SignVerse

SignVerse Security Whitepaper

Last updated: July 4, 2026 · Document version 1.1

1. Introduction

SignVerse is an enterprise digital-signature platform: organizations prepare documents, route them to recipients for legally meaningful signature ceremonies, and retain tamper-evident records of what was signed, by whom, and how. Because a signing platform sits at the intersection of identity, legal evidence, and confidential business content, security is not a feature of SignVerse — it is the product's foundation.

This whitepaper describes the security architecture in plain language. It is derived directly from our maintained engineering documentation and verified against the implementation; where a capability is planned rather than shipped, it is listed in §10 (Roadmap) rather than described in the present tense.

2. Architecture and tenant isolation

- **Every record is owned by an organization.** Every query the platform executes is scoped to the requesting organization's identity — there is no code path that reads another tenant's envelopes, contacts, customers, templates, or audit history.
- **Cross-tenant references are rejected at the boundary.** Any identifier submitted in a request (a customer, a contact, a template) is validated as belonging to the requesting organization before it is used. These checks are covered by automated regression tests that run on every change.
- **Public links grant exactly one resource.** A signing link authorizes one recipient's ceremony on one document — it cannot be used to browse, enumerate, or access anything else.
- **Deployment flexibility.** SignVerse supports fully self-hosted, single-tenant deployment (your infrastructure, your database, your keys), giving organizations complete data-residency control.

3. Identity and authentication

3.1 Passwordless sign-in

SignVerse does not store user passwords for its own login. Sign-in is verified by a short-lived one-time code delivered to the user's email:

- Codes are generated with a cryptographically secure random generator and stored only as **Argon2 hashes** — never in plain text and never written to production logs.
- Codes expire in **10 minutes** and permit a maximum of **5 attempts**.
- Requests are rate-limited per email address *and* per network address, with automatic lockouts on repeated failures.
- Each code is **purpose-bound**: a code issued for login cannot be replayed to accept an invitation, change an email address, or enter the operator console.

Eliminating stored passwords removes the two most common causes of account takeover — credential stuffing and password-database breaches — entirely.

3.2 No account disclosure

Sign-in responses are deliberately identical whether or not an account exists for the entered address, and the platform does no account lookup before responding — so neither the response nor its timing reveals who is a SignVerse user. Account status is disclosed only to someone who proves control of the mailbox. This follows the OWASP Authentication and Forgot Password Cheat Sheet guidance and defends against user-enumeration reconnaissance.

3.3 Administrative separation

The operator console (platform administration) uses a separate, stricter sign-in path: access is verified against an explicit platform-administrator role before any code is issued, with tighter rate limits. Tenant users and platform operators are distinct identities with distinct privileges.

3.4 Machine identities

- **API keys** are shown once at creation and stored only as SHA-256 hashes; verification uses constant-time comparison, so key checks leak no timing information.
- **API client secrets** are hashed with HMAC-SHA256 using a server-side pepper, with pepper **versioning** so the pepper itself can be rotated without breaking existing integrations.
- **Development credentials cannot be used in production.** Development-only authentication shortcuts are rejected outright in production environments.

3.5 Misconfiguration refuses to run

A production deployment that is configured unsafely — a default or weak server secret, or token verification without issuer/audience validation — **fails at startup** with an explicit error rather than running in a weakened state.

4. Authorization

- **69 fine-grained permissions** across the platform's capability areas, assembled into **10 system roles** (from full Organization Admin down to read-only Viewer and Auditor).
- Every protected operation declares the permission it requires; authorization is enforced server-side on every request, never by hiding buttons in the UI.
- Machine access (API clients) is role-scoped exactly like human access.
- Every privileged action is recorded with the **true actor identity** — a human's email, an API client's identifier, never one disguised as the other.

5. Protecting data

5.1 In transit

All production traffic is served over TLS (HTTPS), with upgrade-insecure-requests enforced by policy, HTTP Strict Transport controls at the deployment edge, and secure, HttpOnly, SameSite session cookies.

5.2 Secrets and keys

- Integration secrets (mail credentials, webhook secrets, provider credentials, sealing keys) are encrypted at rest with authenticated symmetric encryption (Fernet).
- Production uses **envelope encryption**: a master key from the environment wraps versioned data-encryption keys, and key rotation is supported with a full, append-only rotation ledger — old data remains readable, new data uses the new key.
- **Bring-your-own-KMS:** organizations can hold secrets in their own AWS Secrets Manager, Azure Key Vault, GCP Secret Manager, or HashiCorp Vault; SignVerse stores only references, never the secret material, and fails closed if the vault is unreachable.
- Raw credentials are never persisted anywhere: API keys, client secrets, one-time codes, and signing tokens all exist in storage only as cryptographic hashes.

5.3 Documents

Documents are protected by tenant isolation, permission checks, capability-token controls (§6), and the tamper-evidence chain described in Signature Integrity. At-rest encryption of document files is provided today at the storage layer of the deployment (encrypted disks / server-side storage encryption), with application-level document encryption on the roadmap (§10).

6. Signing links and tokens

- **Hashed at rest** — every emailed link is a high-entropy (256-bit) capability token stored only as a hash; a database compromise does not yield usable signing links.
- **Lifecycle-tracked** — when a document is revised, old links are tombstoned: a stale link shows the recipient an honest "this link belongs to an older version" page, while an unknown token reveals nothing at all.
- **Expiring and revocable** — including download tokens for completed documents, which carry expiry *and* download-count limits.
- **QR handoff to mobile** uses a short-lived, two-step claim so that link scanners and previewers cannot consume a session; after the claim, the phone holds a secure cookie, not a URL token.

7. Application security

- **Uploads are treated as hostile.** PDFs are size-capped, signature-verified, parsed, and rejected if they contain active content (JavaScript, launch actions, auto-open actions) or embedded executables. Spreadsheets and CSVs are format-verified and parsed defensively. Attachments pass an extension blocklist plus magic-byte/MIME consistency checks. Logos and SVGs are stripped of any scriptable content. Optional **ClamAV malware scanning** can be enforced in fail-closed mode (uploads are refused if the scanner is unavailable).
- **Injection defenses.** All database access is through parameterized ORM queries. User-supplied rich text is sanitized with a modern allowlist sanitizer that removes scripts, event handlers, and dangerous URLs. The UI renders user data as text, never as markup.
- **Server-side request forgery (SSRF) prevention.** Document rendering validates every fetched resource, blocking local files, cloud metadata endpoints, and private network ranges — checked numerically, not by name.
- **Browser hardening.** Every response carries a strict Content-Security-Policy with **per-request script nonces**, clickjacking denial, MIME-sniffing protection, referrer and permissions policies, and no-store caching for application pages. State-changing browser requests are protected by SameSite cookies plus double-submit CSRF tokens.
- **Abuse resistance.** Layered rate limits apply per network address *and* per authenticated identity, with strict budgets on sensitive operations and clear Retry-After responses.

8. Auditability

SignVerse maintains an **append-only audit trail** — audit records are never updated or deleted by application code. It covers envelope lifecycle, every step of each signing ceremony, authentication ceremonies (without ever recording the codes themselves), sealing, forensic quarantine decisions, key rotation, and administrative changes. Personal data inside audit details is masked, and secrets never appear in logs.

9. Secure development lifecycle

- Every change is developed against a mandatory **17-control guardrail catalog** (OWASP Top-10 baseline, brute-force protection, injection, XSS, CSRF, tenancy, secrets, PII, cryptography standards, public-link safety, uploads, headers, logging, supply chain).
- Every feature ships with a **completed security review and a security test report** — no review, no merge.
- Security-critical infrastructure (encryption, validation, sanitization, rate limiting) exists once and is reused; parallel unreviewed implementations are prohibited by policy.
- Cross-tenant isolation, enumeration resistance, and injection defenses are enforced by automated tests that run on every change.

10. Transparency and roadmap

Planned enhancement	Status
Database-level row security (defense-in-depth beneath the application's tenant isolation)	Shipped
External KMS as the default home for all platform secrets (BYO-KMS available today)	Shipped
Application-level encryption of stored document files (storage-layer encryption applies today)	Shipped
Embedded PAdES-LTV / PDF-A-2b seals (detached cryptographic seals ship today)	Shipped
Per-tenant rate-limit budgets (per-IP and per-identity limits apply today)	Shipped
Explicit origin validation on mobile signing ceremonies (SameSite protections apply today)	Shipped

11. Shared responsibility

SignVerse secures the platform; customers keep their side strong by protecting the mailboxes used for sign-in and signing invitations, safeguarding API keys and client secrets (they are shown once), configuring their SMTP/communication providers with least-privilege credentials, using the role catalog to grant users only what they need, and — in self-hosted deployments — maintaining TLS, disk encryption, and patching on the hosting infrastructure.

Questions, disclosures, or audit requests: contact your SignVerse representative. Engineering provenance for every statement in this document is maintained in the SignVerse security design records.