

Lab link : <https://portswigger.net/web-security/api-testing/lab-exploiting-unused-api-endpoint>

---

1. Enter the credentials

## Login

Username

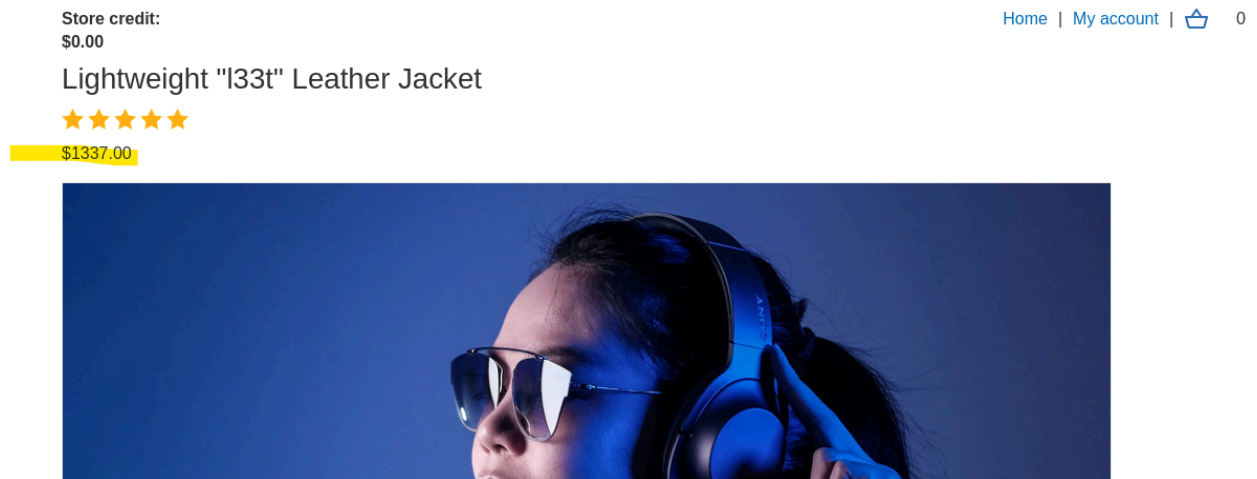
wiener

Password

•••••

Log in

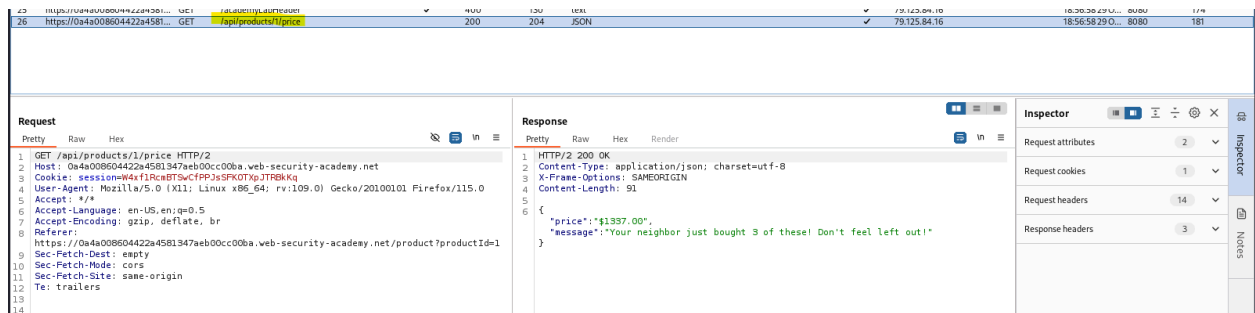
2. Go to the leather jacket and see the price, and then add it to the cart. While doing this, make sure that Burp Suite is on.



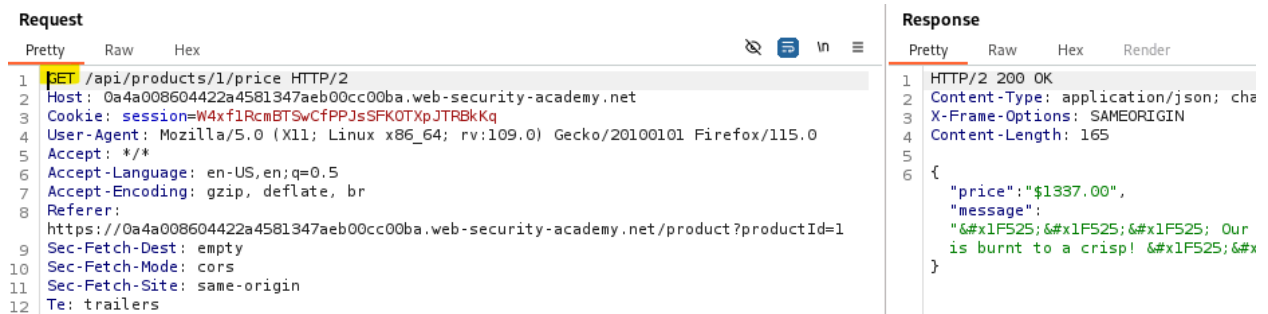
3. Go to history in Burp Suite and locate the api.

The api would look like this :

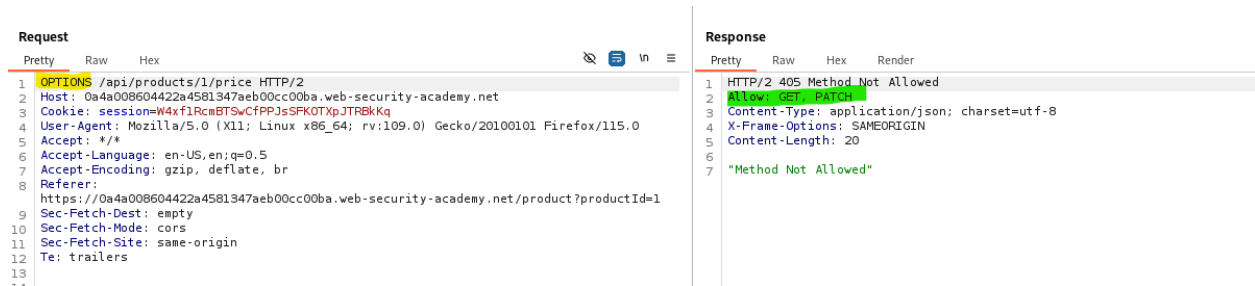
GET /api/products/price HTTP/2



#### 4. Next, send this to the repeater and change the value of GET to OPTIONS.



Then check the results using OPTIONS.



As indicated by the green highlighter above, the allowed methods are GET and PATCH. Therefore, I need to check the response when using the PATCH method.

#### 5. Change the OPTIONS to PATCH. To see what response it will give.



Once it was changed from OPTIONS to PATCH. An error was shown that

“Only application/json’ Content-type is supported”.

This means that the content header must be added.

6. When adding the header for the content-type, make sure to include the request body as JSON. Setting the content-type to application/json indicates that the request body must be formatted as valid JSON.

Add:

```
{  
  "price": "0.00",  
  "message": "testing see if it works"  
}
```

The screenshot shows a REST client interface with a target URL of `https://0a4a008604422a4581347aeb00cc`. The Request tab is active, displaying a PATCH request to `/api/products/1/price` with HTTP/2. The request headers include `Host: 0a4a008604422a4581347aeb00cc00ba.web-security-academy.net`, `Cookie: session=W4xf1RcmBT5wCfPPJsSFKOTXpJTRBKkQ`, `User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0`, `Accept: */*`, `Accept-Language: en-US,en;q=0.5`, `Accept-Encoding: gzip, deflate, br`, `Referer: https://0a4a008604422a4581347aeb00cc00ba.web-security-academy.net/product?productId=1`, `Sec-Fetch-Dest: empty`, `Sec-Fetch-Mode: cors`, `Sec-Fetch-Site: same-origin`, `Te: trailers`, and `Content-Type: application/json` (highlighted in yellow). The request body is a JSON object: `{ "price": 0, "message": "Testing see if this works." }`. The Response tab shows a 200 OK response with headers `Content-Type: application/json; charset=utf-8`, `X-Frame-Options: SAMEORIGIN`, and `Content-Length: 17`. The response body is `{ "price": "$0.00" }`, with `"price": "$0.00"` underlined in red.

As indicated by the yellow highlighter above, since the price parameter must be a non-negative number, change the price from \$0.00 to just an integer 0.

7. After changing the number, the response would be updated,

The screenshot shows the same REST client interface with the target URL `https://0a4a008604422a4581347aeb00cc`. The Request tab displays the same PATCH request, but the request body is now `{ "price": 0, "message": "Testing see if this works." }`. The Response tab shows the same 200 OK response with headers `Content-Type: application/json; charset=utf-8`, `X-Frame-Options: SAMEORIGIN`, and `Content-Length: 17`. The response body is `{ "price": "$0.00" }`, with `"price": "$0.00"` underlined in red.

- Now go back to the home page and see the price of the leather jacket and add it to the cart.



Lightweight "I33t" Leather Jacket

★★★★★ \$0.00

[View details](#)

This shows that the price was updated.

Store credit:  
\$0.00

## Cart

Name	Price	Quantity	
Lightweight "l33t" Leather Jacket	\$0.00	<div>- 1 +</div>	<div>Remove</div>

Coupon:

Apply

Total: \$0.00

Place order

When added to the cart the price is \$0.00.

Therefore ....



Finding and exploiting an unused API endpoint

[Back to lab description >>](#)

LAB Solved

Congratulations, you solved the lab!

Share your skills! [Continue learning >>](#)

Store credit:  
\$0.00

Your order is on its way!

Name	Price	Quantity
Lightweight "l33t" Leather Jacket	\$0.00	1

Total: \$0.00

[Home](#) | [My account](#) | 0

A free jacket its on its way and the lab is solved! 😊



---

## 💡 What have I learned?

In API there are 8 HTTP methods.

- GET
- PUT
- POST
- DELETE
- PATCH
- HEAD
- OPTIONS
- TRACE

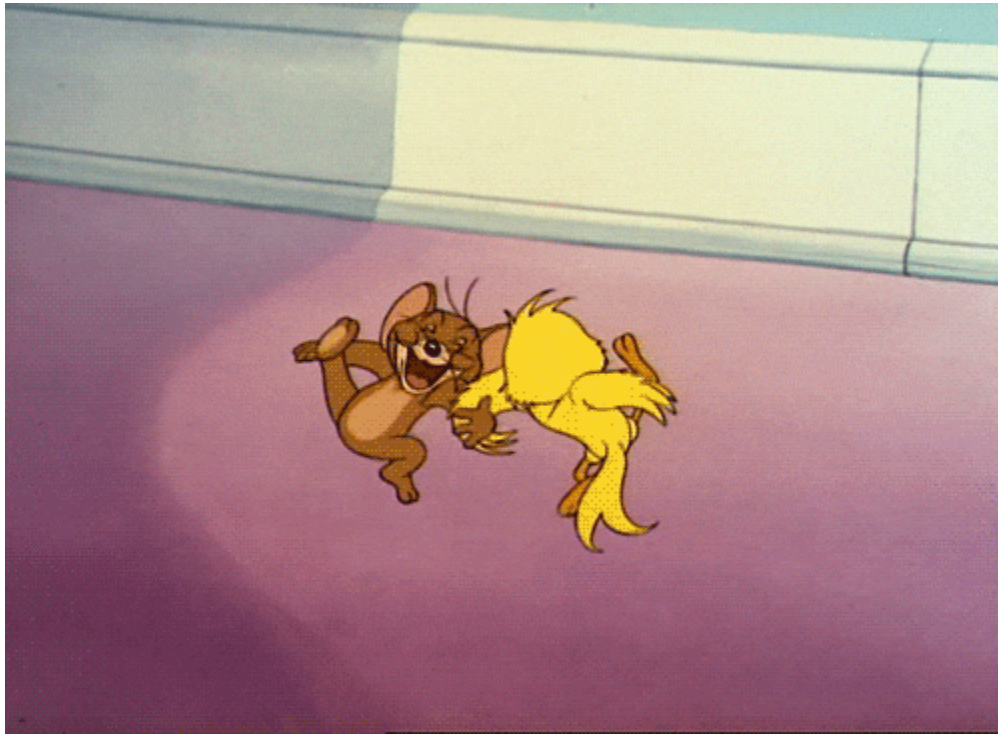
For this lab, I learned and understood how **PATCH** works. Unlike **PUT**, which updates the entire resource, **PATCH** only updates the specific fields of the resource. In this case, I used **PATCH** to update the price of the leather jacket.

Additionally, when using **OPTIONS**, it can reveal the methods that are supported and required by the endpoint.

But I have also learned two new HTTP methods: **TRACE** and **HEAD**.

- **TRACE** is used for debugging and testing communication purposes between the client and the server.
- **HEAD** only retrieves the headers of the response, not the entire content.

That's all for today. 🙌



Signing off, noob3InT 📝