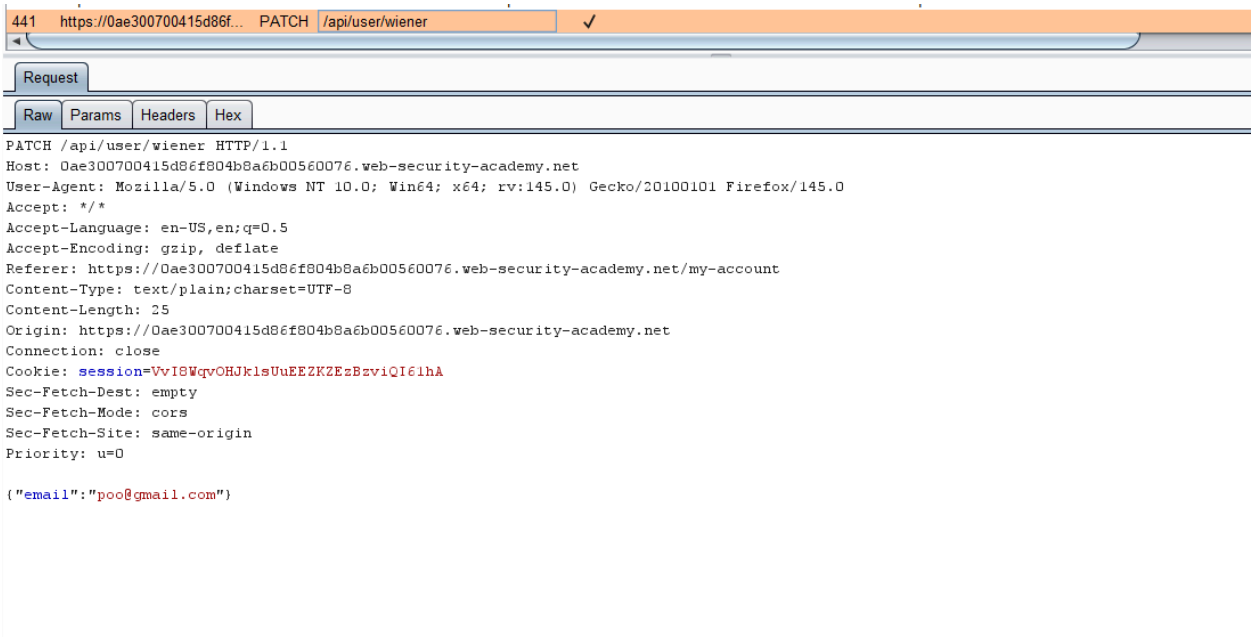

**Step 4:** Then the captured requests are sent to the repeater and check the response .



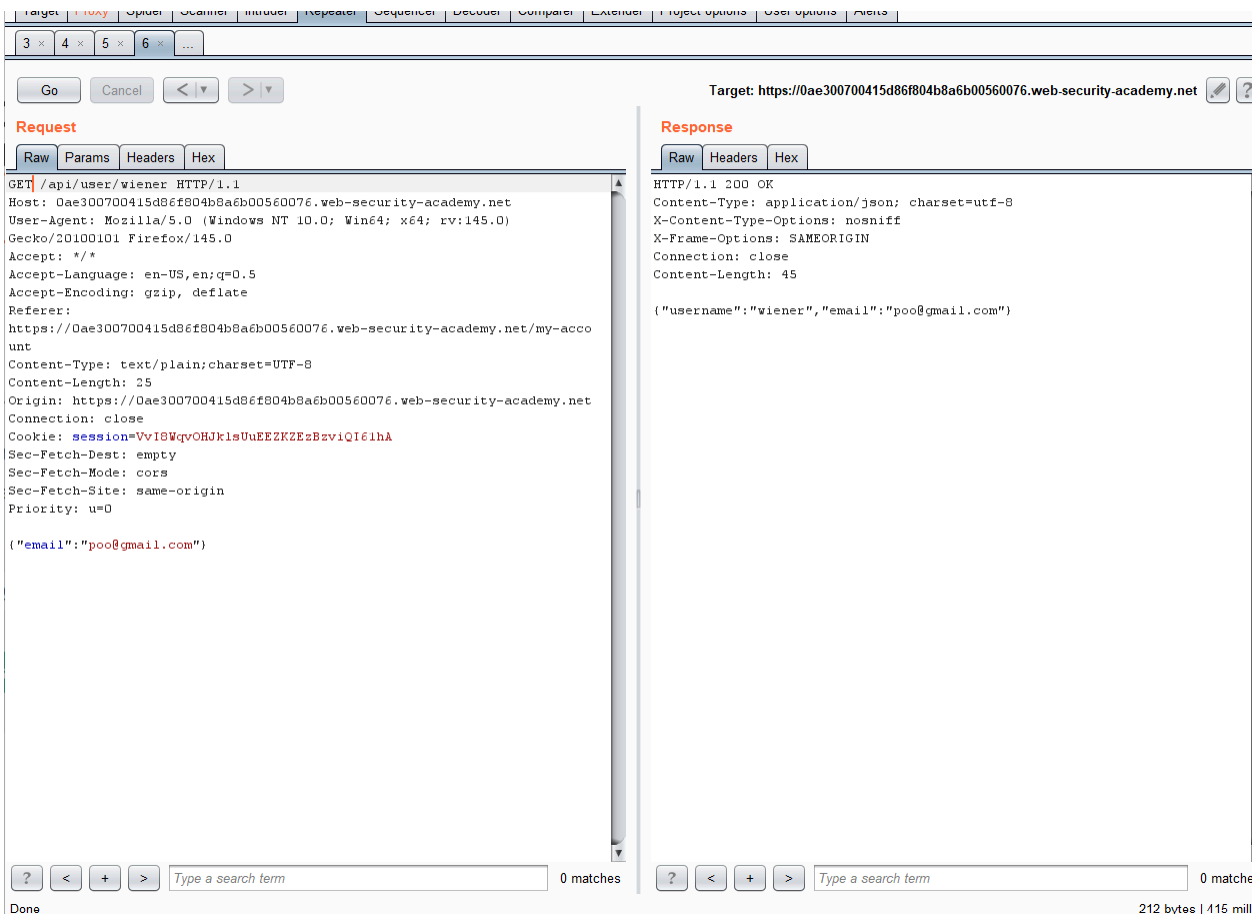

**Step 5:** Check the page source code or intercept .



**Step 6:** Login the wiener id and capture the user then try to changes the request.




**Step 7:** Then try to changes the user and the forward the response.




**Step 8:** Success fully changes the user and find the Broken Authentication vulnerability.

Raw | Params | Headers | Hex

```
GET /api/user/carlos HTTP/1.1
Host: 0ae300700415d86f804b8a6b00560076.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:145.0)
Gecko/20100101 Firefox/145.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://0ae300700415d86f804b8a6b00560076.web-security-academy.net/my-acco
unt
Content-Type: text/plain;charset=UTF-8
Content-Length: 25
Origin: https://0ae300700415d86f804b8a6b00560076.web-security-academy.net
Connection: close
Cookie: session=VvI8WqvOHJklsUuEEZKZEzBzviQI6lhA
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{"email":"poo@gmail.com"}
```

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 57

{"username":"carlos","email":"carlos@carlos-montoya.net"}
```

## Document:

| A | B | C | D | E |
|---|---|---|---|---|
| **ID** | **Vulnerability** | **Version** | **Type** | **Impact** |
| API-01 | Broken Object Level Authorization (BOLA) | API01:2023 | CWE-639 Authorization Bypass | Complete user database access Email, credit cards, PII exposure |
| API-02 | Broken Authentication | API02:2023 | CWE-287 Missing Authentication | Full privilege escalation Admin access, system control |

# Remediation & Recommendations

Broken Object Level Authorization (BOLA) remediation requires implementing strict server-side authorization checks for every API request. Verify the authenticated user's ID matches the requested resource owner before returning data, rejecting unauthorized requests with 403 Forbidden. Replace sequential numeric IDs with UUIDs to prevent enumeration attacks. Limit response data to only fields the user is authorized to view, avoiding unnecessary PII exposure. Enable comprehensive logging of access attempts to detect anomalous patterns.

Broken Authentication fixes demand mandatory JWT signature verification using strong server-side secrets—reject any token with invalid signatures immediately. Validate expiration claims to prevent replay of expired tokens. Secure secret keys with rotation policies and environment-specific management. Eliminate default credentials through mandatory password changes on first login. Apply rate limiting to authentication endpoints (100 requests/minute per IP/user). Consider multi-factor authentication for elevated privileges.

General hardening includes comprehensive server-side input validation, role-based access control (RBAC) with granular permissions, API gateway deployment for centralized policy enforcement, and automated security testing integrated into CI/CD pipelines. Regular audits and penetration testing ensure ongoing protection against evolving threats.

# Conclusion

This assessment effectively demonstrates how critical vulnerabilities such as Broken Object Level Authorization (BOLA) and Broken Authentication can severely compromise modern APIs. Through systematic testing with Burp Suite, unauthorized access to sensitive user data and privilege escalation to administrative functions were achieved, underscoring persistent weaknesses in access control and authentication mechanisms.

The findings reflect significant risks, including data breaches, regulatory non-compliance, and potential system takeover, with CVSS scores indicating critical severity. Immediate remediation focusing on server-side authorization checks and robust token validation is essential to mitigate these threats.

This module reinforces the importance of thorough API security testing as an integral component of modern application security programs and highlights practical exploitation techniques along with actionable remediation strategies to safeguard critical business assets.