Homeork 3
Software Security

Zehra KOLAT
206001007

May 2025

1 Introduction

The purpose of this report is to assess the vulnerabilities of an API by performing simple security tests. These tests focus on various aspects such as authentication, input validation, error handling, and SQL injection. The purpose of these tests is to detect security vulnerabilities and reveal potential weak points, and to recommend improvements to increase the resilience of the API. API security is critical in modern web applications because APIs enable data communication between user data and systems. Therefore, identifying vulnerabilities in an API is crucial to preventing potential attacks.

2 Methods Used

Testing Methods
GET Request Test: This test is conducted to check whether the API is functioning properly by sending a simple GET request. The goal is to confirm if the API returns the expected response for valid requests.
API Key Usage: The accessibility of the API is tested by using an API key in the request header. This method checks whether the API correctly allows access for valid keys and denies it for invalid ones.
Authentication Test: This test ensures that the API correctly handles authentication errors. It is checked by sending requests with no API key or with an incorrect API key to see if the API properly returns an authentication error.
Input Validation Test: This test is focused on the API's ability to handle invalid inputs. By sending incorrect or malformed input, the API's response is checked to see if it returns appropriate error messages, ensuring proper input validation.
Error Handling Test: This test checks how the API handles errors when an incorrect URL or endpoint is requested. The response is observed to determine whether the API returns the correct error message, such as a 404 Not Found error.

SQL Injection Test: In this test, the API's vulnerability to SQL injection attacks is evaluated. By sending input that could potentially exploit SQL injection flaws, the API's response is monitored to see if it can effectively prevent such attacks.

3 Findings

Step 1: GET Request Test:
`Method:` In the first step, a GET request was sent via Postman. This request was made to determine if the API was working properly.

`Result:` It was found that the requested data was returned correctly and the API was working successfully. The API successfully returned a 200 OK response, indicating that the endpoint is functioning correctly.

`Security Area Tested:` The API was working properly, error messages were returned correctly.

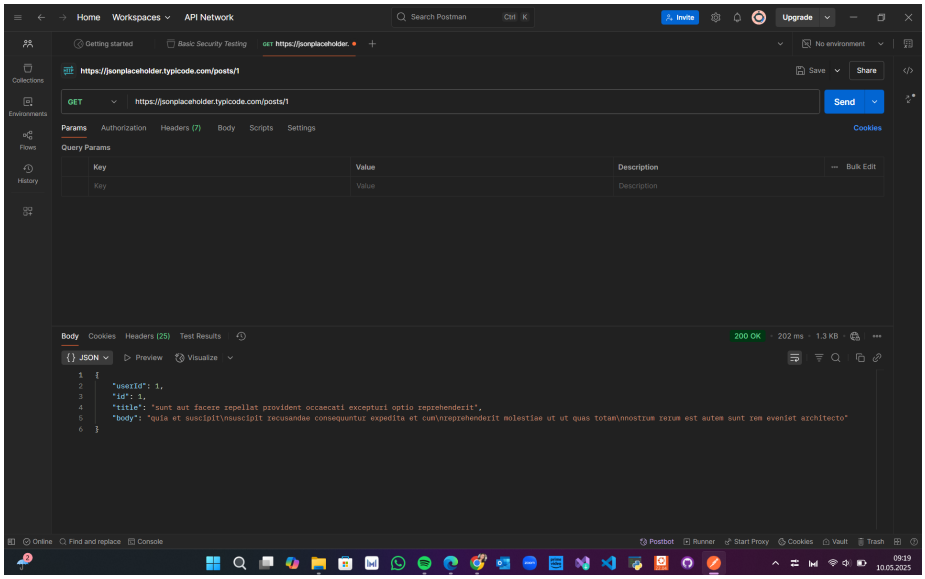

Figure 1: GET Request Test

Step 2: Obtaining an API Key from Reqres:

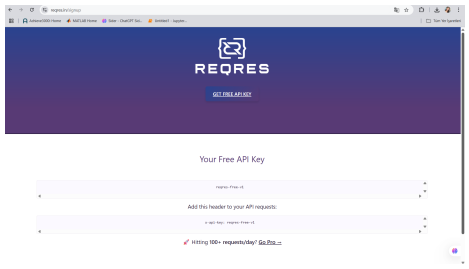

Figure 2: Locate and copy the API key

Step 3: API Key Usage:

`Method:` Tested authorized access using an API key for requests to the API.

`Result:` The API returned a 200 OK response with a valid API key. Requests with invalid keys received an error message. When a valid API key was added, the API returned a 200 OK response, confirming that the API key was properly validated.

`Security Area Tested:` Authorization requirements and access control of the API.

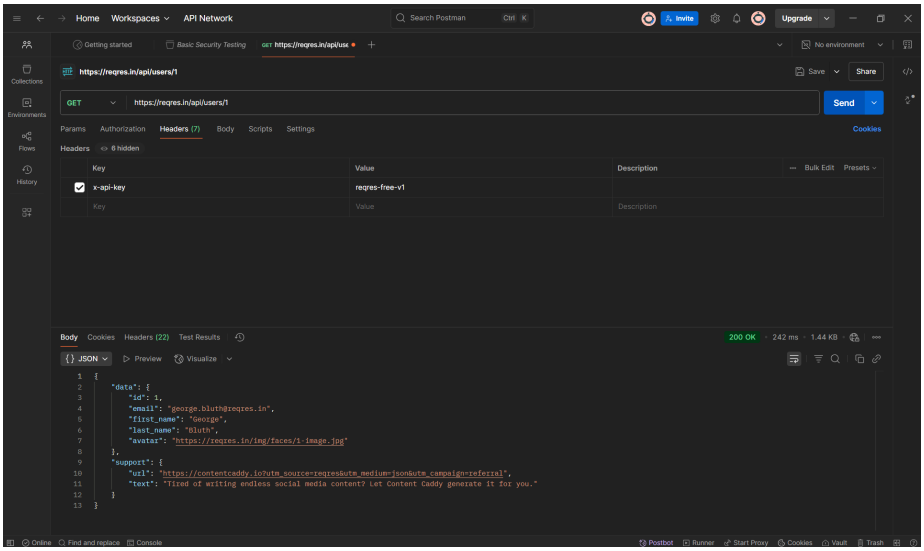

Figure 3: Using the API Key in Postman

Step 4: Authentication Check:

`Method:` Checked whether requests made with a valid API key were successfully completed. Also tested whether requests made without the key were rejected.

`Result:` It was observed that the authentication requirements were working properly and requests made without the API key were rejected. `Security Area Tested:` Authentication and authorization.

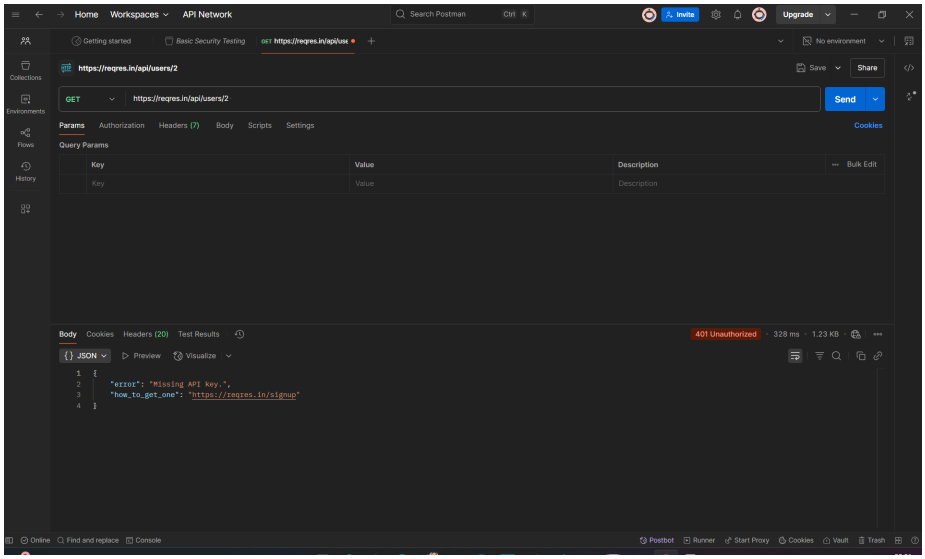

Figure 4: Authentication Check

Step 5: Input Validation Test :

Method: A POST request was made to the API with incorrect login credentials (for example, invalid email and password) and how the API responded was observed.

Result: The API returned correct error messages for invalid logins, returning a 400 Bad Request error.

Security Area Tested: Input validation and user data security.

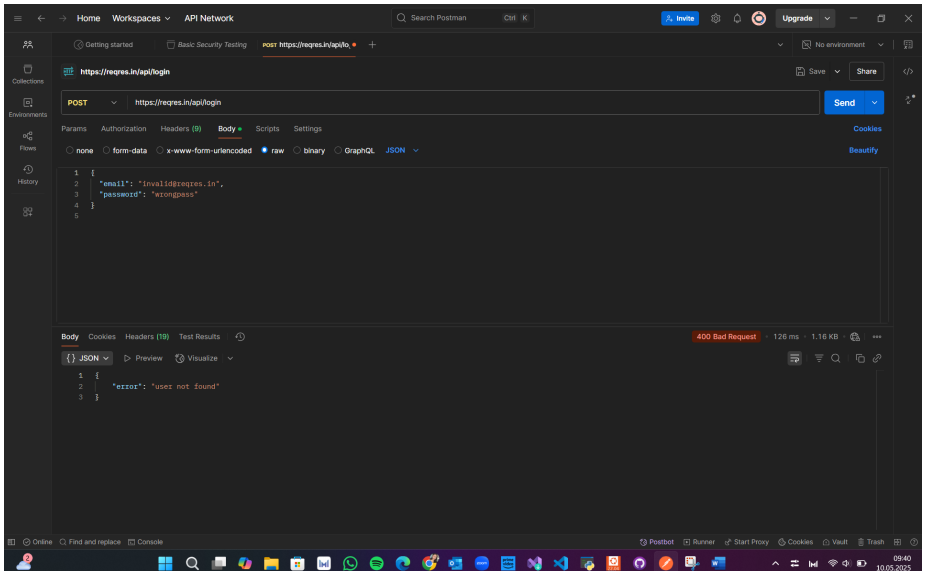

Figure 5: Input Validation Test

Step 6: Error Handling Test:

Method: A GET request with an invalid URL was sent to test how the API responds to bad requests.

Result: The API returned a 404 Not Found error to bad URLs and error messages were handled correctly.

Security Area Tested: Error handling and information privacy.

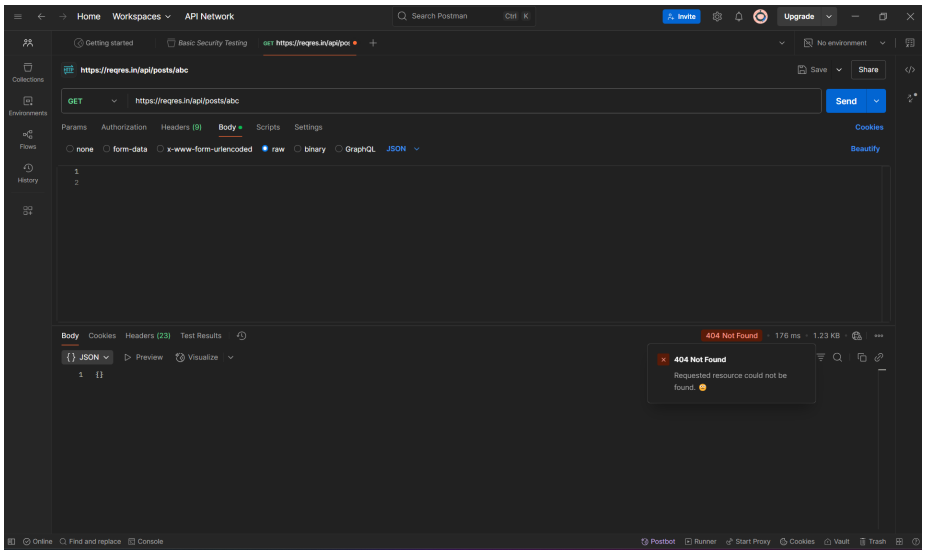

Figure 6: Error Handling Test

Step 7: Simple SQL Injection Test:

Method: A common test attempt for SQL injection (1' OR '1'='1) was added to the URL to observe how the API responded.

Result: The API returned 404 Not Found errors against SQL injection inputs, indicating that SQL injection was effectively blocked.

Security Area Tested: SQL injection and API security measures.

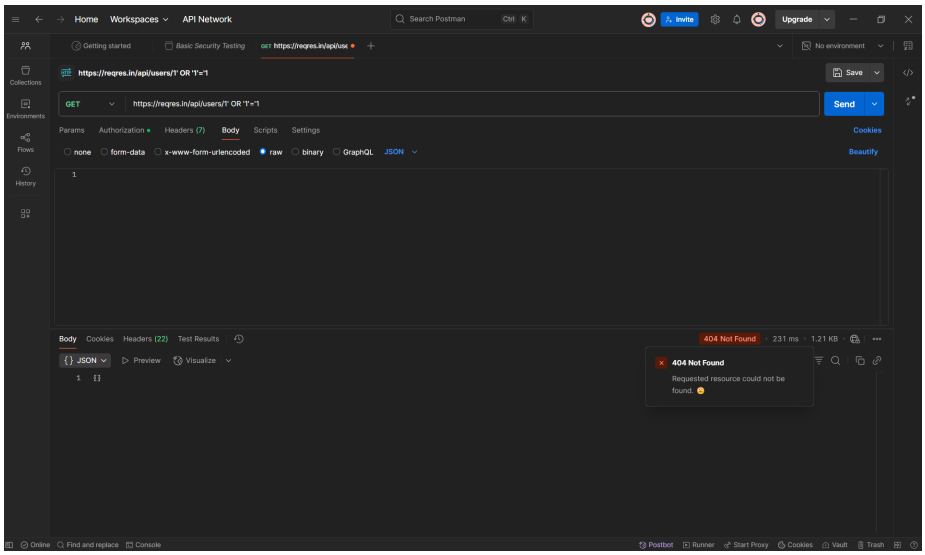

Figure 7: Simple SQL Injection Test

4 Conclusion

In this study, we did basic security tests on an API using Postman. Our goal was to understand if the API is secure or not. We tested things like authentication, API key usage, wrong input, error messages, and SQL injection.

According to the results:

The API works correctly and gives successful responses with a valid API key.

Without the API key, some data cannot be accessed, which shows that security is working.

When wrong information is entered, the system gives the correct error message.

If a wrong URL is used, it returns a 404 Not Found error, which is a good result.

During the SQL injection test, the API gives an error but does not show sensitive information. This is a positive result.

In conclusion, these tests were useful to improve API security. If such tests are done regularly, systems can become more secure.