**OWASP API Top 10 (2023) - Detailed Explanation with issue and practical**

**Vulnerable Target – crAPI**

**API 01:2023 —** Broken Object Level Authorization (BOLA)

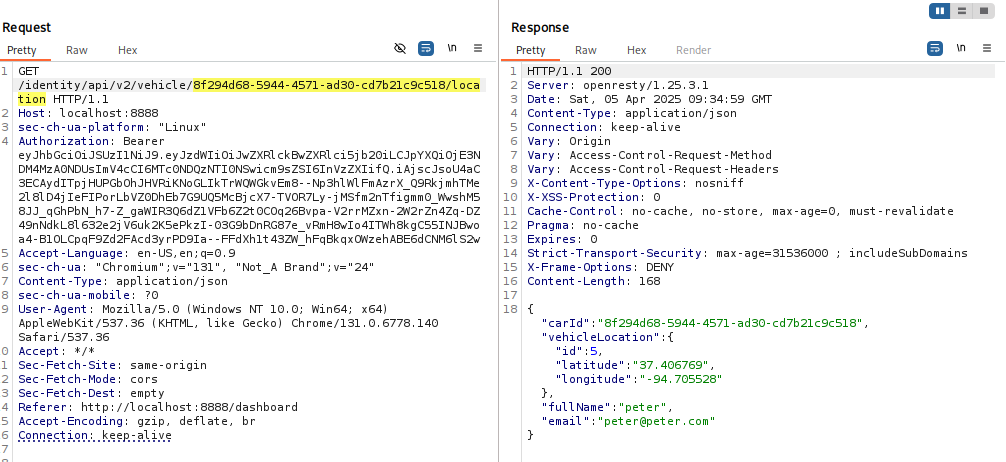
**🔍 Issue:**

APIs expose object IDs in requests.

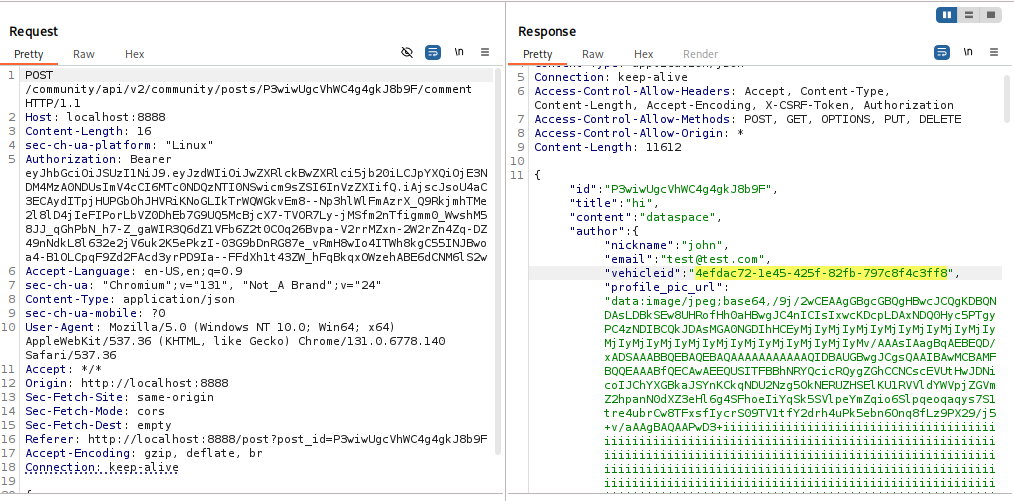
Attackers modify object IDs to access unauthorized resources**.**

**🔨 Exploitation :**

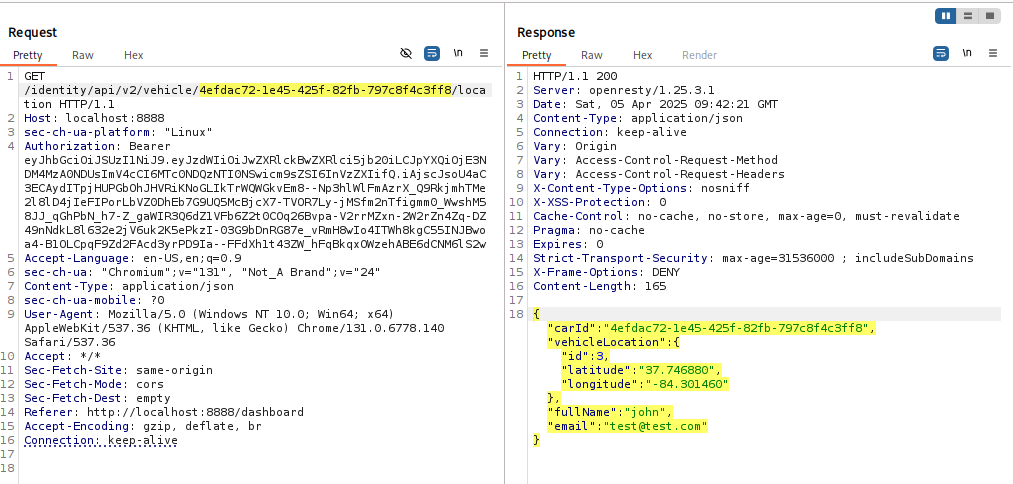
First logging in the application and intercept the location end point

****

**And go to the community section for test bola and intercept this request and I copied vehicle id**

****

**And copied the vehicle id change with real vehicle id and leak the information(location) of other user you can see this here this is confirm to bola**

****

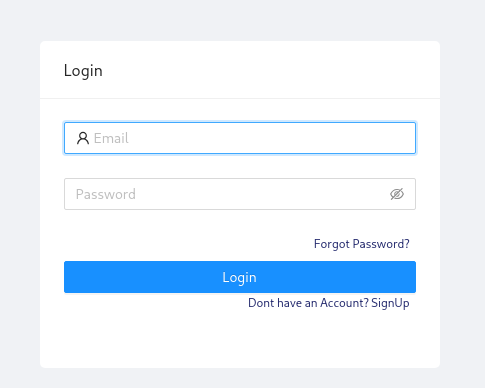
**🛑 API 02:2023 —** Broken Authentication

**🔍 Issue:**

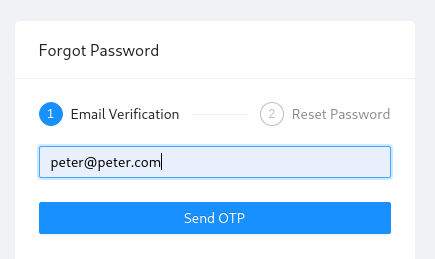
Weak authentication allows attackers to bypass login.

Issues include default passwords, no rate limiting, or JWT token flaws**.**

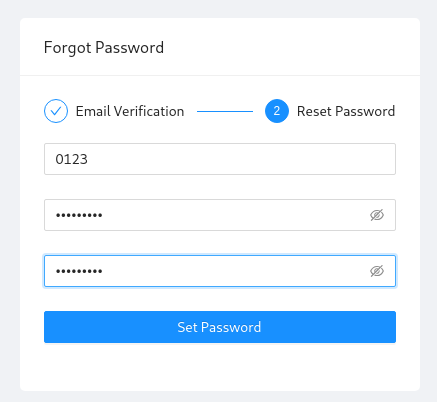
Click on forget password

****

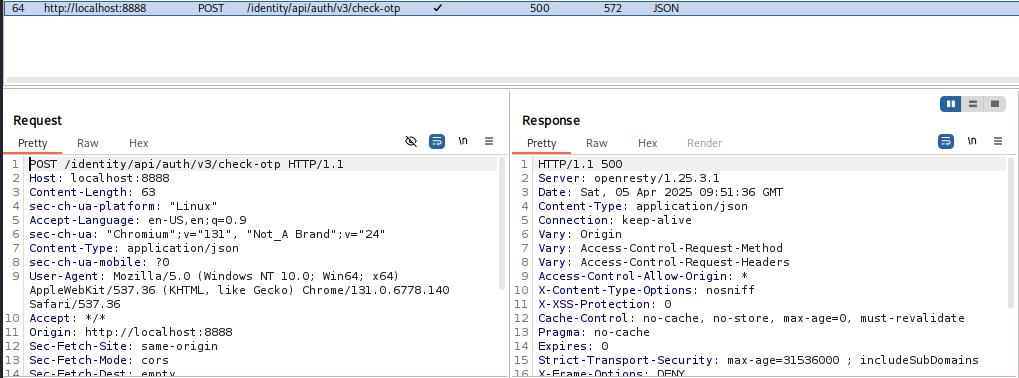
Enter the email

****

Enter the wrong otp

****

**and intercept this request**

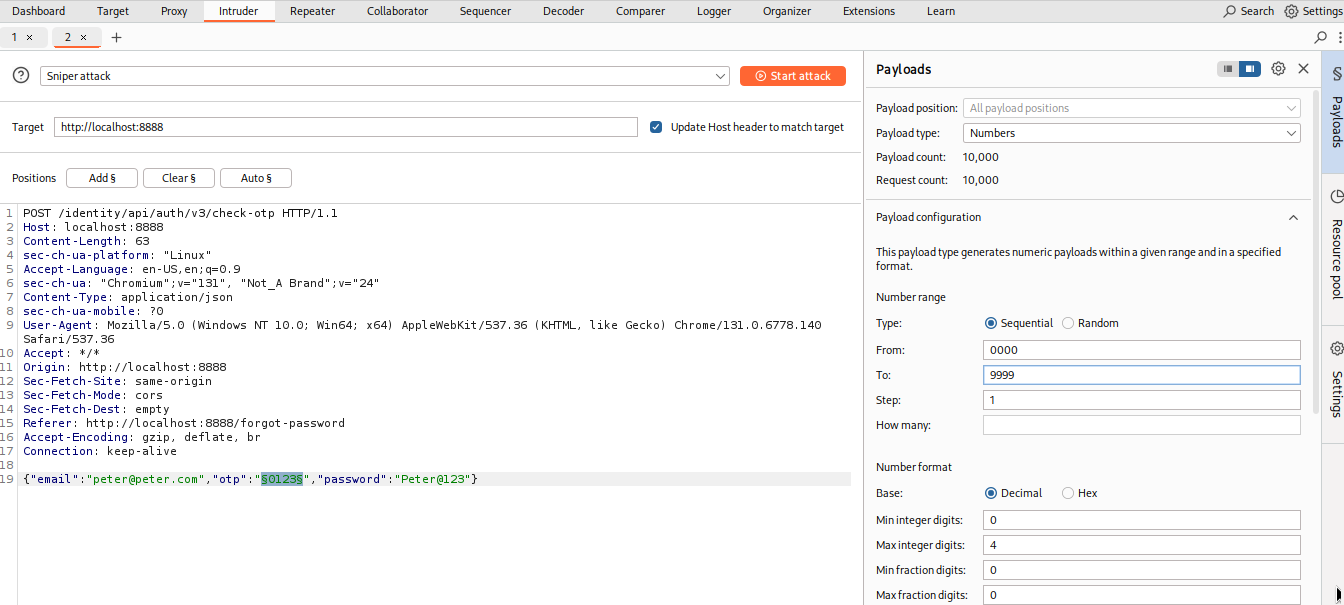


**And send to the intruder for otp bruteforce and set the payload**

**Attack type; sniper**

**Otp set ; 0000-9999**

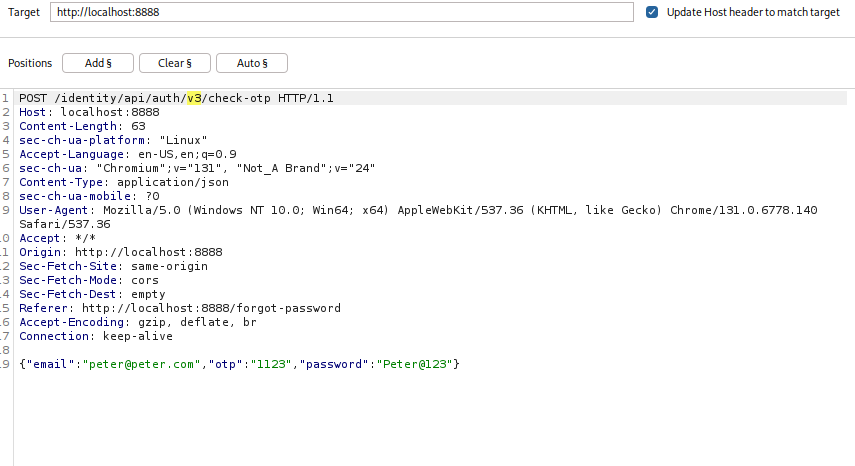
**Start the attack**

****

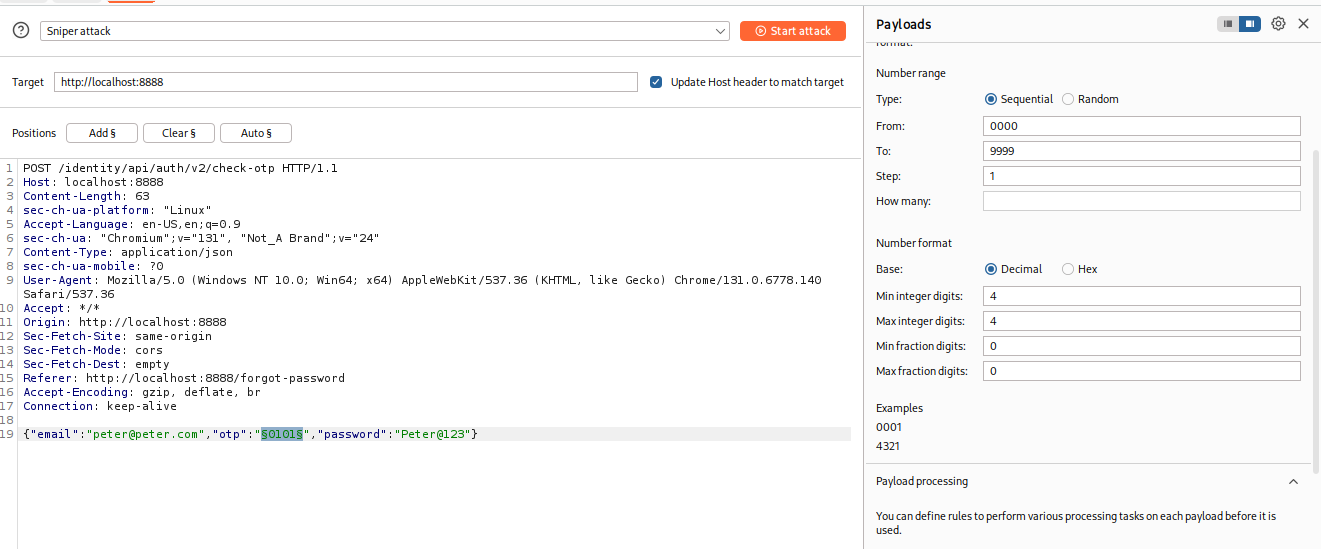
**One thing always remembers every time when you start the attack uncheck the URL-encode**

****

Here my attack is fail because of this api using v3 this not vulnerable

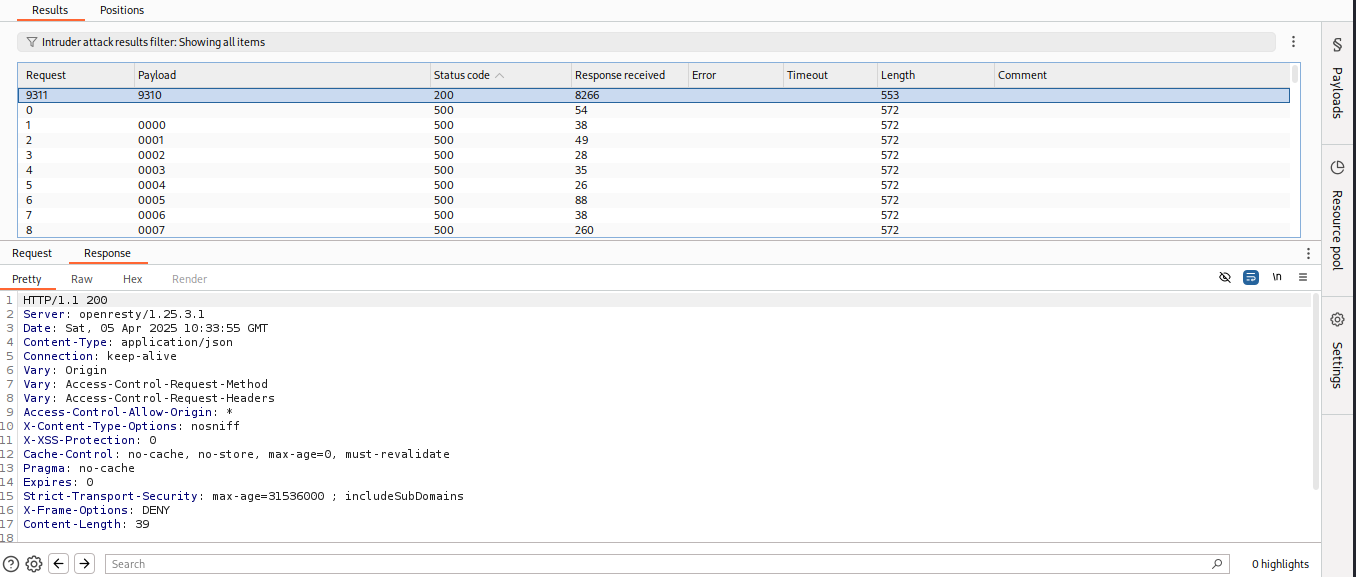
****

And intercept the request again and send to the intruder and change the version v3 to v2

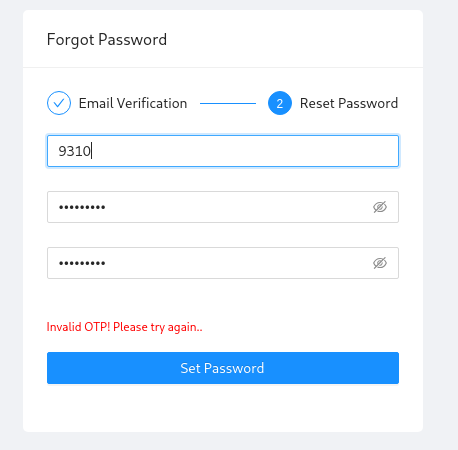
****

**Here is my attack is successful because of not rate limiting**

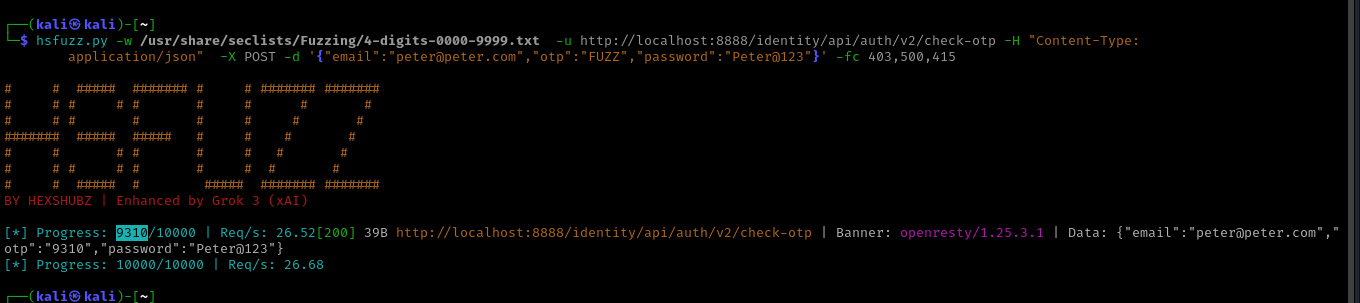
**Here allow attackers to bypass login**

****

**Go the browser and enter the otp**

****

**Here You can bruteforce the otp with tool like hsfuzz and ffuf**

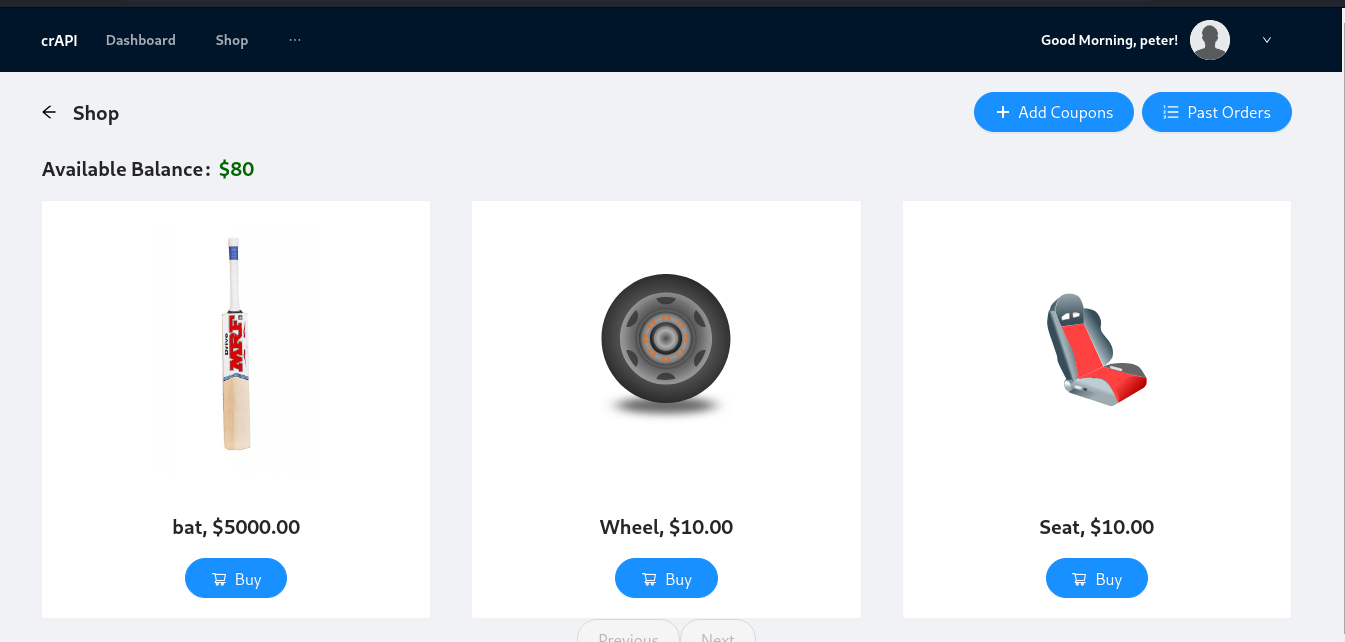
****

**API 03:2023 — Broken Object Property Level Authorization**

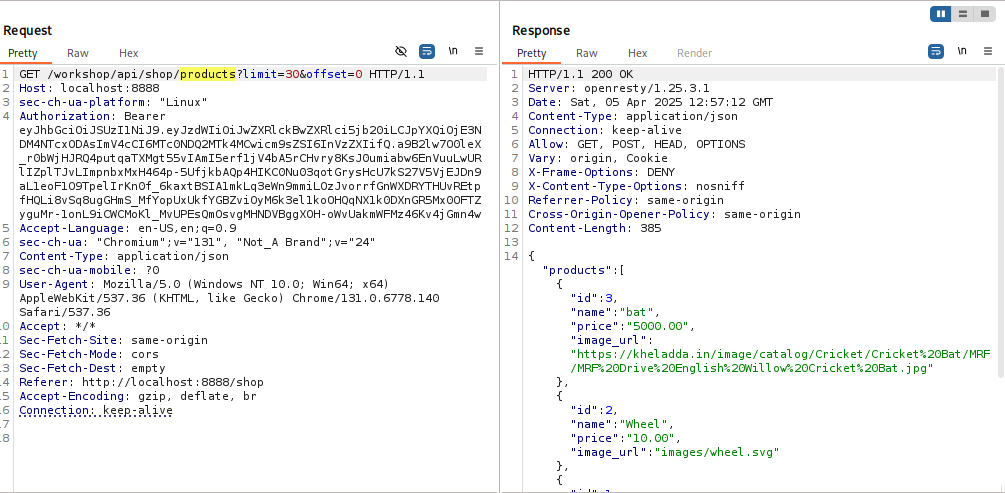
This category combines [API3:2019 Excessive Data Exposure](https://owasp.org/API-Security/editions/2019/en/0xa3-excessive-data-exposure/) and [API6:2019 - Mass Assignment](https://owasp.org/API-Security/editions/2019/en/0xa6-mass-assignment/), focusing on the root cause: the lack of or improper authorization validation at the object property level. This leads to information exposure or manipulation by unauthorized parties.

**Exploitation;**

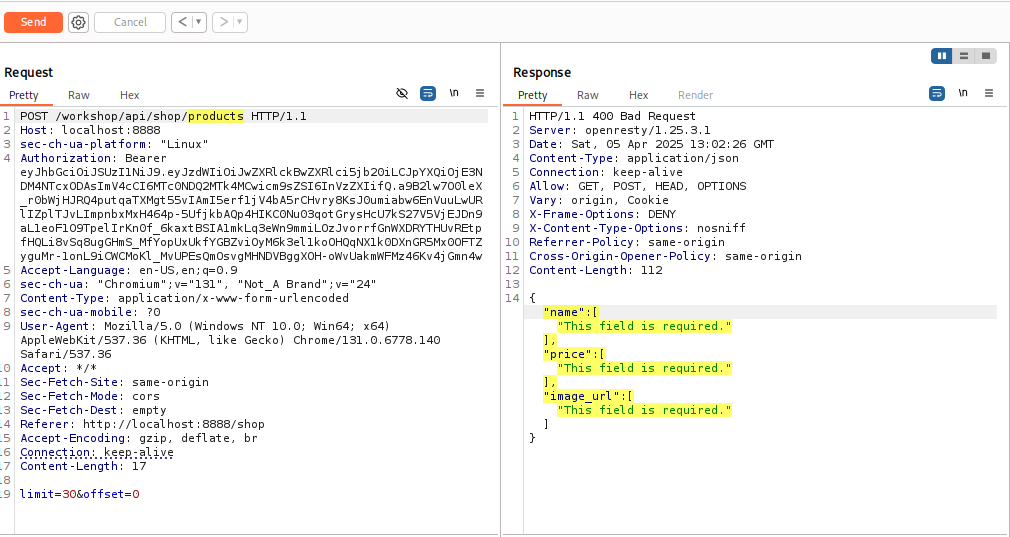
**Go the /shop endpoint here present three products**

****

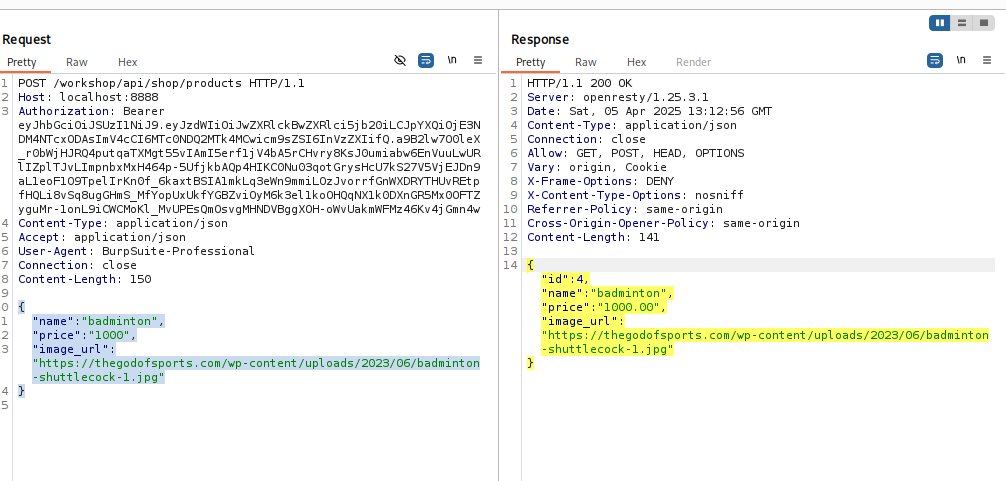
**Intercept this endpoint**

****

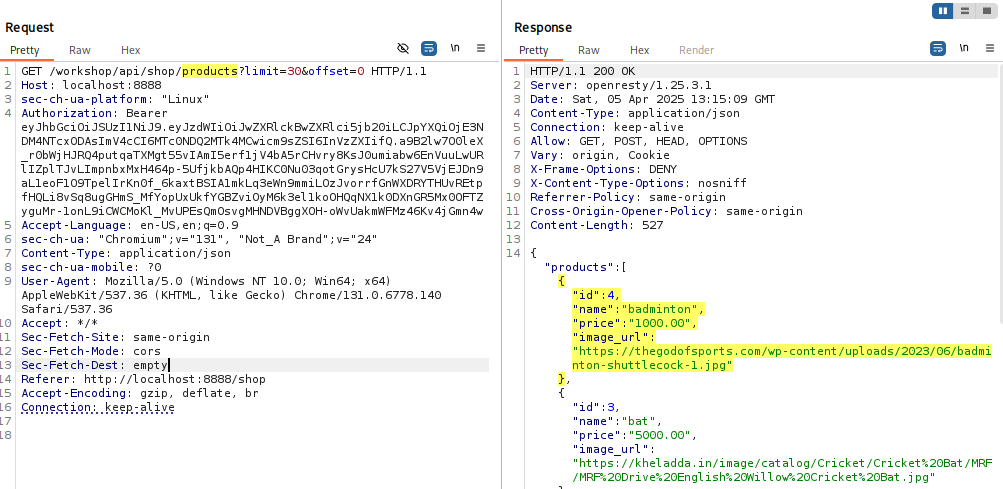
**And change this method to POST and come bad request but you can see this below this field is required**

****

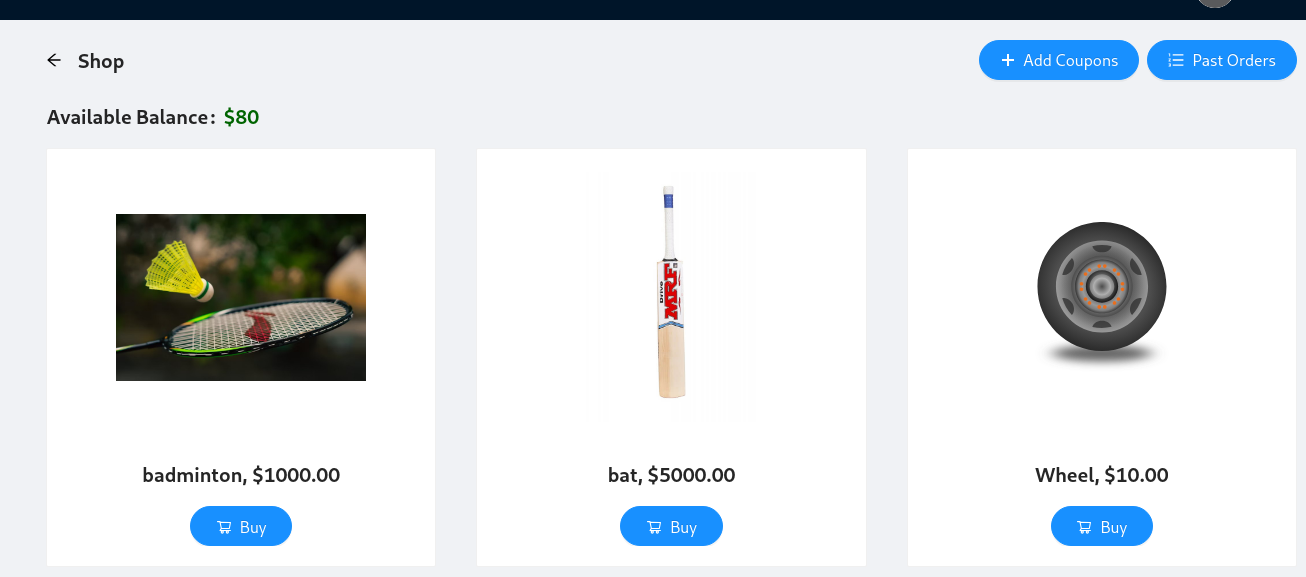
**And I fill-ups the required filled with Jason data send to the request and come 200 ok**

****

**Retrieve the data with GET method here you can see my item(badminton) is added**

****

**Here You can see this item is successfully added on web application**

****

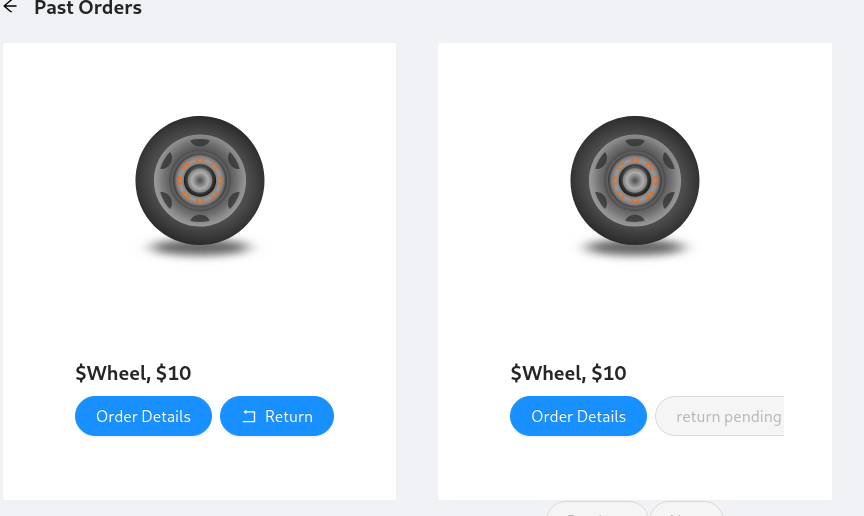
**🛑 API 04:2023 —** Unrestricted Resource Consumption

**🔍 Issue:**

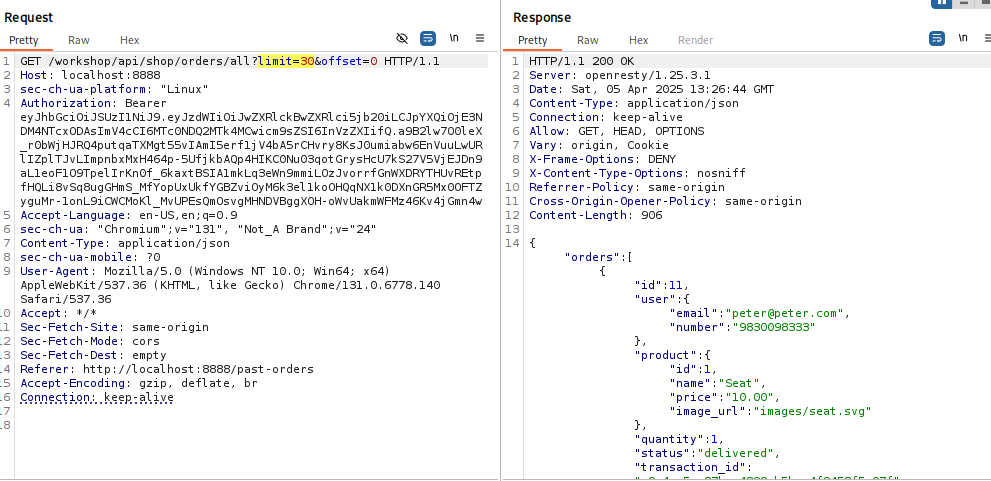
Attackers abuse API requests to overload the system.

API has no rate limits on CPU, memory, or bandwidth usage**.**

**🔨 Exploitation ; go the the order page**

****

**Intercept the request of order**

****

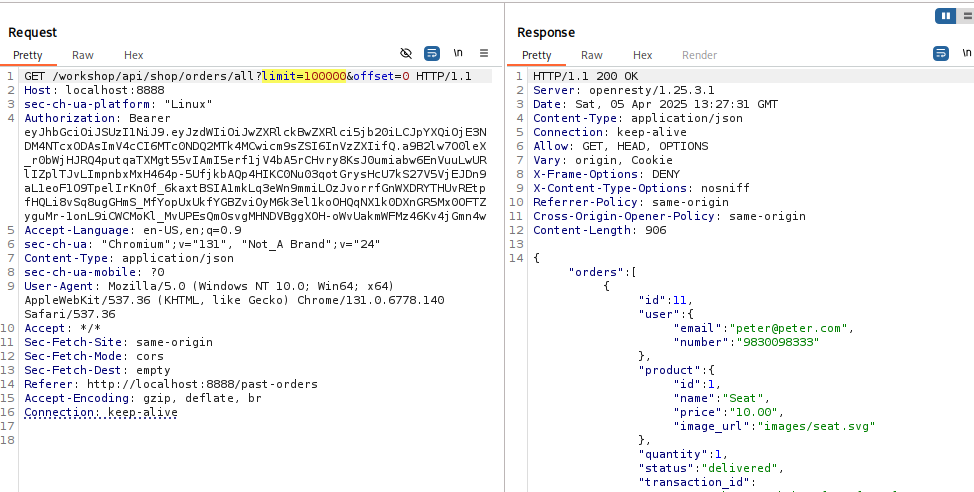
**I increase the large limit (1000000) but come same response**

**Hit this endpoint multiple times with high limits (e.g. Burp Intruder) to:**

**Cause DoS**

**Detect if there's rate limiting**

**Observe performance impact**

****

**🔒 Secure Fix (for devs):**

Server-side validation

**if limit > 100:**

return {"error": "Limit exceeds maximum allowed (100)"}, 400

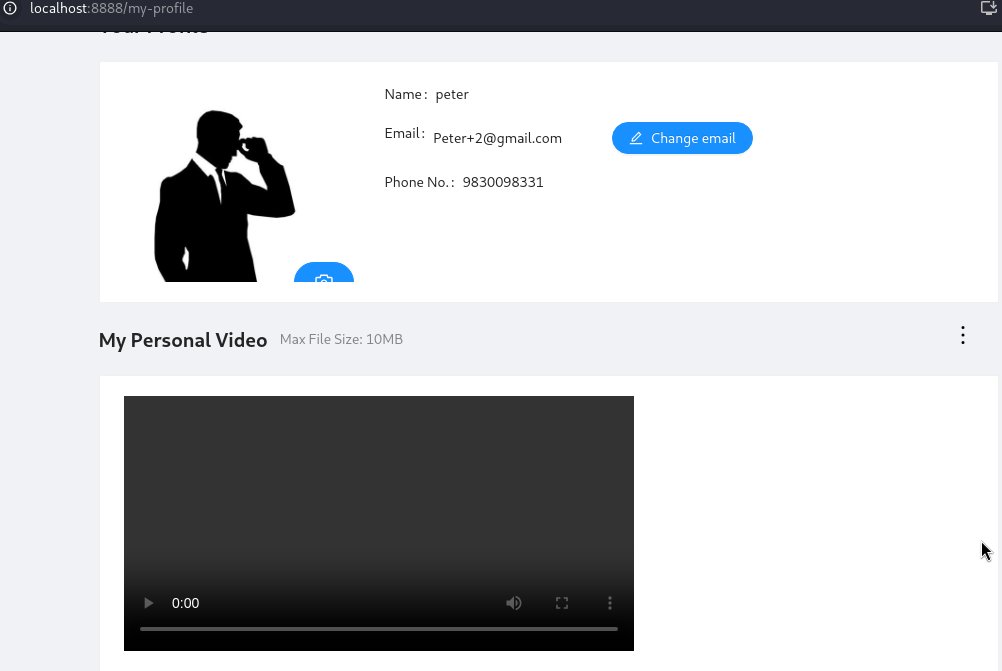
🛑 **API 05:2023** — Broken Function Level Authorization

**🔍 Issue**:

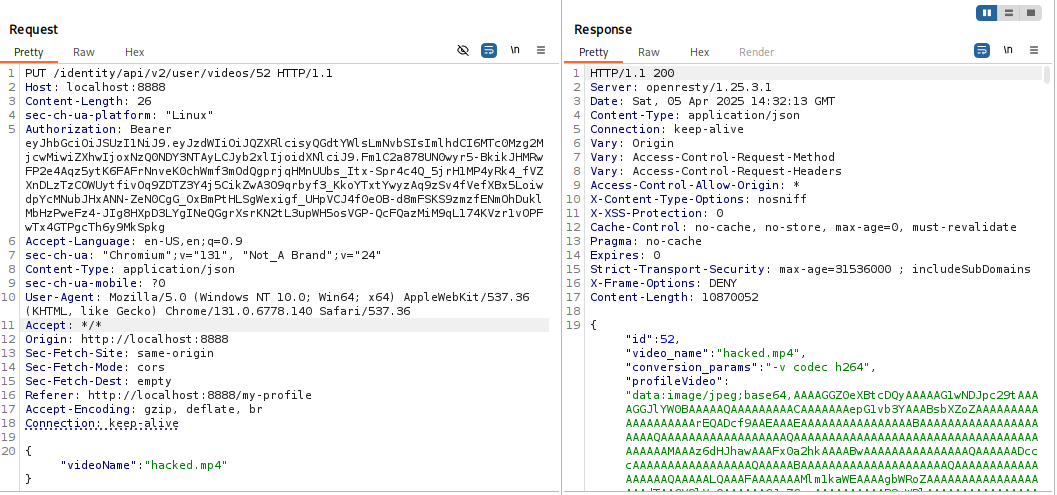
Users can access admin functions due to poor role checks.

**🔨 Exploitation** :

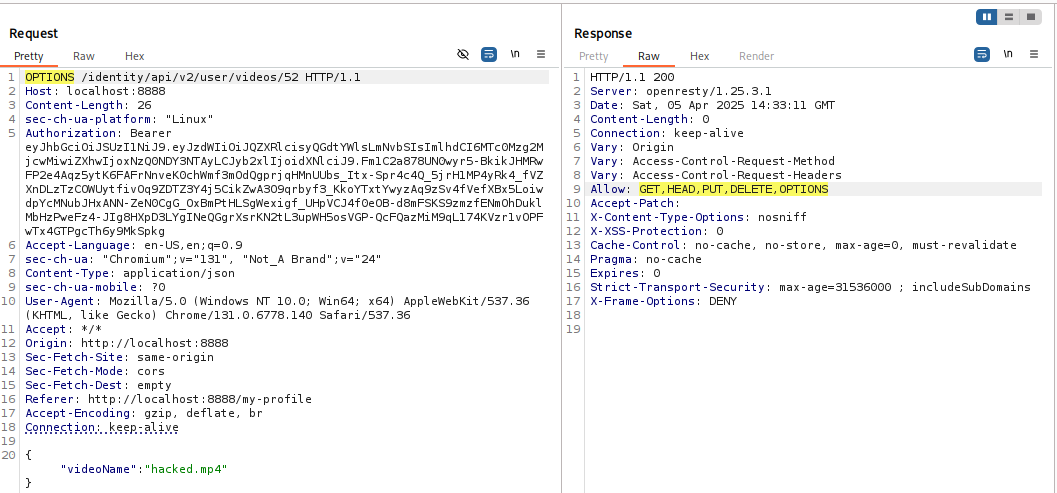
Upload the video to click on three dot and change the name of video



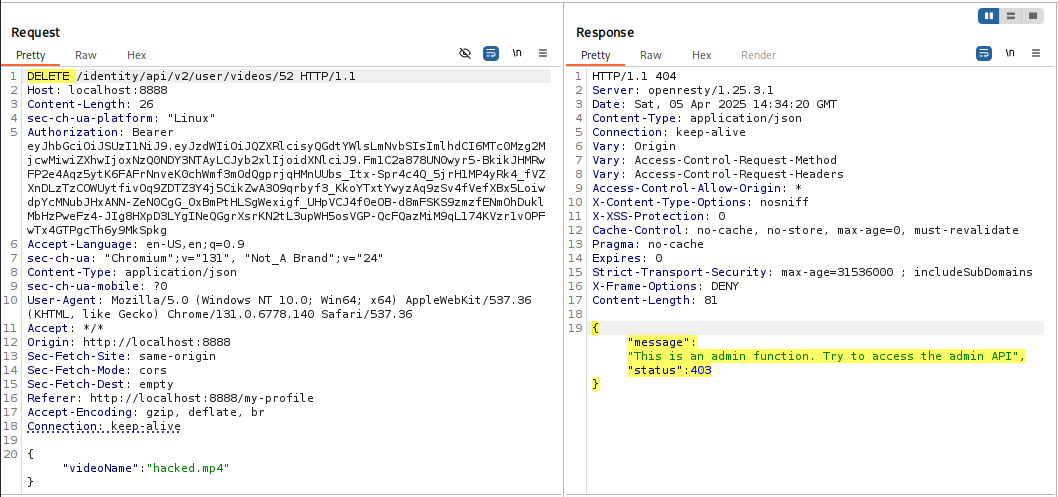
**Intercept the request**

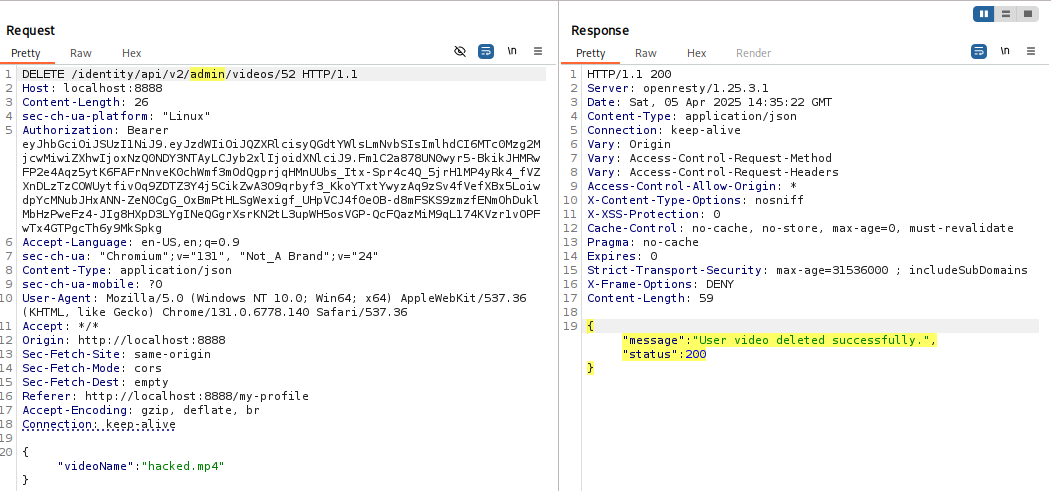


**Here I analyse the request using options methods to see which method is allow here.**



Here I choose DELETE method but come 404 and u read this message in request below



**Here I change user to admin and video deleted successfully and my attack is successful to BFLA vulnerability**

**🛑 API 06:2023 —** Unrestricted Access to Sensitive Business Flows

**🔍 Issue:**

APIs expose business-critical functions without restrictions.

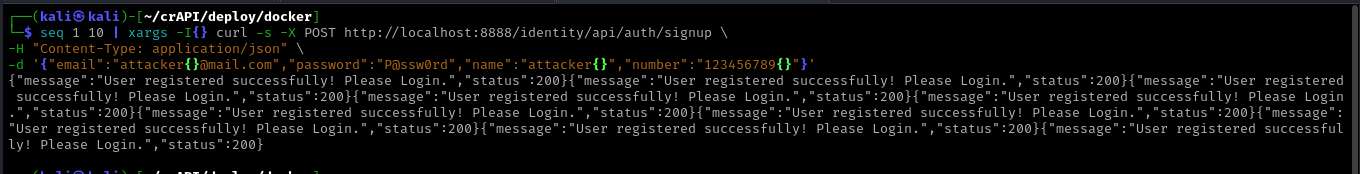
Attackers abuse account creation, financial transactions, etc**.**

**🔨 Exploitation :**

**Now you're sending all required fields**, and the API accepts it:

* email: attacker{}@mail.com
* password: P@ssw0rd
* name: attacker{}
* number: 123456789{}

The backend API is built using **Spring Boot**, and it’s validating input with annotations like @NotBlank. That’s why you got errors like:



### ✅ Why is this a vulnerability?

1. **No rate-limiting / CAPTCHA**  
   ⇒ You can automate user signups without restriction
2. **No authentication needed to hit signup endpoint**  
   ⇒ Anyone can register **multiple fake users**, maybe to:
   * Spam the system
   * Fill up the database
   * Use those accounts to perform further attacks
3. **Critical business flow exposed without control**  
   ⇒ Account creation should have protections in place

### 💥 Vulnerability: Unrestricted Access to Sensitive Business Flows

This happens when:

* Business logic flows (like registration, password reset, checkout) are exposed without security measures
* There's no check for abuse

🛑 **API 07:2023** — Server-Side Request Forgery (SSRF)

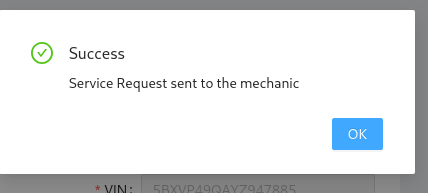
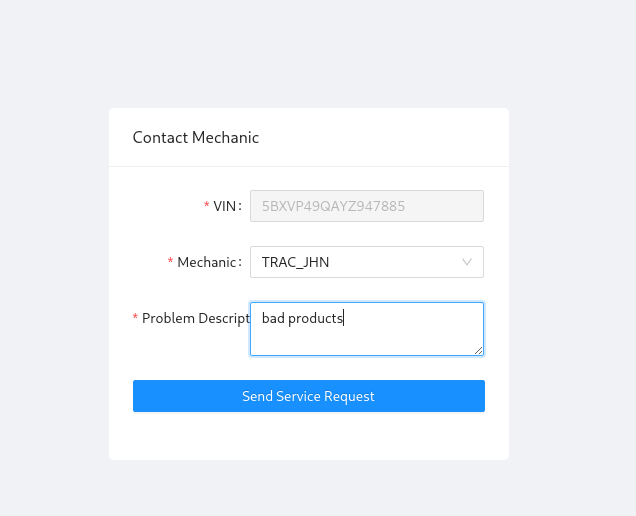
🔍 **Issue:**

APIs allow users to request external/internal URLs.

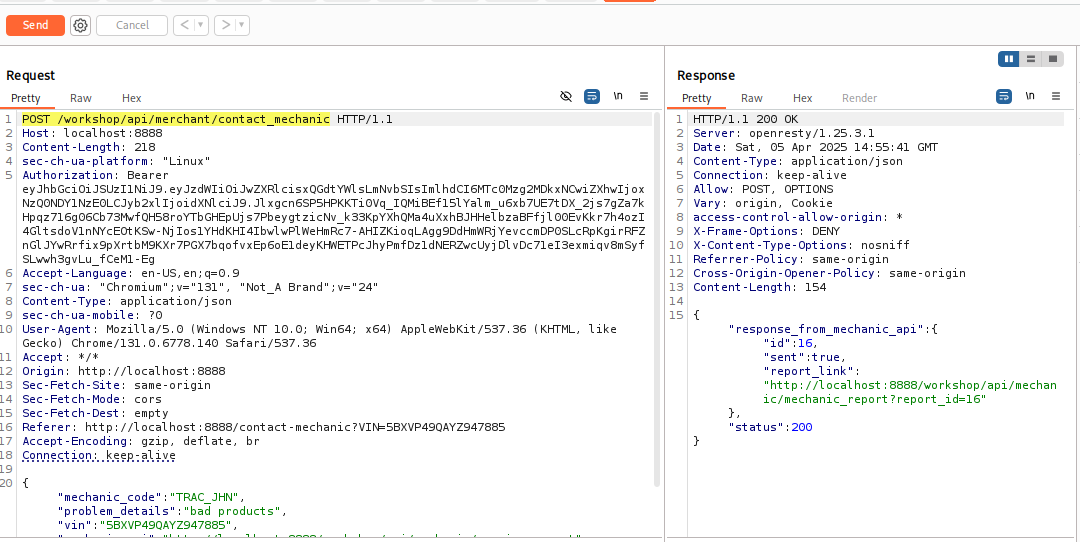
Attackers force the server to fetch unauthorized data.

🔨 **Exploitation;**

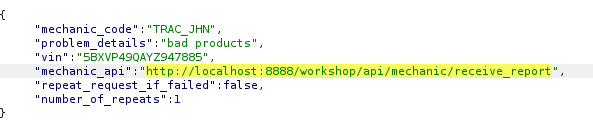
Go the contact page



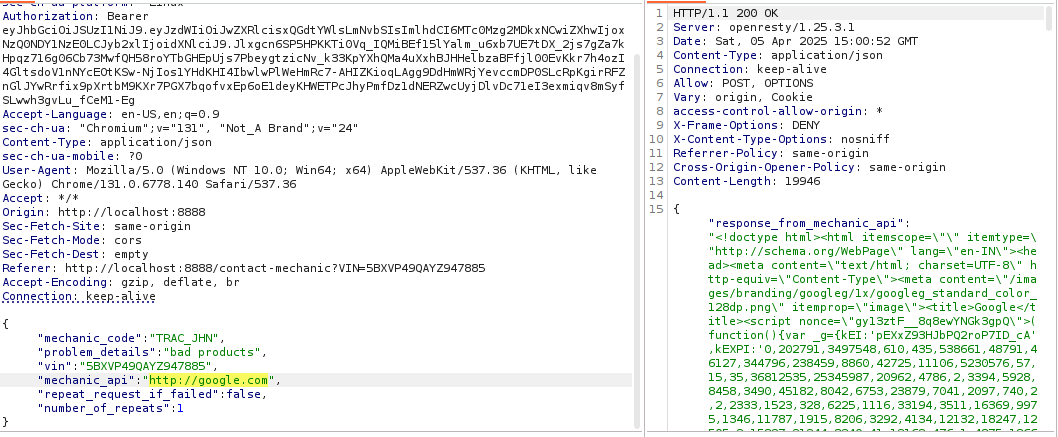
Intercept the request



Here you can check the request data send to this type

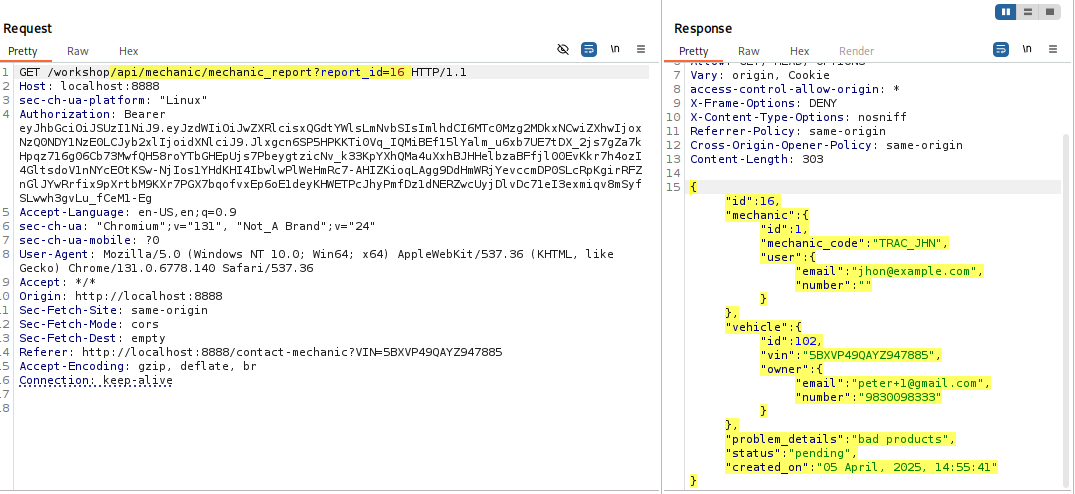


**Here I change the url highlighted in yellow colour and fetch of this url request this is vulnerable to ssrf**

****

**SQLI Check there**

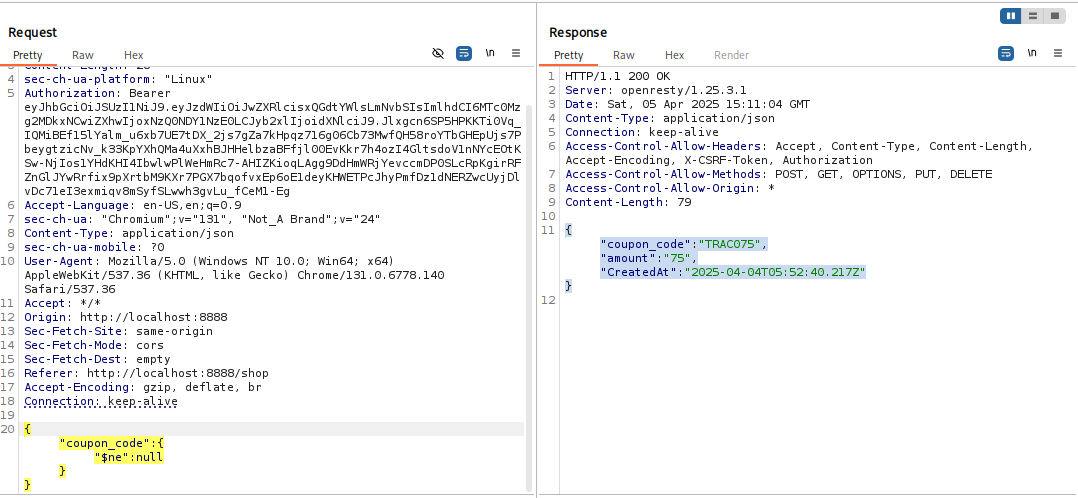
**Fetch this endpoint and intercept the burp and send to the repeater**

****

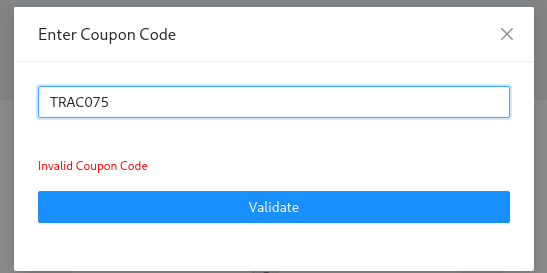
**I try here sqli and come error this is the first clue of error based sqli**

****

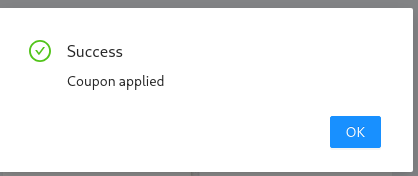
**Here I using no sql payload and this successful you can see this below**

****

**Here I validate the coupon**

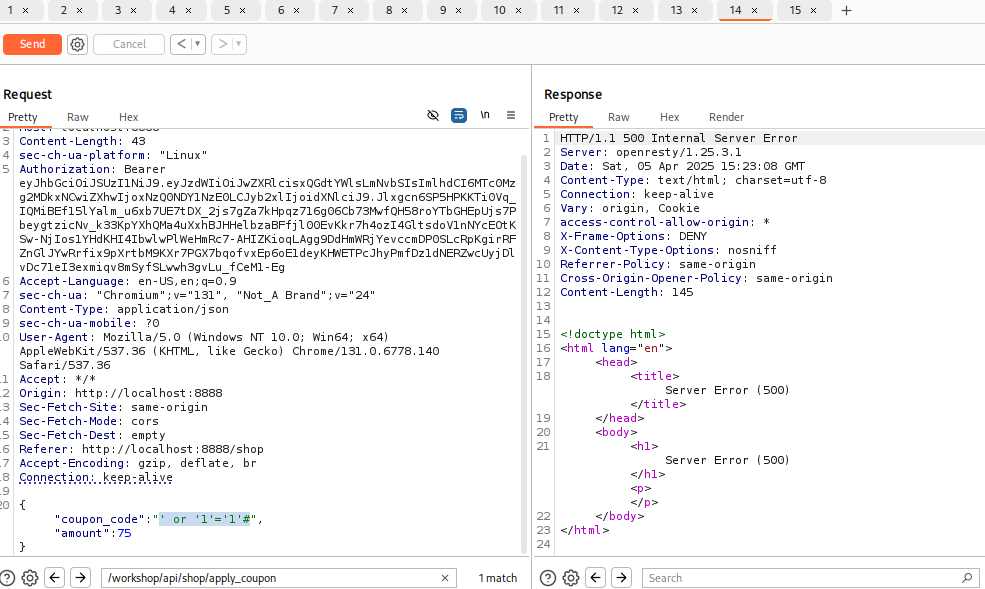
****

**Work this conform to sqli**

****

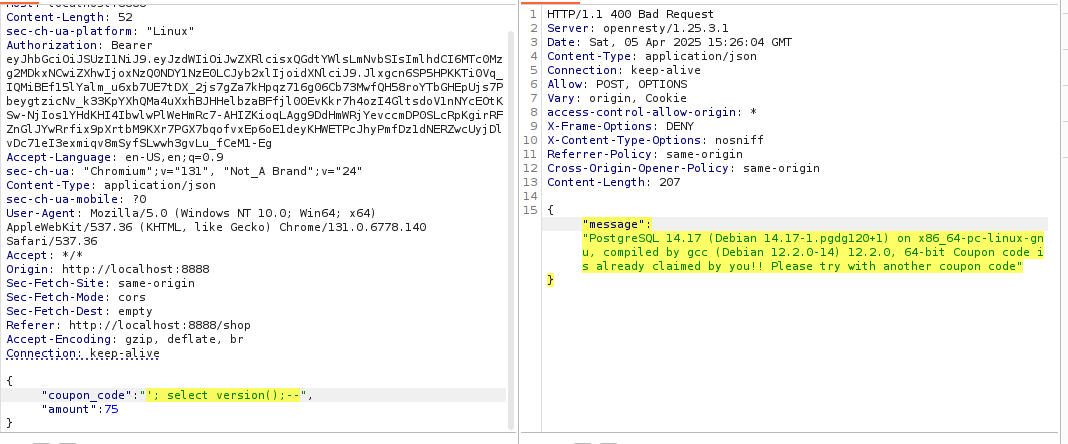
**Second example of sqli**

**Go to the /workshop/api/shop/apply\_coupon endpoint try sql payload and come error again**

****

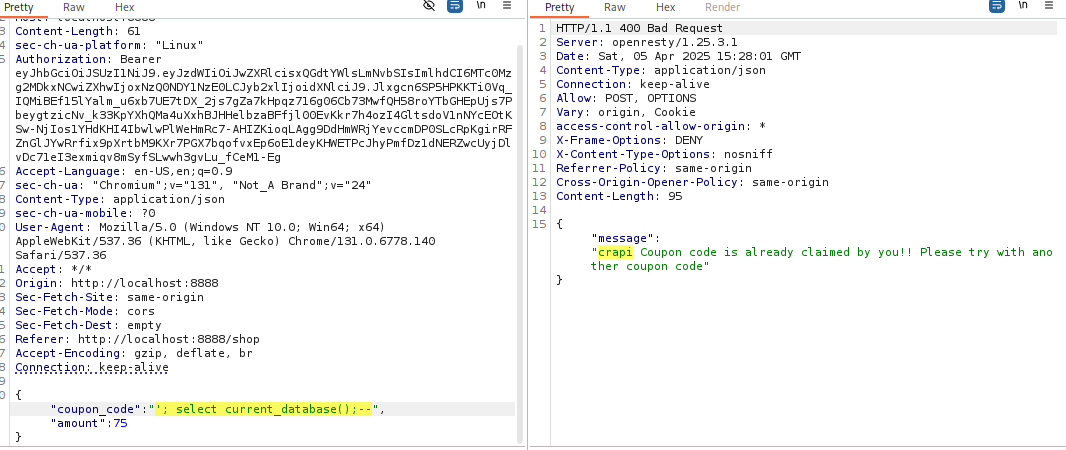
**Use this payload to see the version you can see this below**

**‘ ; select version ();--**

****

**See for data base use this payload**

**‘ ; select current\_database ();--**

****

**🛑 API 08:2023 —** Security Misconfiguration

**🔍 Issue:**

API has default credentials, open admin panels, or unnecessary headers.

Attackers exploit misconfigurations to access sensitive info**.**

**🔨 Exploitation Example:**

**# Checking for open API docs & admin panels**

curl -X GET "https://api.example.com/admin"

curl -X GET "https://api.example.com/swagger"

**✅ Fix:**

Disable API documentation in production.

Enforce least privilege access to admin panels.

**🛑 API 09:2023** — Improper Inventory Management

**🔍 Issue:**

APIs expose deprecated/hidden endpoints attackers can abuse.

**🔨 Exploitation Example:**

# Fuzzing for hidden API endpoints

ffuf -w wordlist.txt -u https://api.example.com/FUZZ

**✅ Fix:**

Regularly audit APIs & remove outdated endpoints.

Implement strict API versioning.

**🛑 API 10:2023 —** Unsafe Consumption of APIs

**🔍 Issue:**

API blindly trusts data from third-party APIs.

Attackers inject malicious payloads via external API calls**.**

**🔨 Exploitation Example:**

# An API integrates with an external payment system

curl -X POST "https://api.example.com/payment" -d '{"amount":"100","status":"paid"}'

**✅ Fix:**

Validate & sanitize all third-party API responses.

Implement signature-based verification for external data.