

API TESTING

Api testing is a process of confirming that everything is working as expected. Developers can run API test manually or automate it with API testing tool. Recently, API testing is being run earlier in API lifecycle. This approach is called shifting left and it helps to fix and catch issues earlier in the development cycle.

There are many ways to test API and they all serve a different purpose.

1.1DIFFERENT WAYS OF API TESTING

1.1.1 CONTRACT TESTING

API contract is readable and machine-readable representation of an API's intended functionality. It forms the basis of SLA's between producers and consumers. It helps ensure that new releases don't violate the contract by checking the content and format of request and response.

1.1.2 UNIT TESTING

API Unit testing is the process of confirming that a single endpoint returns the correct response to a given request. It validates whether an endpoint handled optional parameters correctly or it returned an appropriate error message when an invalid request was sent.

1.1.3 END-TO- END TESTING

They are used to validate key user journeys that may involve multiple API's and endpoints. It involves chaining requests together and confirming that each one is working properly, which helps teams surface issues in complex environments before users do.

1.1.4 LOAD TESTING

API load testing enables developers to confirm whether their API's is able to operate reliably during times of peak traffic. It typically involves using a testing tool to simulate large request volumes and measure the resulting response time and error rates. This type of testing is usually performed in anticipation of significant load increase, such as right before product launch or yearly sale.

1.2 API RACON

Before API testing it is necessary to collect as much information as possible. First step is to identify API endpoints. For example, if we took the example like the following:

```
GET /api/books HTTP/1.1  
Host: example.com
```

The API endpoint for this request is /api/books. This results in interaction with API to retrieve list of books from the library. Another API endpoint might be for example to retrieve the list of mystery books /api/books/mystery., which would retrieve list of mystery books.

Once the endpoints are identified next step is to figure out how to interact with them. You should find information about the following:

- The input data the API processes, including both compulsory and optional parameters.
- The type of request the API accepts, including supported HTTP methods and media formats.
- Rate limits and authentication mechanism.

APIs are usually documented so that developers know how to use them and integrate with them. Documentation can be in both human readable and machine-readable format. It may include detailed explanations, examples and usage scenarios.

Even if the API application isn't openly available, you can access it by browsing applications that use the API.

Burp Scanner can be used for this, to crawl the API. Applications can also be browsed manually using the Burp browser.

Look for endpoints that may refer to the API documentation:

- /api
- /swagger/index.html
- /openapi.json

For example, if you identify the resource endpoint /api/swagger/v1/users/123, then you should investigate the following paths:

- /api/swagger/v1
- /api/swagger
- /api

LAB 1

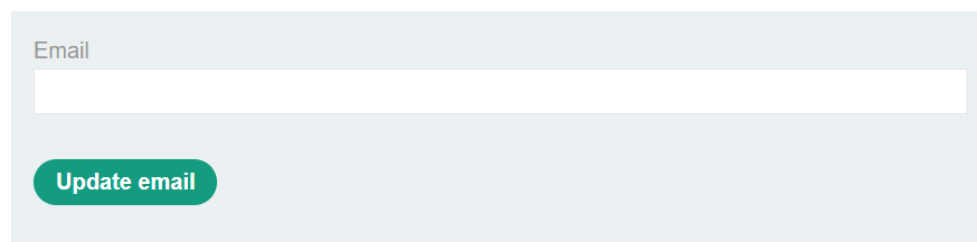
Finding and exposed API documentation and delete Carlos.

First we need to log in with credentials wiener:peter like in the given picture below.

My Account

Your username is: wiener

Your email is: wiener@normal-user.net

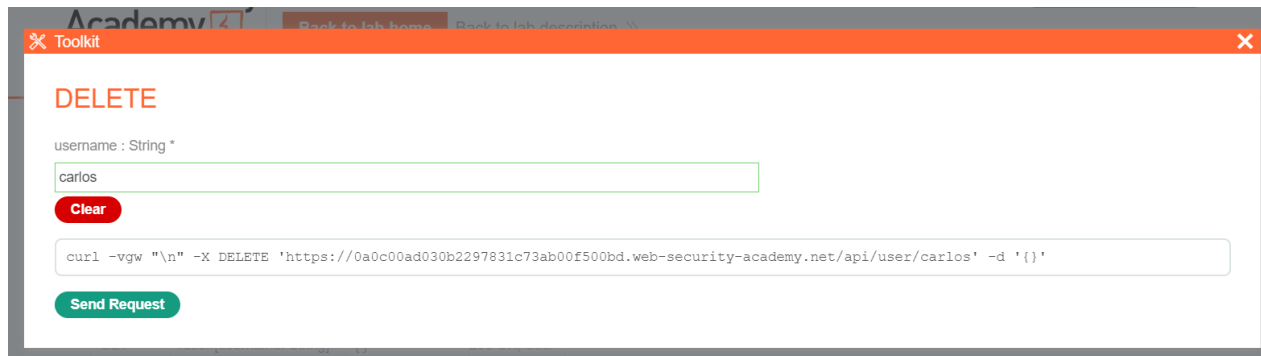
A screenshot of a web application's 'My Account' page. The page has a light blue header with the title 'My Account' in a large, bold, blue font. Below the header, there are two lines of text: 'Your username is: wiener' and 'Your email is: wiener@normal-user.net'. Below this text is a light blue rectangular box containing an email update form. The form has a label 'Email' in a small, gray font above a white input field. Below the input field is a green button with the text 'Update email' in white.

Picture 1. Logging to the account

So for deleting the user Carlos, we changed the url:

<https://0a0c00ad030b2297831c73ab00f500bd.web-security-academy.net/api/>

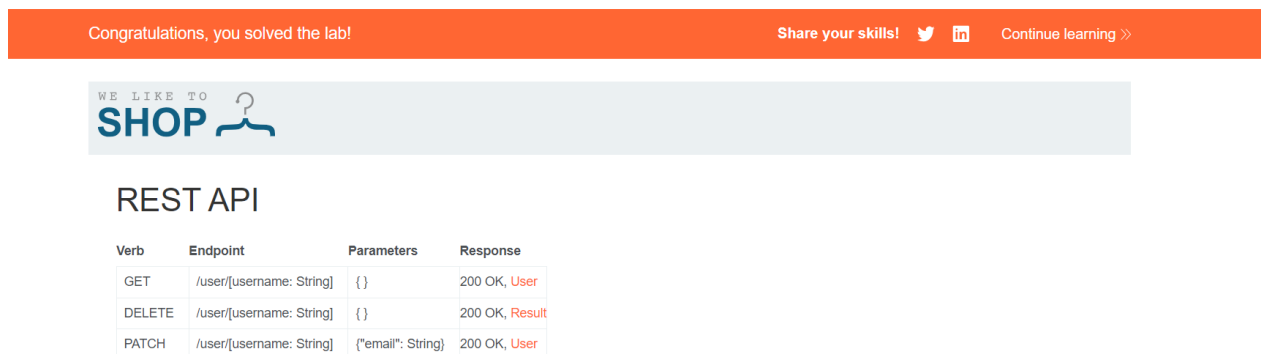
First we went to the rest api to get delete column and typed in Carlos, like in the given picture:



The screenshot shows the 'ToolKit' interface for the Web Security Academy. It features a 'DELETE' section with a text input field containing 'carlos'. Below the input is a 'Clear' button. A text area contains the curl command: `curl -v -X DELETE 'https://0a0c00ad030b2297831c73ab00f500bd.web-security-academy.net/api/user/carlos' -d '{}'`. At the bottom is a 'Send Request' button.

Picture 2. Deleting user Carlos

After deleting Carlos, lab is successfully solved (picture 3).



The screenshot shows a congratulatory message: "Congratulations, you solved the lab!". To the right are links for "Share your skills!" (with Twitter and LinkedIn icons) and "Continue learning >>". Below this is a section titled "WE LIKE TO SHOP" with a shopping bag icon. Underneath is a "REST API" section containing a table with the following data:

Verb	Endpoint	Parameters	Response
GET	/user/{username: String}	{ }	200 OK, User
DELETE	/user/{username: String}	{ }	200 OK, Result
PATCH	/user/{username: String}	{"email": String}	200 OK, User

Picture 3. Lab completion

1.3 IDENTIFYING API ENDPOINTS

A lot of information can be gathered by browsing applications that use the API. This is often worth doing even if you have access to API documentation, as sometimes documentation may be inaccurate or out of date.

Burp Scanner can be used to crawl the application, then manually investigate interesting attack surface using Burp's browser.

While browsing the application, look for patterns that suggest API endpoints in the URL structure, such as /api. Also look out for the JavaScript files. These can contain references to API endpoints that you haven't triggered directly via the web browser.