# API Testing Strategy & Coverage

Comprehensive testing strategy for the RAS Dashboard API with prioritized test suites covering critical security, compliance, and business logic functionality.

## 🎯 Testing Overview

### Current Test Coverage Status

| Priority | Category | APIs Tested | APIs Missing | Coverage |
| --- | --- | --- | --- | --- |
| **Priority 1** | Critical APIs | 5/5 | 0 | ✅ 100% |
| **Priority 2** | Business Logic | 4/15 | 11 | 🟡 27% |
| **Priority 3** | Integration | 0/4 | 4 | ❌ 0% |
| **Priority 4** | AI/Analytics | 0/7 | 7 | ❌ 0% |
| **Overall** | All APIs | 9/31 | 22 | 🟡 29% |

## 📋 Test Suites by Priority

### ✅ Priority 1: Critical APIs (Security & Compliance)

**Status: COMPLETE** - All critical APIs have comprehensive test coverage

1. **RBAC Service Tests** (rbac.test.js)
   * ✅ Role management CRUD operations
   * ✅ Permission management
   * ✅ Role-permission assignments
   * ✅ User permission checking
   * ✅ Security validation & access control
   * ✅ Edge cases & error handling
2. **Audit Log Service Tests** (auditLog.test.js)
   * ✅ Audit log creation & retrieval
   * ✅ Filtering & search functionality
   * ✅ Statistics & analytics
   * ✅ Compliance features
   * ✅ Data integrity & immutability
   * ✅ High-volume logging performance
3. **Notification Service Tests** (notifications.test.js)
   * ✅ Notification CRUD operations
   * ✅ Real-time notification delivery
   * ✅ Bulk operations
   * ✅ User-specific filtering
   * ✅ Statistics & analytics
   * ✅ Performance under load
4. **Authentication Tests** (auth.test.js) - *Existing*
   * ✅ Login/logout functionality
   * ✅ JWT token management
   * ✅ Password validation
   * ✅ Session management
5. **User Management Tests** (users.test.js) - *Existing*
   * ✅ User CRUD operations
   * ✅ Profile management
   * ✅ Role assignments
   * ✅ Access control

### 🟡 Priority 2: Business Logic APIs (Partially Complete)

**Status: 4/15 COMPLETE** - Core business functionality

#### ✅ Completed:

1. **Asset Management Tests** (assetManagement.test.js)
   * ✅ Asset CRUD operations
   * ✅ Cost management
   * ✅ Lifecycle tracking
   * ✅ Analytics & reporting
   * ✅ Search & filtering
2. **STIG Service Tests** (stig.test.js)
   * ✅ STIG library management
   * ✅ Assessment creation & execution
   * ✅ Finding management
   * ✅ Compliance analytics
   * ✅ Security validation
3. **Scanner Integration Tests** (scanner-integration.test.js) - *Existing*
   * ✅ Scan execution (internal, vulnerability, compliance)
   * ✅ Results management
   * ✅ Job status tracking
4. **Settings Management Tests** (settings.test.js) - *Existing*
   * ✅ Settings CRUD operations
   * ✅ Data type handling
   * ✅ Public/private settings
   * ✅ Bulk operations

#### ❌ Missing Tests (11 APIs):

* dashboardService.js - Dashboard metrics & widgets
* reportService.js - Report generation & management
* policyService.js - Policy management
* procedureService.js - Procedure management
* artifactService.js - Artifact management
* atoService.js - Authorization to Operate
* emailService.js - Email functionality
* emailTemplateService.js - Email templates
* metricsService.js - System metrics
* moduleService.js - Module management
* siemService.js - SIEM integration

### ❌ Priority 3: Integration APIs (Not Started)

**Status: 0/4 COMPLETE** - External system integrations

1. **Tenable Service** (tenableService.js)
   * Vulnerability data integration
   * Asset synchronization
   * Scan management
2. **Xacta Service** (xactaService.js)
   * System data integration
   * Control mapping
   * Compliance tracking
3. **NVD Service** (nvdService.js)
   * CVE data integration
   * Vulnerability enrichment
   * Database synchronization
4. **Email Service** (emailService.js)
   * SMTP integration
   * Template processing
   * Delivery tracking

### ❌ Priority 4: AI/Analytics APIs (Not Started)

**Status: 0/7 COMPLETE** - Advanced AI features

1. **AI Assistance Service** (aiAssistanceService.js)
2. **AI Compliance Service** (aiComplianceService.js)
3. **AI Cost Optimization** (aiCostOptimizationService.js)
4. **AI Generation Service** (aiGenerationService.js)
5. **AI Security Training** (aiSecurityTrainingService.js)
6. **AI Threat Hunting** (aiThreatHuntingService.js)
7. **Natural Language Query** (naturalLanguageQueryService.js)

## 🚀 Running Tests

### Quick Test Execution

# Run all priority tests with summary  
node scripts/run-priority-tests.js  
  
# Run with coverage report  
node scripts/run-priority-tests.js --coverage  
  
# Run with verbose output  
node scripts/run-priority-tests.js --verbose --coverage

### Individual Test Suites

# Run specific test suite  
npm test -- tests/rbac.test.js  
npm test -- tests/auditLog.test.js  
npm test -- tests/notifications.test.js  
npm test -- tests/assetManagement.test.js  
npm test -- tests/stig.test.js  
  
# Run all tests  
npm test  
  
# Run with coverage  
npm run test:coverage

### Test Categories

# Critical APIs only  
npm test -- --testPathPattern="(rbac|auditLog|notifications|auth|users).test.js"  
  
# Business Logic APIs  
npm test -- --testPathPattern="(assetManagement|stig|scanner|settings).test.js"  
  
# All priority tests  
npm test -- --testPathPattern="(rbac|auditLog|notifications|assetManagement|stig|auth|users|scanner|settings).test.js"

## 📊 Test Quality Standards

### Test Coverage Requirements

* **Critical APIs (Priority 1)**: 95%+ coverage required
* **Business Logic (Priority 2)**: 85%+ coverage required
* **Integration APIs (Priority 3)**: 75%+ coverage required
* **AI/Analytics (Priority 4)**: 70%+ coverage required

### Test Types Included

1. **Unit Tests** - Individual function testing
2. **Integration Tests** - API endpoint testing
3. **Security Tests** - Authentication & authorization
4. **Performance Tests** - Load & stress testing
5. **Error Handling** - Edge cases & failure scenarios
6. **Data Validation** - Input validation & sanitization

### Test Patterns

Each test suite follows consistent patterns: - ✅ **Setup/Teardown** - Clean test environment - ✅ **Authentication** - Token-based testing - ✅ **CRUD Operations** - Complete lifecycle testing - ✅ **Validation** - Input/output validation - ✅ **Permissions** - Role-based access control - ✅ **Error Handling** - Comprehensive error scenarios - ✅ **Edge Cases** - Boundary conditions - ✅ **Performance** - Load testing where applicable

## 🎯 Next Steps & Recommendations

### Immediate Actions (Priority 2 Completion)

1. **Dashboard Service Tests** - Core UI functionality
2. **Report Service Tests** - Critical for compliance
3. **Policy/Procedure Tests** - Governance requirements
4. **Email Service Tests** - System notifications

### Medium Term (Priority 3)

1. **Integration Testing** - External system connectivity
2. **End-to-End Testing** - Complete workflow validation
3. **Performance Testing** - Load & stress testing
4. **Security Testing** - Penetration testing scenarios

### Long Term (Priority 4)

1. **AI Service Testing** - Machine learning validation
2. **Analytics Testing** - Data processing validation
3. **Natural Language Testing** - Query processing
4. **Advanced Features** - Cutting-edge functionality

## 📈 Test Metrics & Monitoring

### Key Performance Indicators

* **Test Coverage**: Target 85%+ overall
* **Test Success Rate**: Target 95%+ pass rate
* **Test Execution Time**: Target <5 minutes for full suite
* **Test Reliability**: Target <1% flaky tests

### Continuous Integration

* **Pre-commit Hooks**: Run critical tests before commit
* **Pull Request Validation**: Full test suite on PR
* **Deployment Gates**: 100% critical test pass required
* **Nightly Builds**: Full regression testing

### Quality Gates

1. **Code Coverage** - Minimum thresholds enforced
2. **Test Pass Rate** - No failing tests in main branch
3. **Performance** - Response time thresholds
4. **Security** - Vulnerability scanning integration

## 🛠️ Test Infrastructure

### Test Environment

* **Database**: Isolated test database with clean state
* **Authentication**: Mock JWT tokens for testing
* **External APIs**: Mocked services for integration tests
* **File System**: Temporary directories for file operations

### Test Data Management

* **Fixtures**: Predefined test data sets
* **Factories**: Dynamic test data generation
* **Cleanup**: Automatic test data cleanup
* **Isolation**: Each test runs in isolation

### Reporting & Analytics

* **Coverage Reports**: HTML & JSON coverage reports
* **Test Results**: JUnit XML for CI integration
* **Performance Metrics**: Test execution timing
* **Trend Analysis**: Historical test performance

The testing strategy ensures comprehensive coverage of critical functionality while maintaining high quality standards and supporting continuous integration workflows.