

Security Testing

Source File: generated-documents\quality-assurance\security-testing.md

Generated: 15/07/2025 at 11:38:32

Generated by: Requirements Gathering Agent - PDF Converter

Security Testing

Generated by adpa-enterprise-framework-automation v3.2.0

Category: quality-assurance

Generated: 2025-07-14T21:07:04.483Z

Description: Security testing procedures and validation

Security Testing Plan

Project: ADPA - Advanced Document Processing & Automation Framework

Version: 3.2.0

Date: 2025-07-08

Author: Security Testing Lead

1. Security Testing Overview

1.1 Security Testing Objectives and Goals

- **Protect Confidentiality, Integrity, and Availability** of all components (CLI, REST API, Admin Web, integrations).
- **Identify and remediate vulnerabilities** in authentication, authorization, input validation, data storage, and integrations (Adobe, SharePoint, Confluence, AI Providers).
- **Ensure regulatory and standards compliance** (GDPR, SOX, PCI DSS, HIPAA, ISO 27001, OWASP).
- **Enable secure enterprise deployments** for Fortune 500 and regulated industries.

1.2 Security Risk Assessment and Threat Modeling

- **Attack Surfaces:**
 - Public REST API endpoints (Express.js, OpenAPI).
 - CLI and Admin Web Interface.
 - Integration points (Adobe APIs, SharePoint, Confluence, AI providers).
 - OAuth2/SAML/Active Directory-based identity management.
 - Document generation (input/output handling, template injection).
 - Data persistence (configuration files, logs, generated documents, secrets).
- **Threats Identified:**
 - API abuse (rate limiting bypass, unauthorized access).
 - Credential theft, privilege escalation, session hijacking.
 - Injection attacks (SQL/NoSQL, Command, XSS, Template Injection).
 - Sensitive data exposure (PII, API keys in logs, document leaks).
 - Supply chain attacks via dependencies or provider compromise.
 - Insider threats (malicious admins, privilege misuse).
 - Denial of Service (DoS) and resource exhaustion.
 - Weak cryptography and poor key management.
 - Compliance violations (data residency, audit trail gaps).

1.3 Compliance Requirements and Standards

- **Regulatory:** GDPR, HIPAA, SOX, PCI DSS, FINRA, Basel III, MiFID II.
- **Industry Standards:** OWASP Top 10, NIST CSF, ISO 27001, CIS Controls, SANS Top 25.
- **Framework Compliance:** BABOK v3, PMBOK 7th, DMBOK 2.0.

1.4 Security Testing Scope and Limitations

- **In Scope:**
 - All user-facing APIs, CLI, Web UI.
 - Authentication/authorization flows (API keys, JWT, OAuth2, SAML, AD).
 - Integrations (Adobe, SharePoint, Confluence, AI providers).
 - Document processing pipeline (uploads, template generation).
 - Configuration and data storage (including .env, secrets).
 - **Out of Scope:**
 - Third-party provider infrastructure (OpenAI, Google AI, Adobe, etc.).
 - Underlying platform security (Node.js/OS hardening) except where misconfiguration impacts ADPA.
-

2. Security Test Strategy

2.1 Authentication Testing

- **Objective:** Validate identity management (API Key, JWT, OAuth2, SAML, AD).
- Test session management: token expiry, revocation, refresh flows.
- MFA/2FA enforcement (where applicable).
- Test for credential stuffing, brute force, default credentials.

2.2 Authorization Testing

- **Objective:** Ensure strict role-based and permission-based access control.

- Test all user roles (admin, project_manager, analyst, stakeholder, viewer).
- Attempt privilege escalation, horizontal/vertical access bypass.
- Validate resource ownership and team boundary enforcement.

2.3 Input Validation Testing

- **Objective:** Prevent injection and malformed data attacks.
- Test all endpoints for SQL/NoSQL injection, XSS, command injection, template injection.
- Fuzz input fields for REST API, CLI arguments, file uploads.
- Validate content-type, size, and structure of uploaded documents and templates.

2.4 Data Protection Testing

- **Objective:** Ensure encryption, integrity, and privacy.
- Test encryption for data at rest (config, secrets, documents) and in transit (HTTPS, API calls).
- Inspect logging for sensitive data leakage (PII, tokens, passwords).
- Test document output for unintentional metadata or PII exposure.

2.5 Network Security Testing

- **Objective:** Protect against network-based attacks.
- Validate HTTPS/TLS enforcement, secure headers (HSTS, CSP, X-Frame).
- Scan for open/unused ports, misconfigured CORS, rate limit evasion.
- Test public endpoints for DoS susceptibility.

2.6 Application Security Testing

- **Objective:** Identify code-level and configuration vulnerabilities.
 - Static code analysis (TypeScript, Node.js).
 - Dynamic application security testing (DAST) for running services.
 - Dependency analysis for vulnerable npm packages.
-

3. Security Test Types and Methodologies

3.1 Vulnerability Assessment

- Use automated scanners (OWASP ZAP, Nessus, OpenVAS, npm audit).
- Map findings to OWASP Top 10 and SANS Top 25.

3.2 Penetration Testing

- Manual exploitation of high-risk areas (auth flows, integrations, uploads).
- Custom test scripts for business logic and workflow bypass.

3.3 Security Code Review

- Static analysis (SonarQube, Checkmarx, Veracode).
- Manual review of authentication, authorization, input validation, crypto routines.

3.4 Configuration Testing

- Validate secure configuration (CSP, CORS, rate limiting, environment isolation).
- Hardening review for Express.js, Node.js, TypeScript settings.

3.5 Compliance Testing

- Use compliance scanners (Qualys, PCI DSS tools, GDPR checkers).
- Review audit trails, logging, and data retention policies.

3.6 Social Engineering Testing

- (Optional/Enterprise Only) Simulate phishing or privilege escalation via social means (if client authorizes).
-

4. Security Test Scenarios

4.1 Authentication Attacks

- **Brute Force:** Attempt repeated logins with common/user-specific passwords.
- **Credential Stuffing:** Use leaked credential lists against API endpoints.
- **Session Hijacking:** Steal session tokens via XSS, inspect for JWT weaknesses.
- **Token Replay:** Use old/expired tokens to access protected resources.

4.2 Authorization Bypass

- **Privilege Escalation:** Access admin endpoints as lower-privilege user.
- **Horizontal Access:** Access another team's documents or templates.
- **Direct Object Reference:** Manipulate IDs/URLs to access unauthorized resources.

4.3 Injection Attacks

- **SQL/NoSQL Injection:** Test API, CLI, and template processing.
- **Command Injection:** Attempt shell injection via upload/file processing.
- **XSS:** Inject scripts into document templates, metadata, or logs.
- **Template Injection:** Malicious payloads in template files (Markdown, JSON).

4.4 Data Exposure

- **Sensitive Data Leakage:** Inspect API responses, logs, generated docs for PII, API keys, secrets.
- **Unintended Metadata:** Analyze PDFs and generated files for hidden/embedded sensitive data.

- **Information Disclosure:** Error messages, stack traces, version details.

4.5 Denial of Service

- **Resource Exhaustion:** Flood API endpoints, abuse file upload and document generation.
- **Logic Bombs:** Submit malformed or recursive templates causing server lockup.
- **API Rate Limit Bypass:** Test for unlimited requests via multiple vectors.

4.6 Cryptographic Failures

- **Weak Encryption:** Review TLS/HTTPS ciphers, document encryption algorithms.
 - **Improper Key Management:** Test for keys in repo, .env, or logs.
 - **Token Predictability:** Analyze JWT and API key generation methods.
-

5. Security Testing Tools and Technologies

5.1 Vulnerability Scanners

- **OWASP ZAP:** Automated DAST for APIs and Web UI.
- **Nessus/OpenVAS:** Network and OS-level vulnerability scans.
- **npm audit/Retire.js:** Dependency vulnerability analysis.

5.2 Penetration Testing Tools

- **Burp Suite Pro:** Manual and automated attack surface exploration.
- **Metasploit:** Exploit verification (where possible).
- **Nmap:** Port/service discovery.
- **Wireshark:** Packet-level inspection.

5.3 Static Analysis Tools

- **SonarQube Security, Checkmarx, Veracode:** Automated code analysis.
- **ESLint/Prettier:** Enforce secure coding standards.

5.4 Dynamic Analysis Tools

- **OWASP ZAP, Burp Suite Intruder, DAST plugins.**

5.5 Compliance Scanners

- **Qualys, Tenable PCI DSS, GDPR tools:** Automated compliance validation.
-

6. Security Test Environment

6.1 Environment Setup and Isolation

- Dedicated, isolated test environment (no production data).
- Network segmentation for test, staging, and integration environments.
- Read-only test accounts for third-party providers.

6.2 Test Data Security

- Use synthetic/non-production data for all testing.
- Ensure test data and logs are purged post-testing.

6.3 Network Security Controls

- Enable monitoring, logging (Winston, Morgan), and alerting for all test activities.
- Ensure firewalls, IDS/IPS, and segmentation are active.

6.4 Monitoring and Logging

- Enable verbose logging and audit trails for all security-relevant actions.
 - Capture and review all authentication, authorization, and integration events.
-

7. Security Test Execution

7.1 Phases and Timeline

- **Phase 1:** Automated scanning and static analysis.
- **Phase 2:** Manual penetration testing and business logic tests.
- **Phase 3:** Configuration and compliance validation.
- **Phase 4:** Remediation and validation retesting.

7.2 Test Result Analysis and Classification

- Categorize findings: Critical, High, Medium, Low, Informational.
- Map findings to CVSS and OWASP risk ratings.

7.3 Security Incident Response (During Testing)

- Immediate reporting of critical findings.
- Isolate compromised test environments.
- Document all incidents and root cause analysis.

7.4 Remediation Tracking and Validation

- Log all findings in tracking system (e.g., Jira, GitHub Issues).
 - Validate fixes via regression and retesting.
 - Require evidence of remediation for compliance closure.
-

8. Security Compliance and Standards

8.1 Regulatory Compliance

- **GDPR:** Data minimization, user consent, right to erasure.
- **HIPAA:** PHI handling, access control, audit trails.
- **PCI DSS:** Secure handling of payment data (if applicable).
- **SOX:** Change management, audit logging, financial data controls.

8.2 Industry Standards

- **OWASP Top 10:** Reference all findings and mitigations.
- **NIST Cybersecurity Framework & CIS Controls:** Map controls and test coverage.
- **ISO 27001:** Information security management system (ISMS) alignment.

8.3 Compliance Validation

- Maintain detailed audit trails for all critical actions.
 - Generate formal compliance reports for each test phase.
 - Provide executive dashboards and evidence packs for external audits.
-

9. Security Metrics and Reporting

9.1 Vulnerability Metrics and KPIs

- Number of vulnerabilities by severity and category.
- Mean time to remediation (MTTR) per severity.
- Percentage of critical/high issues resolved before production release.
- Test coverage ratio (endpoints, roles, integrations).

9.2 Risk Assessment and Impact Analysis

- Business impact analysis for each finding.

- Risk scoring per asset and integration.

9.3 Test Coverage and Effectiveness

- Coverage map: API endpoints, roles, integrations (Adobe, SharePoint, etc.).
- False positive/negative rates for scanners.

9.4 Executive Security Reporting

- Summarized dashboards for leadership (compliance posture, risk exposure, remediation status).
 - Detailed technical appendices for engineering.
-

10. Security Improvement and Remediation

10.1 Remediation Priorities and Timelines

- **Critical:** Fix within 24-48 hours.
- **High:** Fix within 5 business days.
- **Medium:** Fix within 15 business days.
- **Low/Informational:** Track for future sprints.

10.2 Security Control Implementation and Validation

- Enforce strong MFA, password policies, and RBAC.
- Implement least-privilege and zero-trust principles on all integrations.
- Harden Express.js and Node.js configurations (disable x-powered-by, use Helmet, enforce CORS).
- Rotate secrets and API keys regularly.

10.3 Continuous Security Monitoring and Improvement

- Integrate SAST/DAST into CI/CD pipeline.
- Schedule periodic dependency reviews and patching.
- Conduct annual/biannual external penetration tests.

10.4 Security Awareness and Training

- Regular security training for developers and admins.
- Secure coding guideline enforcement (TypeScript, Node.js best practices).
- Phishing and social engineering simulation exercises (where authorized).

End of Document — For use by ADPA Security Testing Teams and Stakeholders
