DOKRAVE

ILLUSTRATIVE EXAMPLE | SYNTHETIC QA DEMONSTRATION

API Order Creation

Functional, authentication and data-validation testing

Scenarios	Passed	Failed	With workaround	Findings
6	5	1	0	1

Overview

A synthetic demonstration of testing an API endpoint that creates business orders. It covers request handling, authentication, input validation and verification that submitted data is correctly stored.

Testing objective

Verify that an order-creation endpoint accepts valid authenticated requests, rejects unauthorised or invalid requests, and persists submitted information consistently.

Scope

In scope

- Successful order creation using a valid authenticated account.
- Access control when no valid authentication is supplied.
- Required-field and malformed-input handling.
- Verification of created order data in the application and underlying data store.
- Retesting after an authentication defect is corrected.

Out of scope

- Full penetration testing or a formal security assessment.
- Load, stress or performance testing.
- Testing every API endpoint or all possible business rules.

Test approach

1. Review the endpoint purpose and expected request structure.
2. Use an API client or interactive API documentation to submit requests.
3. Compare responses with expected behaviour.
4. Verify persisted order data through the application and, where appropriate, read-only database queries.
5. Record defects with reproducible steps and retest after a change.

Test scenarios and results

ID	Scenario	Steps and expected result	Result
API-01	Valid authenticated request	**Steps:** Submit a complete, valid order request with a permitted account. **Expected:** The endpoint accepts the request and returns the documented success response and order reference.	**PASS** Pass in this illustrative scenario.
API-02	Request without authentication	**Steps:** Submit an otherwise valid request without credentials. **Expected:** The request is rejected; no order is created.	**FAIL** Failed. The unauthenticated request was accepted and an order could be inserted.
API-03	Missing required field	**Steps:** Omit a field defined as mandatory in the API contract. **Expected:** The API returns a clear validation response; no incomplete order is created.	**PASS** Pass in this illustrative scenario.
API-04	Persisted data verification	**Steps:** Retrieve the created order in the application and in the database, and verify the relevant stored fields. **Expected:** Stored values match the submitted request and business rules.	**PASS** Pass in this illustrative scenario.
API-05	Authentication defect retest	**Steps:** Repeat the unauthenticated request after the access-control change. **Expected:** The request is rejected and no order is created.	**PASS** Pass on retest.
API-06	Full repeat after the fix	**Steps:** Repeat API-01, API-03 and API-04 after the access-control change to confirm nothing else was affected. **Expected:** All earlier results still hold.	**PASS** Pass. The change was approved.

Findings

API-F01 Unauthorised order creation

Severity / impact: High

The endpoint accepted a valid order-creation request without an authenticated session or valid credentials. This could allow unauthorised order creation.

Retest / follow-up: The access-control behaviour was corrected and the unauthenticated request was repeated. The request was then rejected, and the earlier scenarios were repeated successfully.

Outcome

This demonstration combines functional API checks with access-control verification and data-persistence checks. It illustrates a defect loop: reproduce, document, verify the change and retest. It is not a full security audit.

Illustrative request (generic data)

```
POST /api/orders
(no Authorization header)

{
  "customerId": "C-1042",
  "currency": "EUR",
  "lines": [
    { "sku": "SKU-001", "quantity": 2, "unitPrice": 19.90 }
  ]
}

Expected:  401 Unauthorized, no order created
Observed:  201 Created, order inserted   (defect API-F01)
```

Tools and techniques

Illustrative tools: Postman or Swagger UI for API requests; application-level verification; read-only SQL queries where database access is available.

Important note. Synthetic demonstration. Names, data and system details are generic. Outcomes are illustrative and are not a claim about a named customer or production system.