Based on the provided **OWASP Top.pdf** document, which describes a hands-on lab

OWASP Top 10 API Security Lab Task Sheet

Application: crAPI (Completely Ridiculous API)
Objective: Identify, exploit, and understand real-world API vulnerabilities aligned with the OWASP API Security Top 10.

Prerequisites

1. A virtual machine (e.g., Ubuntu) with Docker and Docker Compose installed.
2. Burp Suite Community Edition installed on your attacker machine.
3. Firefox browser with FoxyProxy and Burp CA certificate configured.
4. Access to crAPI at `http://<IP>:8888` and MailHog at `http://<IP>:8025`.

Tasks

1. Broken Object Level Authorization (BOLA / IDOR)

- **Challenge 1:** Retrieve another user's vehicle details by manipulating the `vehicle_id` in the API request.
- **Challenge 2:** Access mechanic reports belonging to other users by brute-forcing `report_id`.

2. Broken User Authentication

- **Challenge 3:** Reset another user's password by:
  - Harvesting their email from the Community API.
  - Brute-forcing the 4-digit OTP using the `/v2/check-otp` endpoint.
  - Use the provided Python script (`brute_otp_crapi.py`) to automate this.

3. Excessive Data Exposure

- **Challenge 4:** Identify sensitive user data leaked via `/community/api/v2/community/posts/recent`.
- **Challenge 5:** Upload a video and inspect the response from `/identity/api/v2/user/videos` to find internal video properties exposed.

4. Lack of Rate Limiting

- **Challenge 6:** Trigger a Layer 7 DoS by abusing the "Contact Mechanic" feature:

- Modify `repeat_request_if_failed=true` and `number_of_repeats=10000`.

5. Broken Function Level Authorization (BFLA)

- **Challenge 7:** Delete another user's video by:
  - Changing your `PUT` request to `DELETE`.
  - Switching the endpoint from `/user/videos/` to `/admin/videos/`.
  - Guessing other video IDs (e.g., ID = 20).

6. Mass Assignment

- **Challenge 8:** Get an item for free by changing order status to `returned` via a `PUT` request.
- **Challenge 9:** Inflate your account balance by >$1,000 using mass assignment (`quantity=100`, `status=returned`).
- **Challenge 10:** Modify internal video properties (e.g., add `conversion_params: "-v codec h264"`).

7. Server-Side Request Forgery (SSRF)

- **Challenge 11:** Abuse the mechanic contact form to make crAPI fetch `https://www.google.com` by injecting it into the `mechanic_api` parameter.

8. NoSQL Injection

- **Challenge 12:** Bypass coupon validation using NoSQL injection payload:

```
{"coupon_code": {"$ne": 1}}
```

  - Capture the valid coupon code returned (e.g., `TRAC075`).

9. SQL Injection

- **Challenge 13:** Redeem an already-used coupon by exploiting SQLi:
  - Use payload: `1' OR '1'='1` or `1'; SELECT version()--` on the `/apply_coupon` endpoint.

10. Unauthenticated Access

- **Challenge 14:** Access order details without an `Authorization` header—prove the endpoint lacks proper auth checks.

11. JWT Vulnerabilities

- **Challenge 15:** Forge a valid JWT token by:
  - Changing the algorithm from `RS256` to `none`.
  - Removing the signature.
  - Using the modified token to access `/user/dashboard`.

Deliverables

For each challenge, submit:

1. **Request/Response** (from Burp Repeater or Proxy).
2. **Explanation** of the vulnerability and how it was exploited.
3. **Mitigation Advice** (1–2 sentences per issue).

> **Bonus:** Propose secure code fixes or architecture changes for 3 selected challenges.

Suggested Time

- **Beginner:** 6–8 hours
- **Intermediate:** 3–5 hours

Let me know if you'd like this as a printable PDF or with answer keys!