API Testing (overview & recon)

Overview

APIs (Application Programming Interfaces) enable software systems and applications to communicate and share data. API testing is important as vulnerabilities in APIs may undermine core aspects of a website's confidentiality, integrity, and availability.

All dynamic websites are composed of APIs, so classic web vulnerabilities like SQL injection could be classed as API testing.

recon

To start API testing, you first need to find out as much information about the API as possible, to discover its attack surface.
To begin, you should identify API endpoints. These are locations where an API receives requests about a specific resource on its server. For example, consider the following GET request:

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

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

Once you have identified the endpoints, you need to determine how to interact with them. This enables you to construct valid HTTP requests to test the API. For example, you should find out information about the following:

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

Example:

Let's test the /books endpoint of our hypothetical bookstore API:

GET /books:

- Send a GET request to retrieve all books.
- Verify that the response contains an array of books with expected attributes like title, author, and ISBN.
- Check if the response status code is 200 (OK).

POST /books:

- Send a POST request to create a new book with JSON payload containing book details.
- Ensure that the response contains the newly created book details with a unique ID.
- Verify that the response status code is 201 (Created).

GET /books/{id}:

- Send a GET request to retrieve a specific book by ID.
- Verify that the response contains the book details matching the provided ID.
- Check if the response status code is 200 (OK) for an existing book or 404 (Not Found) for a non-existing book.

PUT /books/{id}:

- Send a PUT request to update an existing book by ID with modified details.
- Verify that the response contains the updated book details.
- Check if the response status code is 200 (OK) for successful update or appropriate error status for invalid requests.

DELETE /books/{id}:

- Send a DELETE request to remove a book by ID.
- Ensure that the book is deleted successfully, and subsequent GET requests for the same ID return a 404 (Not Found) status code