@author Bhimashankar Teli

Test Plan: Security and Performance Testing for All APIs

Introduction

This document outlines the strategy and approach for conducting security and performance testing for all APIs in the application. The objectives are to identify vulnerabilities, ensure secure API interactions, and validate performance metrics under expected and stress conditions.

---

1. Security Testing

Objective

To identify and mitigate vulnerabilities in the API endpoints, ensuring data confidentiality, integrity, and availability.

Scope

- All exposed REST API endpoints.
- Authentication and authorization mechanisms.
- Data input validation and protection against common security threats.

Security Testing Scenarios

1.1 Authentication and Authorization

- Validate proper implementation of token-based authentication (e.g., OAuth2, JWT).
- Test role-based access control (RBAC) for restricted endpoints.
- Verify session expiration and refresh token functionality.

1.2 Input Validation

- Test against SQL Injection by providing malicious inputs.
- Validate protection against Cross-Site Scripting (XSS) attacks.
- Check for buffer overflow vulnerabilities with oversized payloads.

1.3 API Headers

- Validate presence of security headers like X-Content-Type-Options, X-Frame-Options, Strict-Transport-Security.
- Ensure sensitive data (e.g., tokens) are not exposed in headers or responses.

1.4 Rate Limiting and Brute Force Protection

- Verify rate limiting mechanisms to prevent excessive API calls.

- Simulate multiple failed login attempts to ensure brute force protection.

1.5 Error Handling

- Test for generic error messages that do not leak sensitive information.
- Ensure stack traces and internal details are not exposed in API responses.

Tools

- OWASP ZAP/Burp Suite for penetration testing.
- Postman for manual security validation.
- Rest Assured for automated tests.

Deliverables

- Security test cases.
- List of identified vulnerabilities with risk severity.
- Mitigation recommendations.

---

2. Performance Testing

Objective

To ensure the APIs perform optimally under varying loads and meet the defined SLA requirements.

Scope

- Validate response time, throughput, and error rates.
- Conduct load, stress, and scalability tests.

Performance Testing Scenarios

2.1 Load Testing

- Simulate concurrent requests to measure the API's performance under typical loads.
- Validate response times are within acceptable thresholds (e.g., <200ms).

2.2 Stress Testing

- Test API behavior under peak loads beyond expected traffic.
- Identify the breaking point and system recovery time.

2.3 Scalability Testing

- Measure the system's ability to handle increased traffic by scaling resources.

- Test API response times with varying backend database sizes.

2.4 Endurance Testing

- Execute prolonged API calls to test for memory leaks and resource exhaustion.

2.5 Capacity Planning

- Determine the maximum number of simultaneous users the API can support without performance degradation.

Tools

- JMeter for load and stress testing.
- Gatling for scalability and endurance tests.
- New Relic/Dynatrace for monitoring server performance.

Key Metrics

- **Response Time**: Time taken to respond to a request.
- **Throughput**: Number of requests handled per second.
- **Error Rate**: Percentage of failed requests.
- **Resource Utilization**: CPU, memory, and network usage during tests.

Deliverables

- Performance test scripts.
- Detailed test execution reports.
- Recommendations for performance improvements.

Testing Process

1. **Preparation**:
   - Identify API endpoints, expected traffic, and SLA requirements.
   - Configure test environments to simulate real-world conditions.
2. **Execution**:
   - Run security and performance tests iteratively.
   - Log issues and anomalies during test runs.
3. **Reporting**:
   - Summarize findings with detailed insights into vulnerabilities and performance bottlenecks.

- Provide actionable recommendations for fixing identified issues.

4. **Verification**:
    - Retest APIs after fixes to confirm issues are resolved.
    - Perform regression testing to ensure no new issues were introduced.
