Security Design

> **Source File:** generated-documents\technical-design\security-design.md
> **Generated:** 16/07/2025 at 14:00:43
> **Generated by:** Requirements Gathering Agent - PDF Converter

SecurityDesign

Generated by adpa-enterprise-framework-automation v3.2.0
Category: technical-design
Generated: 2025-07-14T21:04:13.268Z
Description:

ADPA (Advanced Document Processing & Automation Framework)

Security Design Document

---

1. Security Overview

ADPA is an enterprise-grade automation framework for document generation, project management, and business analysis, supporting both CLI and REST API interfaces. Security is a central architectural concern, with design patterns and controls to support regulatory compliance

(GDPR, SOX, PCI DSS, etc.), multi-tenant enterprise deployments, and integration with external systems (AI Providers, SharePoint, Confluence, Adobe, etc.).

Key Security Objectives:

- **Confidentiality:** Safeguard enterprise and client data at rest and in transit.
- **Integrity:** Prevent unauthorized or accidental modification of documents and templates.
- **Availability:** Ensure system and data accessibility for authorized users.
- **Accountability:** Maintain audit trails and traceability for all actions.
- **Compliance:** Meet requirements of GDPR, SOX, PCI DSS, and other standards.
- **Extensibility:** Allow for secure integration with new providers and services.

---

2. Authentication Design

2.1 User Authentication

- **API:**
    - **JWT (JSON Web Token)**: Used for stateless authentication.
        - Tokens signed with strong secret (ideally asymmetric keys in production).
        - Short token expiry with refresh token mechanism.
    - **OAuth 2.0 / OpenID Connect:**
        - For enterprise SSO (e.g., Azure AD, SAML, Active Directory).
        - Used for integrations (SharePoint, Confluence, Adobe APIs).
    - **API Key:**
        - Available for automation and CLI users (least privileged, rotate regularly).

        - Sent via secure HTTP header (`X-API-Key`).

- **CLI:**
    - Supports both API Key and interactive OAuth2 login for publishing/integration.
    - Secrets and tokens stored in encrypted configuration files or OS credential stores.
- **Admin Web Interface:**
    - OAuth2-based authentication with enterprise SSO support.
    - Session tokens stored using `HttpOnly` , `Secure` cookies.

2.2 Provider Authentication

- **AI Providers:**
    - API keys/secrets stored in environment variables or secure vaults.
    - No hard-coded credentials.
    - Rate limits and usage monitoring enforced.
- **Integration Providers (Adobe, SharePoint, Confluence):**
    - OAuth2 authorization code/device flow.
    - Least-privilege principle: only request required scopes.

Best Practices:

- Enforce strong password policies and MFA where supported.
- Lock accounts after repeated failed attempts.
- Use secure storage for secrets (Azure Key Vault, AWS Secrets Manager, etc.).

---

3. Authorization Framework

3.1 Role-Based Access Control (RBAC)

- **User Roles:**
  - `admin`, `project_manager`, `business_analyst`, `stakeholder`, `viewer`.
- **Resource Permissions:**
  - Fine-grained permissions on resources: `projects`, `documents`, `templates`, `users`, `integrations`.
- **API Enforcement:**
  - All API endpoints validate JWT and user role before processing.
  - CLI commands check permissions before sensitive actions.
- **Approval Workflows:**
  - For document publishing, changes, and project approvals.
- **Granular Audit Trail:**
  - All permission changes, resource access, and critical operations are logged.

3.2 Integration Authorization

- **Delegated Permissions:**
  - Only allow API integrations (SharePoint, Confluence) to access resources authorized for the current user.
- **Consent & Revocation:**
  - Users can revoke integration consent at any time.

---

4. Data Protection

4.1 Data at Rest

- **Encryption:**
  - All persistent storage (documents, templates, configs) encrypted at rest (AES-256).
  - Secrets and keys stored in secure vaults.
- **Database Security:**
  - If using SQL/NoSQL, enforce encryption, strong access controls, and regular backups.

- **Document Storage:**
    - Generated documents stored in isolated, access-controlled directories.

4.2 Data in Transit

- **TLS/SSL:**
    - All network communications (API, CLI, integrations) require HTTPS with strong TLS (v1.2+).
    - Secure WebSocket (WSS) for real-time collaboration.

4.3 Sensitive Data Handling

- **Input Validation:**
    - Strict validation (Joi, Zod, express-validator) for all API/CLI inputs.
- **Output Escaping:**
    - Prevent XSS by escaping all user-controlled outputs in web/admin UI.
- **Secrets Redaction:**
    - Never log or expose secrets/API keys in logs or error messages.

4.4 Data Retention and Disposal

- **Retention Policies:**
    - Configurable data retention for generated documents and logs.
- **Secure Deletion:**
    - Overwrite and delete sensitive files on removal.

---

5. Network Security

- **Perimeter Controls:**
    - Deploy behind firewalls and reverse proxies.

    - Use WAF (Web Application Firewall) for public endpoints.
- **IP Allowlisting:**
    - Restrict admin/API/CLI access to trusted networks.
- **Rate Limiting:**
    - `express-rate-limit` middleware to prevent brute-force and DoS attacks.
- **CORS:**
    - Restrictive CORS configuration, only allow trusted origins.
- **Helmet:**
    - Use `helmet` middleware for HTTP header hardening.

---

6. Security Controls

6.1 Application Controls

- **Input Sanitization:**
    - All data entering the system is sanitized and validated.
- **Output Encoding:**
    - All outputs to the UI or API responses are encoded to prevent injection attacks.
- **CSRF Protection:**
    - CSRF tokens for admin web interface and API where appropriate.
- **Session Management:**
    - Short-lived, signed, and encrypted tokens.
    - Session expiration and forced logout on sensitive changes.

6.2 Logging & Monitoring

- **Secure Logging:**
    - Use `winston` or similar for structured, tamper-resistant logs.
    - Logs include user, resource, action, timestamp, and outcome.
- **Monitoring:**
    - Health checks, usage metrics, and anomaly detection (API abuse, suspicious logins).

- **Alerting:**
  - Automated alerts for key events (failed logins, privilege escalations, integration failures).

6.3 Dependency & Code Security

- **Regular Dependency Scans:**
  - Use tools (npm audit, Snyk) to detect vulnerable packages.
- **Static Code Analysis:**
  - Enforce linting, type checking, and security rules in CI/CD.
- **Secure Coding Standards:**
  - Adhere to OWASP Top 10, secure design principles, and code reviews.

---

7. Threat Modeling

7.1 Key Threats and Mitigations

Threat	Mitigation
Credential Theft (API Keys, OAuth)	Secrets in vaults, not code; short-lived tokens; audit log access
Injection Attacks (SQL, Command, XSS)	Strict input validation, paramaterized queries, output encoding
Broken Authentication	Multi-factor auth, lockouts, rate limiting, JWT best practices
Excessive Permissions	RBAC, least privilege, integration scopes

Threat	Mitigation
Data Leakage (documents/templates)	Access controls, encryption, audit logging
DoS/Brute Force	Rate limiting, CAPTCHA for web, exponential backoff
Supply Chain Attacks	Dependency scanning, signed packages, minimal required dependencies
Insecure Integrations	OAuth2, permission scopes, regular integration reviews, revocation mechanisms
Insecure File Uploads	Size/type checks, antivirus scanning, upload to isolated storage
Man-in-the-Middle (MitM)	TLS everywhere, certificate pinning where possible
Insider Threats	Audit trails, role separation, regular permissions review
Session Hijacking	Secure cookies, token revocation, session expiration

---

8. Security Testing Strategy

8.1 Automated Testing

- **Unit Tests:**
    - Test all security-sensitive code paths (auth, RBAC, validation).

- **Integration Tests:**
    - Simulate real-world usage, including boundary and negative tests for authentication and authorization.
- **API Security Testing:**
    - Automated tools (OWASP ZAP, Postman, etc.) to test for injection, auth bypass, misconfigurations.

8.2 Manual Testing

- **Penetration Testing:**
    - Annual and pre-release pentests (internal or third-party).
- **Code Reviews:**
    - Peer reviews with security focus for all features.

8.3 Continuous Monitoring

- **Dependency Monitoring:**
    - Automated checks for vulnerabilities in dependencies.
- **Configuration Drift:**
    - Alerts for changes to critical security settings.

---

9. Incident Response Plan

9.1 Preparation

- **Contact Points:**
    - Security lead, DevOps, and legal representatives.
- **Runbooks:**
    - Documented procedures for common incidents (credential leak, data breach, DoS).

9.2 Detection

- **Anomaly Detection:**
    - Monitor for suspicious patterns (failed logins, data exfiltration).

- **Automated Alerts:**
  - Immediate alerting to security team for critical events.

9.3 Response

- **Containment:**
  - Revoke credentials, disable affected services, isolate compromised components.
- **Eradication:**
  - Remove malicious actors, patch vulnerabilities.
- **Recovery:**
  - Restore from clean backups, monitor for recurrence.
- **Notification:**
  - Inform affected users and regulators as required by law (e.g., GDPR breach notification timelines).

9.4 Lessons Learned

- **Postmortem:**
  - Document root cause, response effectiveness, and corrective actions.
- **Policy Updates:**
  - Incorporate learnings into policies and procedures.

---

10. Compliance Requirements

10.1 Data Privacy & Protection

- **GDPR:**
  - Data minimization, right to erasure, DPA agreements with processors.
- **SOX:**
  - Audit trails for document and process integrity.
- **PCI DSS:**

    - If handling payment data, follow PCI DSS segmentation and encryption requirements.

10.2 Security Standards

- **ISO 27001:**
    - Information security management system best practices.
- **NIST SP 800-53:**
    - Control families mapped to access control, audit, and incident response.

10.3 Sector-Specific

- **Healthcare (HIPAA):**
    - If handling PHI, ensure all administrative, physical, and technical safeguards are in place.
- **Financial (FINRA, Basel III, MiFID II):**
    - Maintain records, access controls, and auditability as per sector mandates.

10.4 Auditability

- **Comprehensive Logging:**
    - All sensitive and compliance-relevant actions are logged and retained as per policy.
- **Regular Reviews:**
    - Scheduled compliance and security reviews.

---

11. Security Monitoring

- **Health Endpoints:**
    - `/api/v1/health` and `/api/v1/health/ready` for liveness/readiness with restricted sensitive info.
- **Metrics & Analytics:**

    - Real-time usage monitoring, anomaly detection, and performance.
- **SIEM Integration:**
    - Support for forwarding logs and events to SIEM tools.
- **Admin Dashboard:**
    - Security alerts, audit trail review, and compliance status.

---

12. Security Patterns Used

- **Defense in Depth:**
    - Multiple layers of controls (validation, RBAC, encryption, monitoring).
- **Least Privilege:**
    - Users, services, and integrations only have permissions required for their purpose.
- **Fail Securely:**
    - Default to deny, explicit allow rules.
- **Secure Defaults:**
    - All services run with secure default configurations (no open admin endpoints, strong CORS, etc.).
- **Zero Trust:**
    - Assume breach, verify each request, and minimize trust boundaries.

This Security Design Document will be updated as the ADPA framework evolves to address emerging threats, compliance requirements, and feature expansions (e.g., mobile, real-time collaboration, advanced workflow automation).

Prepared by: Security Architecture Team
Date: [Current Date]