### **1. Valid Token**

* Test Case: Verify API response when a valid token is used.
* Precondition: Obtain a valid OAuth2 bearer token.
* Steps:
  1. Make a GET request to the API with Authorization: Bearer <valid\_token>.
  2. Observe the response.
* Expected Result: API should return 200 OK with the expected data.

### **2. Expired Token**

* Test Case: Verify API response when using an expired token.
* Precondition: Obtain a previously valid token that has expired.
* Steps:
  1. Make a GET request to the API with Authorization: Bearer <expired\_token>.
  2. Observe the response.
* Expected Result: API should return 401 Unauthorized with an error message indicating token expiration.

### **3. Revoked Token**

* Test Case: Verify API response when using a revoked token.
* Precondition: Obtain a token that was valid but has been revoked.
* Steps:
  1. Make a GET request to the API with Authorization: Bearer <revoked\_token>.
  2. Observe the response.
* Expected Result: API should return 401 Unauthorized.

### **4. Invalid Token**

* Test Case: Verify API response when using an invalid token.
* Precondition: Use a randomly generated string as a token.
* Steps:
  1. Make a GET request to the API with Authorization: Bearer <invalid\_token>.
  2. Observe the response.
* Expected Result: API should return 401 Unauthorized with an appropriate error message.

### **5. Missing Token**

* Test Case: Verify API response when no token is provided.
* Precondition: No token is provided in the request.
* Steps:
  1. Make a GET request to the API without an Authorization header.
  2. Observe the response.
* Expected Result: API should return 401 Unauthorized.

### **6. Token in Query Parameter (If applicable)**

* Test Case: Verify API response when the token is passed in the query string.
* Precondition: API should not allow tokens in the query string (security best practice).
* Steps:
  1. Make a GET request like:

bash

CopyEdit

GET /api/resource?access\_token=<valid\_token>

* 1. Observe the response.
* Expected Result: API should return 400 Bad Request or 401 Unauthorized.

### **7. Insufficient Scope**

* Test Case: Verify API response when using a token with insufficient permissions.
* Precondition: Use a valid token with limited access.
* Steps:
  1. Make a GET request to an API endpoint requiring higher privileges.
  2. Observe the response.
* Expected Result: API should return 403 Forbidden with an appropriate error message.

### **8. Valid Token with Different User Roles**

* Test Case: Verify API response for different user roles.
* Precondition: Obtain valid tokens for different roles (e.g., admin, user, guest).
* Steps:
  1. Make GET requests using different tokens.
  2. Observe the response.
* Expected Result: API should return appropriate data based on the user's role.

### **9. Rate Limiting**

* Test Case: Verify rate limiting enforcement.
* Precondition: Obtain a valid token.
* Steps:
  1. Send multiple consecutive requests in a short time.
  2. Observe the response after exceeding the limit.
* Expected Result: API should return 429 Too Many Requests after the limit is exceeded.

### **10. Token with Extra Spaces**

* Test Case: Verify API behavior with extra spaces in the token.
* Precondition: Obtain a valid token.
* Steps:
  1. Send a request with Authorization: Bearer <valid\_token> (extra spaces).
  2. Observe the response.
* Expected Result: API should trim the token and return 200 OK.

### **11. Replay Attack**

* Test Case: Verify API defense against token replay attacks.
* Precondition: Obtain a valid token and use it for an authenticated request.
* Steps:
  1. Capture the request with the token.
  2. Replay the exact request multiple times.
  3. Observe the response.
* Expected Result: API should prevent reuse of the same token beyond its intended use.

### **12. Cross-Origin Requests (CORS)**

* Test Case: Verify CORS policy for bearer token requests.
* Precondition: Set up a test with a different origin (e.g., another domain).
* Steps:
  1. Make a GET request from a different origin.
  2. Observe the response.
* Expected Result: API should enforce proper CORS policies (either allowing or denying the request as per security configuration).

### **13. Token Exposure in Logs**

* Test Case: Verify that tokens are not logged in API responses or server logs.
* Precondition: Enable API request logging.
* Steps:
  1. Make an authenticated request.
  2. Inspect logs to check if the token is exposed.
* Expected Result: Tokens should not be present in logs or error messages.