PUBLIC

Security Whitepaper

Technical Security Architecture & Compliance Overview

SURVEYS PEOPLE FINISH

A comprehensive examination of the security controls, encryption standards, access management, compliance frameworks, and operational practices that protect customer data across every layer of the Claform infrastructure.

VERSION	PUBLICATION DATE	CLASSIFICATION
2.1	**April 2026**	**PUBLIC**

PREPARED BY	APPROVED BY	REVIEW CYCLE
Security Engineering Team	**Chief Technology Officer**	**Quarterly**

REVISION HISTORY

v2.1	April 2026	Added Mobile SDK security, Staff Admin security, enhanced compliance coverage
v2.0	January 2026	Major update: omnichannel distribution security, SAML SSO, security monitoring
v1.0	August 2025	Initial publication covering core platform security architecture

CONFIDENTIALITY NOTICE

This document is classified as PUBLIC and may be shared freely with prospective customers, partners, and auditors evaluating Claform's security posture. The information herein describes the security architecture of the production system as of the publication date. Security controls are subject to continuous improvement and may be enhanced beyond what is described in this version.

For questions regarding the contents of this document, or to request a detailed security review, please contact security@claform.com.

Claform
claform.com/security

About This Document

Purpose

This whitepaper provides a thorough, transparent description of the security controls that protect data within the Claform ecosystem. It is designed to answer the questions that security teams, compliance officers, and procurement reviewers ask when evaluating a new vendor: How is our data encrypted? Who can access it? What happens if something goes wrong? How do you meet regulatory requirements? Every answer in this document is grounded in the actual production system, not aspirational roadmap items.

Intended Audience

This document is written for multiple audiences. Technical readers such as security engineers and architects will find specific algorithms, key sizes, and protocol configurations. Business readers such as compliance officers and IT directors will find coverage of regulatory frameworks, audit capabilities, and data handling policies. We have structured the content so that each section opens with a plain-language summary before diving into technical specifics.

Document Scope

This whitepaper covers the full security surface of Claform: infrastructure and deployment security, application-layer protections, data encryption and key management, authentication and identity management, role-based access control, API security, security monitoring and audit capabilities, regulatory compliance, mobile SDK security, staff administration security, and incident response procedures. It does not cover feature functionality, pricing, or integrations unless those topics have a direct security implication.

How to Read This Document

Sections are organized by security domain and are designed to be read independently. If you are completing a vendor security questionnaire, the Table of Contents will help you locate the relevant section for each question. Key terms are introduced in context rather than in a separate glossary, so that each section is self-contained.

Document Conventions

- Tables present specific configuration values and standards references.
- Bullet lists describe multiple related controls or parameters.
- Highlighted callout boxes contain important notes or contact information.
- Industry standards (e.g., NIST SP 800-132, OWASP Top 10) are cited where our controls align with established benchmarks.
- All technical specifications reflect the production system at the time of publication.

> *Questions about the contents of this document? Contact security@claform.com. To schedule a live security architecture walkthrough with our engineering team, reach out to sales@claform.com.*

Executive Summary

This page provides a one-page overview for executive stakeholders. Detailed technical specifications follow in the body

WHAT WE PROTECT

- Survey responses across 8 channels
- 43 question types with validated input
- Personal data encrypted with AES-256-GCM
- 68 countries with local compliance
- 44 third-party integration credentials
- Subscription billing via Stripe (PCI DSS L1)

HOW WE PROTECT IT

- AES-256-GCM encryption at rest (PBKDF2, 100K iterations)
- TLS 1.2/1.3 in transit (ECDHE forward secrecy)
- bcrypt-12 password hashing (~250ms/hash)
- TOTP MFA with 10 backup codes
- SAML 2.0 SSO with DNS domain verification
- 8 roles, 42 granular permissions (RBAC)
- Dual-layer rate limiting (Nginx + application)
- 43 audited action types with severity classification
- 20 automated security alert types
- 90-day automatic key rotation

PLATFORM SCALE

40	backend modules
119	API controllers
178	service classes
131	database entities
92	frontend hooks
312	automated tests
6	supported languages
5	mobile SDK platforms
24	database migrations
6	CI/CD workflows

COMPLIANCE

- GDPR (EU) — Data subject rights, DPA, consent management
- TCPA (US) — Opt-out management, quiet hours, consent tracking
- CASL (Canada) — Anti-spam compliance for campaigns
- LGPD (Brazil) — Data protection, consent expiry
- OWASP Top 10 (2021) — Full coverage, mapped in Section 10
- NIST SP 800-132 — Key derivation standard alignment
- SOC 2 Type II — Annual third-party audit (NDA)
- HIPAA BAA — Available for healthcare organizations

Security: security@claform.com | Sales: sales@claform.com | Web: claform.com/security

Table of Contents

Each section covers an independent security domain. Use this index to navigate directly to the topic relevant to your evaluation.

1. Executive Summary

What Is Claform

This whitepaper covers Claform, the survey and feedback product operated by Clarock Inc. References to "Claform" throughout this document refer to the product and its infrastructure; the legal entity responsible for the platform is Clarock Inc.

Claform is an enterprise-grade survey, feedback, and data collection platform. It enables organizations to build sophisticated surveys with 43 question types, distribute them through eight distinct channels, and analyze responses with built-in statistical tools, AI-powered insights, and customizable dashboards. The platform is used by organizations of all sizes, from startups running their first customer satisfaction survey to regulated enterprises collecting patient health data, employee engagement scores, and citizen feedback.

The eight distribution channels are email, SMS, WhatsApp, voice and IVR (interactive voice response), web link sharing, in-person kiosk, printed paper with QR codes, and workplace messaging through Slack and Microsoft Teams. Each channel carries its own security and compliance requirements. SMS and voice campaigns must comply with telecommunications regulations such as TCPA and CASL. WhatsApp distribution requires explicit consent management under GDPR. Kiosk mode must protect data on shared physical devices. The platform addresses all of these requirements natively, rather than relying on customers to implement their own compliance controls.

Why Security Matters in Survey Platforms

Survey platforms occupy a unique position in the data landscape. Unlike most SaaS tools that handle structured, predictable data types, a survey platform can collect virtually anything: names, medical conditions, salary information, political opinions, employee grievances, or product feedback that contains trade secrets. The platform operator cannot predict what data a customer will collect, which means the security architecture must protect all data as if it were sensitive by default.

Additionally, survey platforms interact with external respondents who may not have accounts, cannot be required to use multi-factor authentication, and may access surveys from untrusted networks and devices. This creates a fundamentally different threat model from internal business tools. Our security controls are designed to protect data integrity and confidentiality even when the respondent endpoint is outside our control.

Our Security Philosophy

Security at Claform is not a feature — it is an architectural principle. This distinction matters because features can be toggled off, deferred, or deprioritized. An architectural principle shapes every design decision, from the database schema to the HTTP response headers. Three core beliefs guide our approach:

- Defense in depth: No single control is trusted to be sufficient. Encryption protects data even if access controls fail. Rate limiting protects the system even if authentication is bypassed. Audit logging records events even if monitoring alerts are missed. Every layer operates independently.
- Secure by default: Security controls are enabled out of the box, not opt-in. Passwords are hashed with bcrypt-12 automatically via database lifecycle hooks — a developer cannot accidentally store plaintext. Access tokens expire in 15 minutes regardless of configuration. Rate limits are enforced on every endpoint. Customers cannot weaken these defaults.
- Least privilege everywhere: Users receive the minimum permissions required for their role. API keys are scoped to specific operations and specific teams. Database queries are filtered by team ownership at the service layer. Staff administrators have different clearance levels with different token lifetimes. No actor in the system has more access than they need.

Platform Scale

Understanding the scale of the platform provides context for why each security control exists. The numbers below describe the production system architecture:

Metric	Value	Security Implication
Backend modules	40 NestJS modules	Each module has isolated guards and validation
API controllers	119 controllers	Every endpoint is rate-limited and authorized individually
Database entities	131 TypeORM entities	All queries parameterized, team-scoped, indexed
Frontend hooks	92 React Query hooks	Server state isolated from client state
Translation keys	33,179 across 6 locales	Localized error messages prevent information leakage
Mobile SDK platforms	5 (Core, RN, Flutter, iOS, Android)	Each implements encrypted storage natively
Third-party integrations	44 (Salesforce, Slack, HubSpot...)	All credentials encrypted with AES-256-CBC
Distribution channels	8 (email, SMS, WhatsApp, voice...)	Per-channel compliance enforcement
Question types	31 types across all SDKs	Input validation per type, no arbitrary code execution
Database migrations	24 tracked migrations	Schema changes are versioned and reversible

Security Controls at a Glance

The following table summarizes the primary security controls covered in this whitepaper. Each row includes the specific implementation used in production and the industry standard it aligns with. Subsequent sections provide detailed descriptions of each control.

Control Domain	Implementation	Standard Reference
Encryption at rest	AES-256-GCM, PBKDF2-SHA512 key derivation (100K iterations)	NIST SP 800-132
Encryption in transit	TLS 1.2/1.3, ECDHE forward-secret cipher suites only	NIST SP 800-52 Rev. 2
Password hashing	bcrypt with cost factor 12 (~250ms per hash)	OWASP Password Storage
Multi-factor authentication	TOTP (RFC 6238), 10 backup codes, 30-day trusted devices	NIST SP 800-63B
Session management	15-min access token, 24h/30d refresh, rotation on use	OWASP Session Mgmt.
Access control	8 hierarchical roles, 42 permissions, ownership enforcement	NIST AC-3
Rate limiting	Dual-layer (Nginx + app), per-IP/user/key, progressive lockout	OWASP API Security Top 10
Audit trail	43 tracked actions, 4 severity levels, sensitive field redaction	SOC 2 CC7.2
Key management	90-day automatic rotation, 3 prior versions retained	PCI DSS Req. 3.6

Enterprise SSO	SAML 2.0 with DNS-based domain verification	SAML 2.0 Core Spec
Input validation	8 specialized validation pipes, recursive sanitization	OWASP Input Validation
Security monitoring	20 alert types, automated threshold detection	NIST SP 800-137
Webhook security	HMAC-SHA256 payload signing, circuit breaker pattern	IETF HTTP Message Signatures
Mobile SDK security	Encrypted storage, biometric gates, event isolation	OWASP Mobile Top 10

2. Platform Architecture

Understanding the overall architecture is essential to evaluating security, because security controls are only as strong as the system they protect. This section describes the components that make up Claform, how they communicate, how data flows between them, and how the system is deployed. Each component has specific security responsibilities, and together they form the foundation that every subsequent section of this whitepaper builds upon.

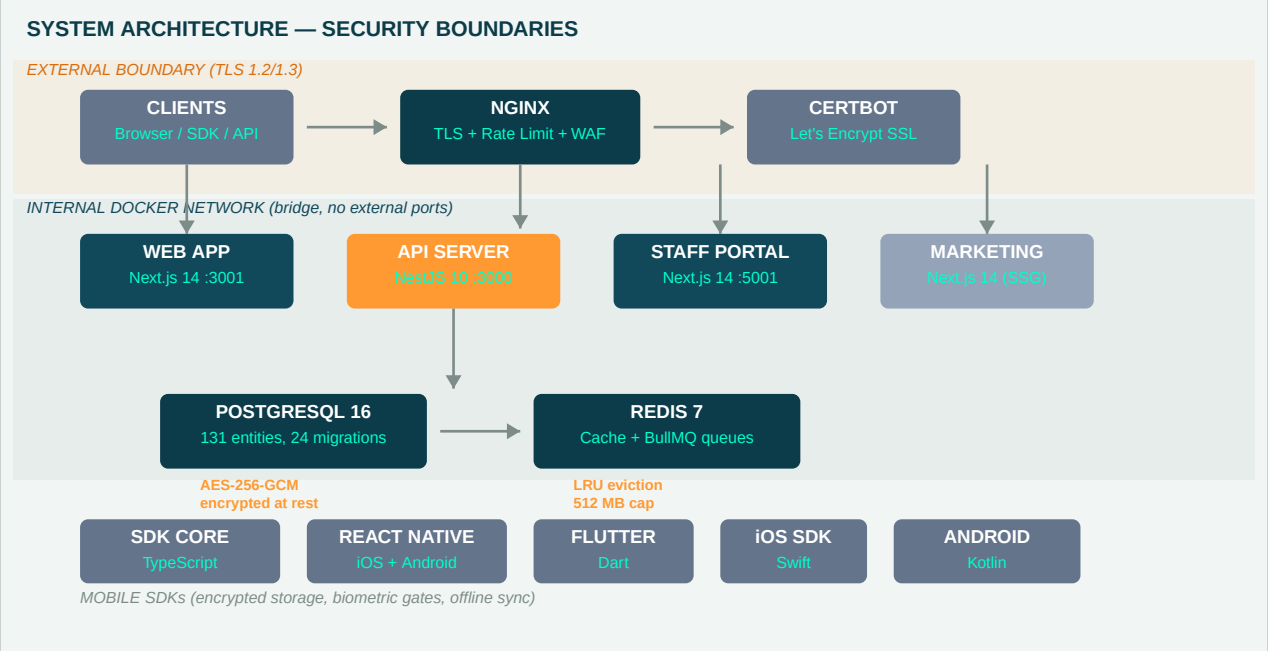
System Components

Claform is composed of four web applications, a centralized API layer, two data stores, and five mobile SDK packages. All source code is managed within a single Turborepo monorepo with strict workspace isolation, meaning that each application has its own dependencies, build process, and deployment configuration. A change to the marketing site cannot inadvertently affect the API server.

Component	Technology	Role in the System
API Server	NestJS 10 on Node.js 20	Central business logic layer. 40 feature modules, 119 REST controllers, 178 services. Handles authentication, authorization, data access, queue dispatch, and webhook delivery.
Web Application	Next.js 14, React 18	Primary customer-facing interface. Survey builder, analytics dashboards, campaign management, team administration. Runs on port 3001.
Marketing Site	Next.js 14 (SSG)	72 statically generated pages across 4 locales. Public-facing: pricing, documentation, this whitepaper. No access to customer data.
Staff Portal	Next.js 14	Internal administration console for platform operations. Elevated security controls, hidden route configuration, separate authentication. Runs on port 5001.
PostgreSQL 16	Alpine container	Primary data store. 131 entities, parameterized queries only, 50-connection pool, 30-second statement timeout, automated daily backups.
Redis 7	Alpine container	In-memory data layer for session state, rate limit counters, workflow cache, and BullMQ job queue broker. 512 MB memory cap with LRU eviction.
Mobile SDKs	TypeScript, RN, Flutter, Swift, Kotlin	Five SDK packages enabling native survey rendering on iOS, Android, React Native, and Flutter. Shared core engine with platform-specific UI.

Architecture Diagram

The diagram below illustrates the system architecture and the security boundaries between components. All external traffic enters through a single point (Nginx), which terminates TLS and enforces rate limits before forwarding to internal services. The internal Docker network is not externally accessible.



How Data Flows Through the System

Every piece of data in the platform follows a predictable path, and understanding this path clarifies where each security control takes effect. Here is the typical lifecycle of a survey response:

A respondent opens a survey link in their browser or mobile app. The request is received by Nginx, which terminates TLS, applies rate limiting, adds security headers, and forwards the request to the appropriate upstream service. If the survey is accessed through the web application, Next.js renders the survey form. If accessed through a mobile SDK, the SDK fetches the survey definition from the API and renders it natively.

When the respondent submits their answers, the response payload travels from the client to Nginx, then to the API server. The API server processes the request through its full security pipeline: sanitization, validation, authentication (if required), and authorization. The validated response data is written to PostgreSQL with any sensitive fields encrypted at the application layer before reaching the database. If the survey has workflows configured, the response submission triggers an asynchronous job via BullMQ and Redis, which processes notifications, webhook deliveries, or integration syncs independently of the original request.

At no point does the respondent's browser communicate directly with the database or the Redis cache. All data access is mediated by the API server, which enforces team-scoped queries, permission checks, and audit logging on every operation.

Multi-Tenancy and Data Isolation

Claform serves multiple organizations from a shared infrastructure, which makes tenant isolation a critical security requirement. We use a shared-database, logically-isolated tenancy model: all organizations store data in the same PostgreSQL instance, but every record includes a team identifier foreign key that establishes ownership.

Data isolation is enforced at three layers to prevent any single failure from exposing cross-tenant data:

- Service layer: Every database query includes a team scope filter. The NestJS service methods require a team ID parameter, and the query builders add WHERE clauses automatically. A query that omits the team filter returns zero results — not results from all teams.

- Guard layer: Before a request reaches the service layer, the RBAC guard verifies that the authenticated user is a member of the team that owns the requested resource. A valid JWT token from User A on Team X cannot access surveys belonging to Team Y, even if User A somehow constructs a request with Team Y's survey ID.
- API key layer: API keys are scoped to the team that created them. The key's permission scopes (such as SURVEY_READ or RESPONSE_EXPORT) only apply to resources owned by that team. A key with FULL_ACCESS on Team X has zero access to Team Y's data.

This three-layer enforcement means that tenant isolation does not depend on any single mechanism. If a developer writes a service method that accidentally omits the team filter, the guard layer will still block cross-tenant access. If the guard has a bug, the service-layer query will still return only team-scoped results. Defense in depth applies to data isolation just as it applies to encryption and authentication.

Deployment Architecture

The platform is deployed using Docker Compose with a multi-container architecture. Each service runs in its own container with defined resource limits, health checks, and restart policies. The production deployment includes the following containers:

Container	Image	Resources	Health Check
API Server	Node 20 Alpine (multi-stage build)	1 GB max / 512 MB reserved	HTTP GET /api/v1/health every 15s
Web Application	Node 20 Alpine	512 MB max / 256 MB reserved	HTTP readiness probe
Staff Portal	Node 20 Alpine	512 MB max / 256 MB reserved	HTTP readiness probe
Marketing Site	Node 20 Alpine	512 MB max / 256 MB reserved	HTTP readiness probe
PostgreSQL	PostgreSQL 16 Alpine	2 GB max / 512 MB reserved	pg_isready every 10s
Redis	Redis 7 Alpine	512 MB max (LRU eviction)	redis-cli ping every 10s
Nginx	Nginx Alpine	Default	Process monitor
Certbot	Certbot	On-demand	Certificate renewal cron

All containers communicate over an internal Docker bridge network. PostgreSQL and Redis do not expose ports to the host machine — they are only accessible from other containers within the bridge network. The only externally accessible entry point is Nginx, which terminates TLS and routes requests to the appropriate upstream service based on the domain and path.

Production Docker images are built using multi-stage Dockerfiles. The first stage installs dependencies and compiles TypeScript. The second stage copies only the compiled JavaScript output into a clean, minimal Alpine image with no build tools, development dependencies, or source code. The resulting image is smaller, has a reduced attack surface, and runs under a non-root user account (nestjs:nodejs) to limit the impact of a potential container escape.

Asynchronous Job Processing

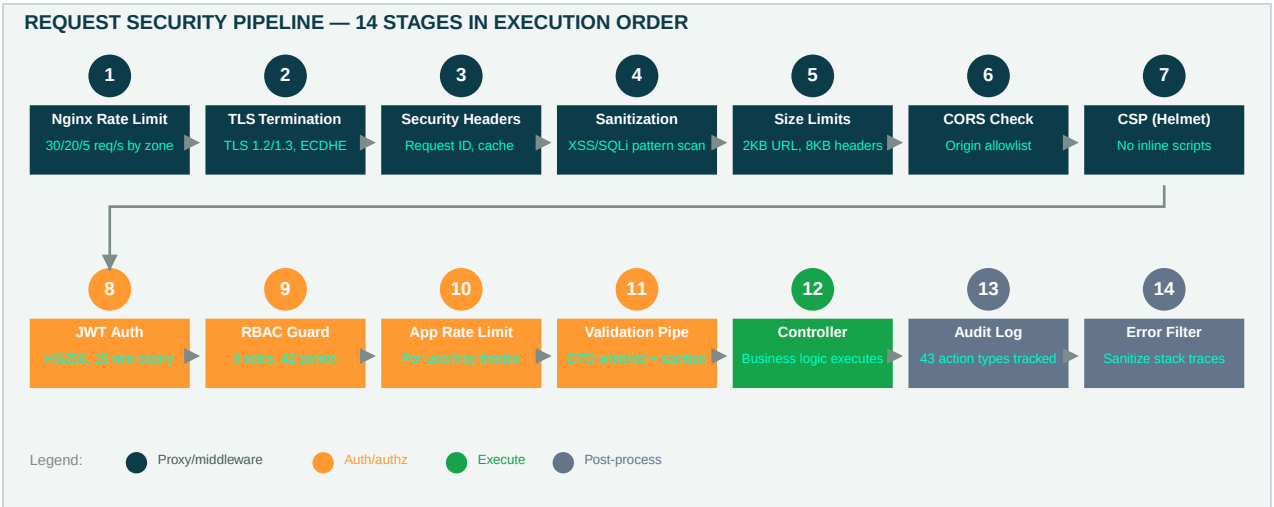
Many operations in the platform are asynchronous: sending campaign emails, delivering SMS messages, executing workflow steps, generating reports, and processing webhook deliveries. These tasks are handled by BullMQ, a Redis-backed job queue that provides reliable, at-least-once delivery with configurable retry logic.

Queue	Purpose	Retry Policy
email-send	Campaign and transactional email delivery via SES, SendGrid, or SMTP	3 attempts, exponential backoff
sms-send	SMS message delivery via Twilio with compliance pre-checks	3 attempts, exponential backoff
whatsapp-send	WhatsApp message delivery with consent verification	3 attempts, exponential backoff
voice-call	Voice call initiation via Twilio with TwiML generation	3 attempts, exponential backoff
workflow-exec	Workflow step execution (notifications, integrations, conditions)	3 attempts, exponential backoff
report-gen	Scheduled report generation in PDF and Excel formats	2 attempts, fixed delay

Job processors run within the API server process. Each processor checks campaign pause state via a Redis O(1) lookup before processing each job, allowing campaigns to be paused instantly without waiting for in-flight jobs to drain. Failed jobs are classified as transient (retry) or permanent (dead-letter) based on the error type, preventing infinite retry loops on non-recoverable errors such as invalid phone numbers or revoked API credentials.

Request Lifecycle

Every inbound HTTP request passes through a multi-stage security pipeline before reaching the application logic. The pipeline is designed so that cheap, fast checks run first, rejecting obviously invalid requests before more expensive operations like database lookups or cryptographic verification are performed. This ordering is deliberate: it protects the system from resource exhaustion attacks by rejecting malicious traffic as early as possible.



The complete pipeline, in execution order:

- Stage 1 — Nginx rate limiting: IP-based rate zones reject flood traffic before it reaches the application. Three zones are defined: general (30 requests/second), API (20 requests/second), and authentication (5 requests/second). Requests exceeding these limits receive a 429 response directly from Nginx.
- Stage 2 — TLS termination: Nginx decrypts the TLS connection using the server's private key. Only TLS 1.2 and 1.3 are accepted. ECDHE cipher suites ensure forward secrecy. OCSP stapling is enabled to avoid the latency of real-time certificate status checks.

- Stage 3 — Security headers: A custom middleware assigns a unique request ID (UUID v4) to every request, sets cache-control headers to no-store on sensitive endpoints, and strips the Server header to prevent technology fingerprinting.
- Stage 4 — Request sanitization: The RequestSanitizationMiddleware scans the request body, query parameters, and headers for patterns associated with cross-site scripting, SQL injection, and protocol-level attacks. Dangerous content triggers a 400 rejection with an audit log entry.
- Stage 5 — Request size limits: URL length is capped at 2,048 characters (8,192 for OAuth callbacks). Header size is capped at 8,192 bytes. These limits prevent buffer overflow attacks and header-based denial-of-service.
- Stage 6 — CORS validation: The origin header is checked against a strict allowlist. Requests from unknown origins are rejected. Credentials (cookies and authorization headers) are enabled for trusted origins only.
- Stage 7 — Content Security Policy: Helmet enforces a restrictive CSP: no inline scripts, no iframes, no object embeds, and script sources limited to the application origin. This prevents injected scripts from executing even if they bypass input sanitization.
- Stage 8 — JWT authentication: The JwtAuthGuard extracts the Bearer token, verifies its signature (HS256), checks expiry, and validates the issuer and audience claims. Invalid or expired tokens are rejected with a 401 response.
- Stage 9 — RBAC authorization: The RbacGuard checks the authenticated user's role against the permissions required by the endpoint. Role inheritance is evaluated, ownership is verified where applicable, and denials are logged to the audit trail.
- Stage 10 — Application rate limiting: Per-endpoint throttle decorators apply identity-aware rate limits. For example, the login endpoint allows 5 attempts per 15 minutes per IP+email combination. API key endpoints allow 1,000 requests per minute per key.
- Stage 11 — Input validation: The SecureValidationPipe transforms the raw request body into a typed DTO (Data Transfer Object), stripping unknown properties, enforcing type constraints, and applying recursive sanitization to all string values.
- Stage 12 — Controller execution: The validated, authenticated, authorized request reaches the controller method, which delegates to the service layer for business logic.
- Stage 13 — Audit logging: The AuditInterceptor records the request method, endpoint, user, response status, and duration for sensitive operations (authentication, data mutations, exports).
- Stage 14 — Error sanitization: The SecurityExceptionFilter catches any unhandled exceptions, removes internal details (SQL errors, stack traces, file paths), and returns a safe, generic error message to the client. The original error is logged server-side.

3. Infrastructure Security

Container Architecture

The platform runs in Docker containers orchestrated by Docker Compose. Production containers are built using multi-stage Dockerfiles: a build stage compiles TypeScript and bundles assets, a production stage copies only the compiled output into a minimal Node.js 20 Alpine image. Containers run as a non-root user (nestjs:nodejs) to limit the blast radius of a container escape.

Resource limits are enforced at the container level. The API service is capped at 1 GB memory with 512 MB reserved. Frontend services are capped at 512 MB with 256 MB reserved. PostgreSQL receives 2 GB maximum with 512 MB reserved. These limits prevent a memory leak or denial-of-service condition in one service from starving others.

Reverse Proxy & TLS

Nginx serves as the reverse proxy and TLS termination point. The TLS configuration follows Mozilla's "Modern" compatibility profile:

Parameter	Value
Protocols	TLSv1.2, TLSv1.3
Cipher suites	ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES256-GCM-SHA384, ECDHE-RSA-AES256-GCM-SHA384
Session cache	Shared, 10 MB
Session tickets	Disabled (prevents forward secrecy bypass)
OCSP stapling	Enabled with 1.1.1.1 / 8.8.8.8 resolvers
HSTS	max-age=31536000; includeSubDomains; preload
SSL certificates	Let's Encrypt with Certbot auto-renewal

Static assets served through Next.js receive a Cache-Control header of "public, immutable" with a 365-day max-age. Dynamic API responses receive no-cache directives on sensitive endpoints. Gzip compression is enabled at level 6 for text, JSON, JavaScript, CSS, and SVG content types.

Database Security

PostgreSQL 16 is deployed in Alpine containers with the following hardening measures:

- Connection pooling: 50 maximum connections, 5 minimum idle, preventing connection exhaustion attacks
- Statement timeout: 30 seconds, preventing long-running queries from locking tables or consuming resources
- Shared buffers: 256 MB with 768 MB effective cache size for optimal query planning
- Slow query logging: Queries exceeding 1,000 ms are logged for performance and anomaly detection
- Parameterized queries only: TypeORM's query builder generates parameterized SQL, eliminating SQL injection
- Production indexes: A dedicated migration adds composite indexes on high-frequency query patterns
- Automated daily backups via pg_dump with gzip compression and 7-day rolling retention

Redis Security

Redis 7 is deployed on an internal Docker bridge network with no port exposure to the host. It uses LRU eviction with a 512 MB memory cap. Rate limit counters use atomic Lua scripts (INCR + EXPIRE in a single operation) to prevent race conditions between increment and expiry. Campaign pause state lookups are O(1) operations, ensuring that high-volume queue processors can check pause state without blocking.

4. Application Security

Input Validation

All user input passes through a multi-layer validation pipeline before it reaches business logic. The platform uses NestJS's class-validator and class-transformer libraries with a custom SecureValidationPipe that enforces strict whitelisting — any properties not explicitly declared in the DTO are silently stripped, and optionally rejected with a 400 error.

The validation pipeline includes eight specialized pipes:

- SecureValidationPipe: Class-validator integration with automatic whitelist, sanitization before validation, and recursive nested object validation
- UUID Validation Pipe: RFC 4122 v4 format enforcement for all resource identifiers
- Integer Validation Pipe: Range-constrained numeric input with configurable min/max bounds
- String Sanitization Pipe: HTML stripping, entity encoding, max-length enforcement, whitespace normalization
- File Validation Pipe: MIME type allowlist, 5 MB default limit, double-extension detection to prevent polyglot attacks
- Array Validation Pipe: Maximum 100 items, per-item type validation (string, number, UUID), uniqueness enforcement
- Boolean Validation Pipe: Case-insensitive parsing accepting true/1/yes/on
- Enum Validation Pipe: Strict enum membership checking

Cross-Site Scripting Prevention

XSS prevention operates at three layers. First, the RequestSanitizationMiddleware scans all request bodies, query parameters, and headers for dangerous patterns. Second, a dedicated sanitization utility performs recursive HTML entity encoding on all string values before storage. Third, the Content Security Policy header enforced by Helmet blocks inline script execution.

The middleware detects 13 specific attack patterns via regular expressions. Matched content triggers a 400 rejection with an audit log entry recording the IP, user agent, and matched pattern:

- Script tags with attributes and closing script tags
- JavaScript protocol handlers (javascript:)
- HTML event handlers with and without quoted values (onclick=, onerror=, onload=)
- CSS expression() functions (used for script execution in legacy browsers)
- VBScript protocol handlers
- Data URIs containing text/html or SVG payloads (data:text/html, data:image/svg+xml)
- Iframe, object, and embed tags (DOM injection vectors)
- SVG elements with onload handlers

Additionally, a SQL injection detection pattern identifies statement chaining (;--), boolean injection (OR/AND with comparison operators), UNION SELECT, destructive DDL (DROP/ALTER/TRUNCATE), and execution commands (EXEC). Matched input is blocked before reaching the query layer, providing defense independent of TypeORM's parameterized queries.

Security Headers

The platform enforces a comprehensive set of HTTP security headers through both Nginx and application-level middleware. These headers are applied to every response:

Header	Value	Purpose
Content-Security-Policy	default-src 'self'; script-src 'self'; frame-src 'none'	Prevents XSS and clickjacking
Strict-Transport-Security	max-age=31536000; includeSubDomains; preload	Forces HTTPS for 1 year
X-Frame-Options	DENY	Prevents framing by any origin
X-Content-Type-Options	nosniff	Prevents MIME type sniffing
Referrer-Policy	strict-origin-when-cross-origin	Limits referrer information leakage
Permissions-Policy	camera=(), microphone=(), geolocation=()	Disables sensitive browser APIs
Cross-Origin-Embedder-Policy	require-corp	Isolates cross-origin resources
X-Permitted-Cross-Domain-Policies	none	Blocks Flash/Acrobat cross-domain requests

Error Handling & Information Leakage

A global SecurityExceptionFilter intercepts all unhandled exceptions before they leave the application boundary. The filter applies pattern-matching to detect internal error messages that reference database queries, SQL syntax, connection strings, stack traces, or Node.js module paths. Matched messages are replaced with safe, generic alternatives mapped to the HTTP status code (e.g., 500 becomes "Internal server error"). Stack traces are never included in production responses. The original error is logged server-side with full context for debugging.

HTTP Status	Safe Message Returned to Client	Original Error Preserved
400	Invalid request. Please check your input and try again.	Logged with validation details
401	Authentication required. Please sign in.	Logged with token state
403	Access denied. You do not have permission for this action.	Logged with role and permission context
404	The requested resource was not found.	Logged with requested path
429	Too many requests. Please try again later.	Logged with rate limit zone and counter
500	An internal error occurred. Please try again later.	Full stack trace logged server-side
503	Service temporarily unavailable. Please try again shortly.	Logged with service health context

Dependency Management & Supply Chain Security

The platform's dependency supply chain is secured through multiple controls. All package installations use lock files (package-lock.json) that pin exact dependency versions and include integrity hashes. Production Docker images are built with npm ci, which fails if the lock file is out of sync with package.json, preventing undeclared dependency changes from entering the build.

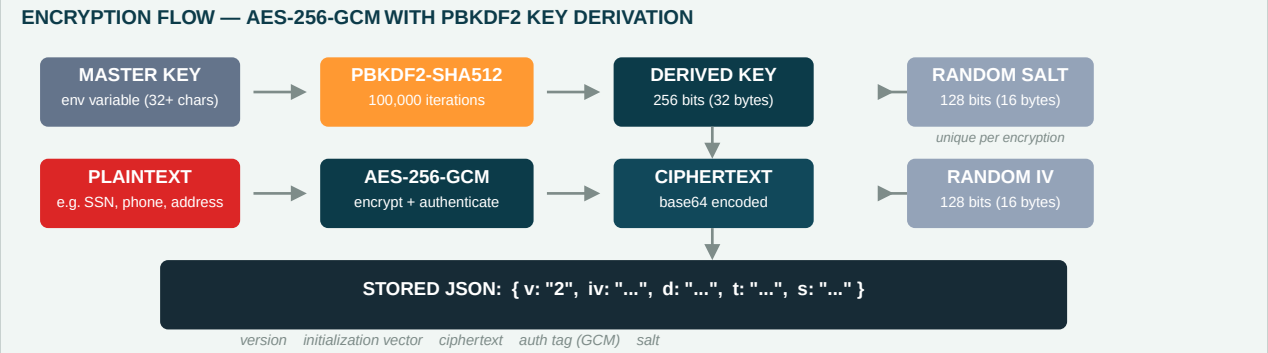
- GitHub Dependabot monitors all workspace packages for known vulnerabilities and creates automated pull requests for security patches

- npm audit is run as part of the CI pipeline; builds fail on critical or high-severity vulnerabilities
- Production images are built from pinned base images (node:20-alpine) with explicit version tags, not :latest
- Development dependencies (test frameworks, linters, build tools) are excluded from production images via multi-stage builds
- No eval(), Function(), or dynamic code execution patterns are used anywhere in the codebase

5. Data Protection & Encryption

Encryption at Rest

Sensitive fields are encrypted at the application layer before being written to the database. This provides a security boundary independent of database-level encryption: even with full database access, encrypted fields remain protected without the application encryption key.



Algorithm & Parameters

Parameter	Value	Reference
Algorithm	AES-256-GCM	NIST SP 800-38D
Key length	256 bits (32 bytes)	NIST SP 800-131A
IV length	128 bits (16 bytes), random per encryption	NIST SP 800-38D Section 8.2
Authentication tag	128 bits (16 bytes)	GCM specification
Salt length	128 bits (16 bytes), random per derivation	NIST SP 800-132
Key derivation	PBKDF2 with SHA-512, 100,000 iterations	NIST SP 800-132
Master key source	ENCRYPTION_MASTER_KEY env var (min 32 chars)	Configuration management

The choice of GCM mode provides both confidentiality and authenticity. The authentication tag ensures that tampering with the ciphertext is detected at decryption time, preventing ciphertext manipulation attacks that are possible with modes like CBC.

Encrypted Fields

The following entity fields are encrypted at rest:

- User: social security number, date of birth, phone number, physical address
- Survey: internal notes (may contain PII referenced by survey creators)
- Response: personalData field (freeform respondent-provided personal information)
- API Key: TOTP secret (used for MFA recovery)

Key Rotation

Encryption keys are rotated on a 90-day cycle. The system retains the three most recent prior key versions. Each encrypted value stores a version identifier (v field) alongside the ciphertext, allowing the decryption routine to select the correct key version. This design enables zero-downtime key rotation — new writes use the current key, while reads transparently handle values encrypted with prior versions.

Encrypted Data Structure

Each encrypted value is stored as a JSON object containing all the information needed to decrypt it, except the master key itself. This self-describing format enables transparent key rotation and eliminates the need for a separate key-version lookup table:

Field	Content	Purpose
v	Key version identifier (e.g., "2")	Selects correct key version for decryption
iv	Initialization vector (base64, 16 bytes)	Unique per encryption, prevents pattern analysis
d	Ciphertext (base64)	The encrypted data payload
t	Authentication tag (base64, 16 bytes)	GCM integrity verification — detects tampering
s	Salt (base64, 16 bytes)	Unique per derivation, hardens the key derivation

Encryption in Transit

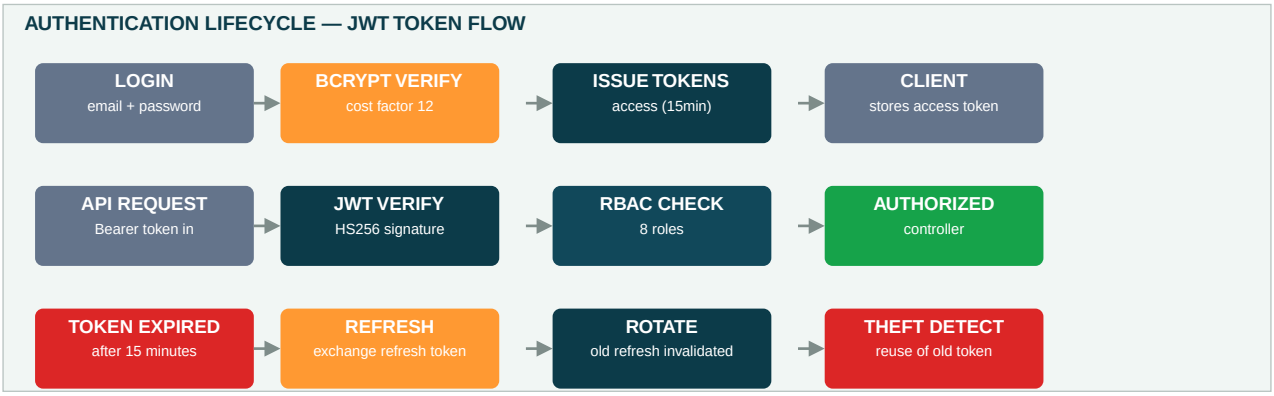
All client-to-server communication is encrypted using TLS 1.2 or 1.3 with forward-secret ECDHE key exchange. The Nginx configuration explicitly disables older protocols (SSLv3, TLS 1.0, TLS 1.1) and weak cipher suites. HSTS with a one-year max-age and preload flag prevents protocol downgrade attacks. Internal service-to-service communication within the Docker network uses unencrypted connections, as the network boundary is isolated and not externally accessible.

Hashing

Irreversible data is protected with cryptographic hashing rather than encryption:

- Passwords: bcrypt with cost factor 12 (approximately 250ms per hash on modern hardware)
- API key secrets: SHA-256 with random 16-byte salt, timing-safe comparison via crypto.timingSafeEqual()
- MFA backup codes: SHA-256 with per-code salt, single-use enforcement
- Refresh tokens: HMAC-SHA256 with the JWT secret as the HMAC key
- Device fingerprints: SHA-256 hash of user agent concatenated with IP address

6. Authentication & Identity



JWT Token Architecture

Authentication uses a short-lived access token / long-lived refresh token pair. This design limits the window of exposure if an access token is compromised while avoiding the latency of database-backed session validation on every request.

Token	Lifetime	Storage	Details
Access token	15 minutes	Client memory	JWT signed with HS256, contains userId, email, role
Refresh token	24 hours (default)	HttpOnly cookie / secure storage	Cryptographically random, 64 bytes, base64url-encoded
Refresh (remember me)	30 days	HttpOnly cookie / secure storage	Same generation, extended TTL

JWT tokens include issuer ("survey-platform") and audience ("survey-platform-users") claims. The verification step validates both claims, rejecting tokens issued by other services or intended for other audiences. The signing secret is sourced from the JWT_SECRET environment variable with no hardcoded fallback — the application will not start without a configured secret.

Token Rotation & Session Control

Refresh token rotation is enabled by default. Each time a client exchanges a refresh token for a new access/refresh pair, the previous refresh token is invalidated. If a previously-invalidated token is presented (indicating potential token theft), all sessions for that user are revoked. The system enforces a maximum of one active session per user — logging in from a new device terminates the existing session. Device fingerprinting (SHA-256 hash of user agent and IP) provides additional context for anomaly detection.

Password Security

Passwords are hashed using bcrypt with a cost factor of 12, which produces approximately 250ms of computational work per hash on current hardware — slow enough to deter brute-force attacks, fast enough to avoid user-perceptible login delay. Hashing occurs automatically via TypeORM @BeforeInsert and @BeforeUpdate lifecycle hooks on the User entity, ensuring that plaintext passwords never persist to the database even if a developer omits explicit hashing in the service layer.

Multi-Factor Authentication

The platform supports TOTP-based MFA (RFC 6238) with the following configuration:

- Time step: 30 seconds with a window of 1 step before/after (90 seconds total tolerance)
- Secret generation: authenticator.generateSecret() producing a base32-encoded shared secret
- QR code provisioning: OTP Auth URI rendered as a QR code via the qrcode library
- Backup codes: 10 codes per setup, 12-character hex format (XXXX-XXXX-XXXX), SHA-256 hashed
- Backup code warning: System alerts the user when only 2 codes remain
- Failed attempt lockout: 5 attempts maximum, 15-minute lockout duration
- Trusted devices: 30-day exemption with a maximum of 10 trusted devices per user
- Audit trail: MFA_ENABLED, MFA_DISABLED, MFA_VERIFIED, MFA_FAILED events recorded

SAML 2.0 Single Sign-On

Enterprise customers can configure SAML 2.0 SSO with any compliant identity provider. The implementation uses the @node-saml/node-saml library and includes the following security controls:

- Domain verification: DNS TXT record validation using a cryptographic token (survey-platform-verify={16-byte hex})
- Signed assertions: wantAssertionsSigned is enforced — unsigned assertions are rejected
- Certificate verification: The IdP's X.509 certificate is verified against the stored certificate on every assertion
- Just-in-time provisioning: New users are automatically created on first SSO login with a configured default role
- Metadata import: XML parsing extracts entityID, SSO/SLO URLs, and certificates from IdP metadata
- SSO activation is gated behind successful domain verification — an unverified domain cannot enable SSO

OAuth 2.0 Social Authentication

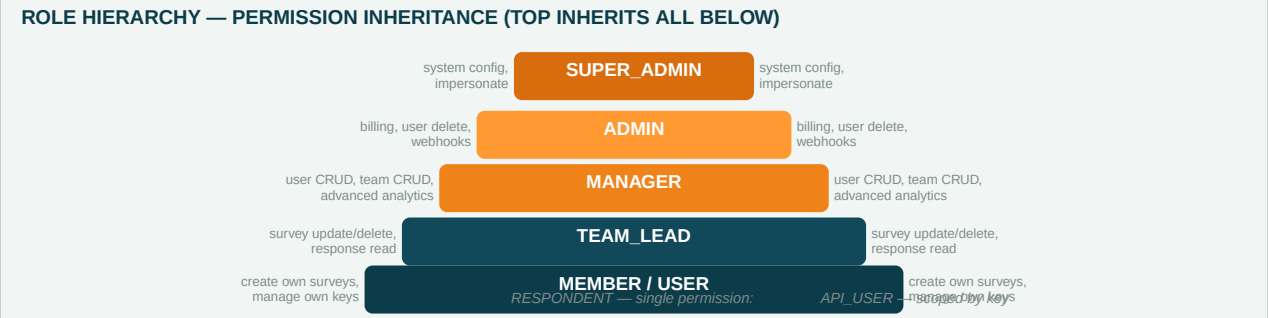
The platform supports social authentication through Google and Microsoft OAuth 2.0 providers. OAuth flows use the authorization code grant type (not the implicit grant), which keeps access tokens on the server side and never exposes them to the browser. The OAuth state parameter is cryptographically random to prevent CSRF attacks during the authentication callback. If a user authenticates via OAuth for the first time, a local account is provisioned with the email verified flag set to true, since the identity provider has already verified the email address.

Password Reset & Account Recovery

Password reset tokens are generated using crypto.randomBytes(32) and stored as SHA-256 hashes. The raw token is sent to the user's email and never stored in plaintext. Tokens expire after a configurable period (default: 1 hour). The password reset endpoint is rate-limited to 3 requests per hour per email address, preventing reset-based denial-of-service attacks. After a successful reset, all existing sessions for that user are revoked, forcing re-authentication on all devices.

7. Authorization & Access Control

Role Hierarchy



The platform defines eight roles in a strict hierarchy. Each role inherits all permissions from the roles below it. This eliminates permission gaps while keeping role definitions concise — an ADMIN does not need 43 explicit permission grants because it inherits from MANAGER, TEAM_LEAD, MEMBER, and USER.

Role	Inherits From	Key Exclusive Permissions
SUPER_ADMIN	All roles	System configuration, user impersonation
ADMIN	MANAGER and below	User deletion, role management, billing, API/webhook management
MANAGER	TEAM_LEAD and below	User CRUD, team CRUD, advanced analytics
TEAM_LEAD	MEMBER and below	Survey update/delete/archive, response read, analytics export
MEMBER	USER	Survey read (team-wide), team read
USER	(base role)	Create own surveys, manage own API keys, MFA/session management
RESPONDENT	(none)	Response create (single permission)
API_USER	(scoped)	Subset of permissions defined by API key scopes (SURVEY_READ, RESPONSE_EXPORT, etc.)

Permission System

The platform defines 42 granular permissions organized by domain: survey (9), response (5), analytics (3), user (5), team (5), organization (4), API/integration (4), admin (4), and security (3). Permissions use a noun:verb pattern (e.g., SURVEY_CREATE, RESPONSE_EXPORT, ADMIN_AUDIT_LOG) that maps directly to controller actions. An eighth role, API_USER, is assigned to API key-authenticated requests and inherits a scoped subset of permissions based on the key's configured scopes.

Authorization is enforced through NestJS guards and decorators. Four decorator patterns are available:

- @RequireRoles(...roles): Checks that the user holds one of the specified roles
- @RequirePermissions(...permissions): Checks that the user's role grants all specified permissions
- @RequireOwnership(ownerField): Verifies that the user owns the requested resource (e.g., survey.creatorId === userId)
- @RequirePermissionOrOwnership(permission, ownerField): Grants access if the user has the permission OR owns the resource

The RBAC guard evaluates in a fail-secure manner: if any check is indeterminate (for example, the resource cannot be loaded to verify ownership), access is denied. Denial events are written to the audit log with the user ID, attempted action, and resource identifier.

API Key Authorization

API keys are scoped to a set of permission grants. The available scopes map to the permission system: SURVEY_READ, SURVEY_WRITE, SURVEY_DELETE, RESPONSE_READ, RESPONSE_WRITE, RESPONSE_EXPORT, ANALYTICS_READ, USER_READ, USER_WRITE, and FULL_ACCESS. Each API request authenticated with a key is checked against both the key's scopes and the team membership of the key's owner. A key with SURVEY_READ scope can only read surveys belonging to the team it was issued under.

API keys are stored as SHA-256 hashes. The full key is displayed exactly once at creation time. Keys follow the format sk_live_{prefix}_{random} where the 8-character prefix allows key identification without exposing the secret. Keys support configurable rate limits, status tracking (ACTIVE, REVOKED, EXPIRED), and full usage audit trails.

8. API Security

Rate Limiting Strategy

Rate limiting is applied at two layers: the Nginx reverse proxy (connection-level) and the NestJS application (identity-aware). This dual-layer approach ensures that unauthenticated flood attacks are stopped at the proxy before consuming application resources, while authenticated abuse is tracked per user or API key.

Nginx Layer

Zone	Rate	Burst	Purpose
general	30 req/s per IP	20	All routes
api	20 req/s per IP	30	API endpoints
auth	5 req/s per IP	10	Login, register, password reset

Application Layer

Endpoint Category	Limit	Window	Tracking
Auth endpoints	10 attempts	1 minute	Per IP
Login	5 attempts	15 minutes	Per IP + email
Password reset	3 requests	1 hour	Per email
API key requests	1,000 requests	1 minute	Per key prefix
Sensitive operations	Configurable	Configurable	Per user

When a rate limit is exceeded, the response includes X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, and Retry-After headers. Repeated violations trigger security monitoring alerts — 10 rate limit violations within 5 minutes creates a RATE_LIMIT_EXCEEDED security alert.

Complete Rate Limit Decorator Reference

The following table lists every rate limit decorator available in the codebase. These are applied to controller methods via NestJS decorators, providing precise per-endpoint control:

Decorator	Limit	Window	Use Case
@ApiRateLimit	60 requests	1 minute	Standard API endpoints
@AuthRateLimit	10 requests	1 minute	Auth-related operations
@LoginRateLimit	5 attempts	15 minutes	Login endpoint (brute-force protection)
@PasswordResetRateLimit	3 requests	1 hour	Password reset (enumeration prevention)
@UploadRateLimit	10 requests	1 minute	File upload endpoints
@PublicRateLimit	200 requests	1 minute	Public/unauthenticated endpoints
@SearchRateLimit	30 requests	1 minute	Search and listing endpoints
@ExportRateLimit	5 exports	1 hour	Data export (CSV, Excel, PDF)

@WebhookRateLimit	100 requests	1 minute	Inbound webhook receivers
@SensitiveOperationRateLimit	3 operations	1 hour	Account deletion, role changes
@NoRateLimit	Unlimited	N/A	Health checks, internal endpoints

CORS Policy

Cross-Origin Resource Sharing is configured with a strict allowlist of origins. In production, only the web application URL, staff portal URL, and explicitly configured CORS_ORIGIN domains are permitted. Credentials (cookies, authorization headers) are enabled. The preflight cache is set to 86,400 seconds (24 hours) to minimize OPTIONS requests. Custom headers including X-API-Key, X-Request-ID, X-Device-Fingerprint, and X-MFA-Token are allowlisted.

Webhook Security

Outbound webhooks are signed with HMAC-SHA256 using a per-webhook secret. Recipients can verify the signature to confirm that the payload originated from Claform and was not tampered with in transit. The webhook system includes a circuit breaker pattern: after repeated delivery failures, the webhook is automatically disabled to prevent queue buildup. Webhook secrets support rotation, and all delivery attempts are logged with status, response code, and latency.

Integration Credential Security

The platform connects to 44 third-party services including Salesforce, HubSpot, Slack, Microsoft Teams, Jira, and Stripe. Each integration stores OAuth tokens or API credentials that must be protected with the same rigor as user passwords. Integration credentials are encrypted at rest using AES-256-CBC with a random initialization vector per credential. The encryption key is derived from the platform's master encryption key and is not stored alongside the encrypted credentials.

A ConnectionHealthService periodically validates that stored credentials remain valid. An automated CircuitBreaker disables integrations that fail repeatedly, preventing cascading errors. When an OAuth token is refreshed, the old token is immediately invalidated and the new token is encrypted before being stored.

Payment Processing Security

Subscription billing is handled entirely through Stripe. Claform never stores, processes, or transmits credit card numbers, CVVs, or bank account details. Payment forms use Stripe Elements, which render in an isolated iframe directly served by Stripe's PCI DSS Level 1 certified infrastructure. The platform stores only Stripe customer and subscription identifiers.

Stripe webhook events are verified using the Stripe-Signature header, which contains an HMAC-SHA256 signature computed over the raw request body. The platform preserves the raw body via a dedicated RawBodyMiddleware that runs before JSON parsing, ensuring the signature verification operates on the exact bytes received. Webhook events processed include checkout.session.completed, customer.subscription.updated, customer.subscription.deleted, invoice.payment_succeeded, and invoice.payment_failed.

9. Distribution Channel Security

Claform distributes surveys through eight channels, each with distinct security and compliance requirements. This section describes the channel-specific controls that protect data and ensure regulatory compliance during survey delivery. All channels share the same core security infrastructure (TLS, RBAC, audit logging) described in earlier sections; this section covers the controls unique to each distribution method.

Email Distribution

Email campaigns are processed asynchronously through BullMQ with three provider options: custom SMTP servers via Nodemailer with connection pooling (20 max connections, 500 messages per connection), AWS SES with configuration sets, and SendGrid with API-based delivery. Provider failover is automatic — if the primary provider fails with a transient error, the system retries with exponential backoff (30-second initial delay, 3 attempts maximum).

The email processor classifies errors as transient or permanent. Transient errors (timeout, connection refused, SMTP codes 421/450/451/452, rate limits) trigger retries. Permanent errors (authentication failure code 535/EAUTH, invalid recipients) route to the dead-letter queue without retry. SMTP transport connections are pooled with a 5-minute idle cleanup cycle. Campaign pause state is checked via Redis O(1) lookup before each job is processed, allowing instant campaign suspension without draining the queue.

SMS Distribution

SMS delivery supports three providers (Twilio, AWS SNS, Vonage/Nexmo) with per-country provider preferences and automatic failover. The platform supports 60+ countries organized into six regions, each with country-specific configurations for cost estimation, delivery rates, regulations, quiet hours, and timezone support.

Compliance controls include E.164 phone number validation, opt-out keyword detection (STOP, UNSUBSCRIBE), multi-scope opt-out management (global, campaign-level, survey-level), and pre-send compliance checks. SMS events are stored in an append-only table for regulatory audit. Error classification maps provider-specific codes: Twilio 21211 (invalid number), 21610 (blacklisted), carrier rejections, and rate limit responses are each handled with appropriate retry or dead-letter routing.

WhatsApp Distribution

WhatsApp campaigns require explicit consent management. The platform implements a full double opt-in workflow: initial consent creates a PENDING record, a verification message is sent via WhatsApp (interactive button preferred, text fallback), and consent is confirmed only after the recipient clicks the verification button. Tokens expire after 24 hours.

Six consent types are tracked (marketing, transactional, service, surveys, notifications, all) from 12 possible sources (web form, mobile app, SMS, paper, phone, API, import, QR code, click-to-chat, double opt-in, existing customer, manual). Consent expires after 2 years by default, with a daily cron job that automatically expires stale consents. Country-specific quiet hours are enforced — messages are queued, not sent, during restricted periods. GDPR data deletion requests anonymize all message history and delete consent records.

Voice/IVR Distribution

Voice surveys use Twilio's programmable voice API with TwiML (Twilio Markup Language) for dynamic call flow generation. DTMF (touch-tone) input is collected via the Gather TwiML verb. Machine detection identifies answering machines and voicemail. Call state transitions are tracked (QUEUED, INITIATED, RINGING, IN_PROGRESS, COMPLETED/FAILED), and campaign completion is monitored by checking when all calls reach terminal states. Webhook callbacks from Twilio are validated using request signature verification.

Kiosk, Print & Workspace Channels

Kiosk mode uses device-specific tokens (32-byte cryptographically random hex) for authentication, with optional admin approval gates before devices can collect responses. Offline responses are synced with duplicate detection (clientResponseId matching) and configurable conflict resolution (server-wins or client-wins). PIN-to-exit security prevents unauthorized kiosk access.

Print distribution generates PDFs with embedded QR codes linking to the survey URL. QR codes support configurable error correction levels (L/M/Q/H) and custom colors. PDF files are stored as binary data (PostgreSQL bytea) with automatic expiry and cleanup via scheduled cron jobs. Slack and Microsoft Teams integration distributes surveys through workspace bots with channel-level access control.

Cross-Channel Security Summary

Control	Email	SMS	WhatsApp	Voice	Kiosk
Provider failover	SES/SendGrid/ SMTP	Twilio/SNS/Nexmo	Meta API only	Twilio only	N/A
Compliance	Bounce handling	E.164, opt-out, quiet hours	Double opt-in, GDPR deletion	Signature verification	Device approval, PIN
Rate limiting	50-200 msg/sec per provider	Provider + BullMQ	80 msg/sec (Meta limit)	Provider limits	Per-device
Audit trail	EmailEvent table	SmsEvent (append-only)	ComplianceAudit + events	Call state transitions	Sync status log
Data retention	Campaign job history	Opt-out history retained	2-year consent expiry	Call SID in DB	Package expiry
Error handling	Transient vs permanent	Provider code mapping	Consent blocking	Machine detection	Conflict resolution

10. Integration & Workflow Security

Claform connects to 44 third-party services across 20 categories: CRM (Salesforce, HubSpot, Pipedrive, Zoho, Freshsales), project management (Jira, Asana, Monday, Trello), communication (Slack, Microsoft Teams), email marketing (Mailchimp, Brevo, ActiveCampaign, Klaviyo), analytics (Google Analytics, Segment, Mixpanel, Amplitude), business intelligence (Power BI, Tableau), storage (Google Drive, Dropbox, OneDrive), payments (Stripe, PayPal), helpdesk (Zendesk, Freshdesk, Intercom), e-commerce (Shopify, WooCommerce), rewards (Tremendous, Tango Card), and automation (Zapier, Make). Each integration stores credentials that must be protected with the same rigor as user passwords.

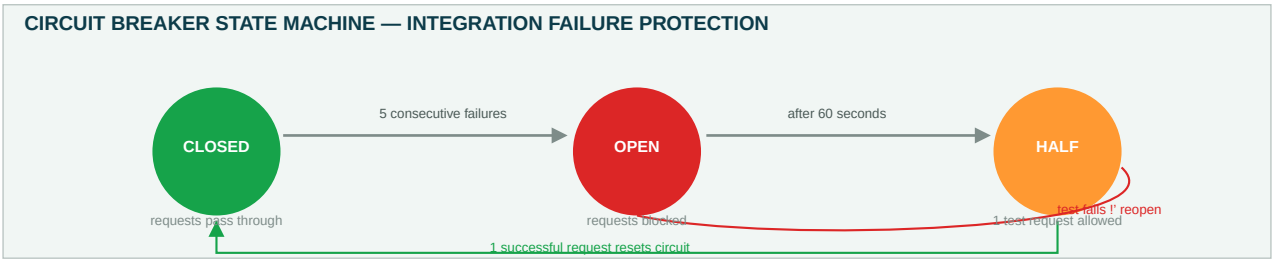
Credential Encryption

Integration credentials (OAuth access tokens, refresh tokens, API keys) are encrypted at rest using AES-256-GCM with a randomly generated 16-byte IV per encryption. The encrypted payload is stored in the format iv_hex:authTag_hex:ciphertext_hex. The encryption key is derived from the platform master secret using SCRYPT, producing a 32-byte key. Decryption validates the GCM authentication tag, detecting any tampering with the stored credentials.

OAuth Token Lifecycle

A scheduled cron job runs every 10 minutes to check for OAuth tokens expiring within the next hour. For each expiring token, the system executes the provider-specific refresh flow: POST to the provider's token endpoint with the refresh token and client credentials (sourced from environment variables, not stored in the database). Supported providers include Salesforce, HubSpot, Google, Notion, Airtable, Dropbox, OneDrive, Jira, Asana, Zendesk, Intercom, Zoho, Teams, and PayPal. If a refresh fails, the connection is marked EXPIRED and requires manual reauthorization.

Circuit Breaker Pattern



Outbound requests to third-party APIs are protected by a circuit breaker to prevent cascading failures. The circuit has three states: CLOSED (normal operation), OPEN (all requests blocked), and HALF_OPEN (testing recovery). After 5 consecutive failures, the circuit opens for 60 seconds. During the open period, all requests return immediately without contacting the provider. After 60 seconds, a single test request is allowed (HALF_OPEN); if it succeeds, the circuit closes. If it fails, the circuit reopens for another 60 seconds.

Webhook Delivery Security

Outbound webhooks support five authentication methods: HMAC-SHA256, HMAC-SHA512, HTTP Basic Auth, Bearer Token, and API Key Header. HMAC signatures include an optional timestamp to prevent replay attacks — the signed payload is formatted as {timestamp}.{json_body}. Custom header names and signature prefixes (sha256=, sha512=) are configurable per webhook.

Retry logic uses exponential backoff with jitter: base delay 1 second, multiplier 2, maximum delay 30 seconds, with plus-or-minus 50% random variation to prevent thundering herd effects when multiple webhooks fail simultaneously. After 3 failed attempts, the delivery is logged as failed. IP allowlist filtering restricts which CIDR

filtering restricts which CIDR blocks can receive webhook deliveries.

Workflow Engine Security

The workflow engine executes automated actions triggered by survey events (response completed, NPS scored, scheduled). Workflow conditions support nested rule groups with AND/OR logic, limited to a maximum depth of 10 to prevent runaway evaluation. Circular workflow chains (A triggers B triggers A) are detected and blocked via a chain depth counter.

SSRF (Server-Side Request Forgery) protection validates all webhook URLs before execution via a dedicated checkSsrfUrl() function. Workflow execution is processed asynchronously through BullMQ, with per-plan usage limits (Free: 50 runs/month, Starter: 500, Professional: 5,000, Business: 25,000, Enterprise: unlimited). Active workflows are cached in Redis with a 5-minute TTL to eliminate repeated database lookups during high-volume event processing.

11. Security Monitoring & Audit

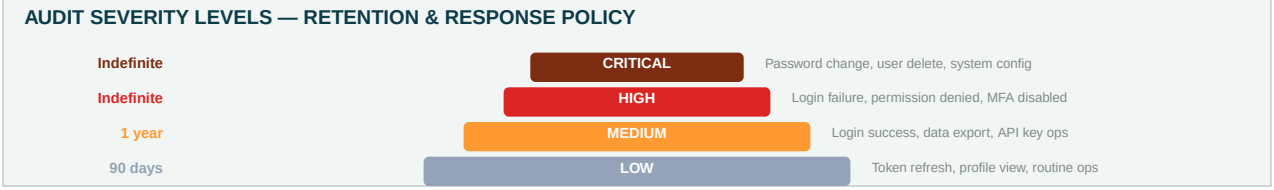
Audit Logging

The audit system captures every security-relevant action in a structured, queryable format. Each audit entry records the category, action, severity, status, user context (ID, email, role), resource context (type, ID, name), request context (IP, user agent, HTTP method, endpoint, request ID, duration), and the before/after values for data mutations.

Tracked Actions

The system tracks 43 distinct action types across six categories:

- Authentication: LOGIN_SUCCESS, LOGIN_FAILURE, LOGOUT, TOKEN_REFRESH, PASSWORD_CHANGE, PASSWORD_RESET
- MFA: MFA_ENABLED, MFA_DISABLED, MFA_VERIFIED, MFA_FAILED
- User management: USER_CREATE, USER_UPDATE, USER_DELETE, USER_ACTIVATE, USER_DEACTIVATE, ROLE_CHANGE
- Data operations: SURVEY_CREATE, SURVEY_UPDATE, SURVEY_DELETE, SURVEY_PUBLISH, RESPONSE_SUBMIT, RESPONSE_EXPORT, RESPONSE_DELETE
- Security events: PERMISSION_DENIED, RATE_LIMIT_EXCEEDED, SUSPICIOUS_ACTIVITY, SESSION_REVOKED, IP_BLOCKED
- API operations: API_KEY_GENERATED, API_KEY_REVOKED, API_KEY_INVALID, WEBHOOK_TRIGGER, SYSTEM_CONFIG_CHANGE



Severity Classification

Severity	Examples	Retention
CRITICAL	Password change, role change, user delete, system config change, IP block	Indefinite
HIGH	Login failure, permission denied, rate limit, MFA disabled, data delete	Indefinite
MEDIUM	Login success, user create/update, data export, API key operations, MFA verify	1 year
LOW	Token refresh, profile view, routine operations	90 days

Sensitive Data Redaction

Before writing to the audit log, the system scans all metadata for sensitive field names: password, token, secret, apiKey, authorization, creditCard, ssn, and refreshToken. Matching fields are replaced with "[REDACTED]". The scan is recursive, case-insensitive, and applies to nested objects. This ensures that a developer accidentally passing a full request body to the audit logger does not result in credentials being written to the audit table.

Security Monitoring & Alerting

A dedicated SecurityMonitoringService aggregates security events and generates alerts when thresholds are exceeded. This provides automated detection of attack patterns that may span multiple individual audit entries.

Alert Type	Threshold	Window
BRUTE_FORCE_ATTEMPT	5 failed logins from same IP	15 minutes
SUSPICIOUS_LOGIN	Login from new geography or device	Per event
SQL_INJECTION_ATTEMPT	SQL pattern detected in input	Per event
XSS_ATTEMPT	Script/event handler pattern in input	Per event
RATE_LIMIT_EXCEEDED	10 violations from same source	5 minutes
API_KEY_ABUSE	Anomalous API key usage pattern	Configurable
ACCOUNT_LOCKOUT	MFA lockout triggered	Per event
BULK_DATA_EXPORT	1,000+ records accessed	Per operation
PERMISSION_ESCALATION	Attempt to access higher-privilege resource	Per event

Alerts follow a lifecycle: NEW, ACKNOWLEDGED, RESOLVED, or FALSE_POSITIVE. A security dashboard provides real-time statistics with 24-hour and 7-day trend calculations. False positive tracking helps reduce alert fatigue over time.

12. Compliance & Regulatory

General Data Protection Regulation (GDPR)

GDPR compliance is embedded in the platform architecture, not layered on top of it. Data subject rights are supported through dedicated service methods and API endpoints:

- Right of access: Data export endpoint returns all personal data held for a data subject in a machine-readable format
- Right to rectification: Users can update their personal data through standard profile management
- Right to erasure: Account deletion anonymizes or removes all personal data, with audit trail entries preserved in anonymized form
- Right to data portability: Export in standard JSON/CSV formats
- Consent management: WhatsApp and SMS channels implement explicit opt-in with consent source tracking, expiry management, and revocation
- Data processing records: Audit logs serve as processing activity records required under Article 30

Telecommunications Compliance

SMS and voice distribution channels implement compliance controls for TCPA (United States), CASL (Canada), LGPD (Brazil), and regional equivalents across 68 supported countries.

- Opt-out management: Global, campaign-level, survey-level, and user-level opt-out scopes with keyword detection (STOP, UNSUBSCRIBE, etc.)
- Quiet hours: Country-specific quiet hour enforcement — messages are queued, not sent, during restricted periods
- Consent tracking: Consent type (marketing, transactional, survey), source (web form, SMS, QR code, etc.), and verification status recorded per contact
- Double opt-in: WhatsApp channel supports double opt-in workflow with token-based verification and configurable expiry
- Immutable event logs: SMS and WhatsApp message events are stored in append-only tables for regulatory audit purposes

Data Residency & Retention

Audit log retention is configurable by severity. Low-severity operational logs are automatically purged after 90 days. Critical security events and compliance-relevant records are retained indefinitely. Data retention policies can be configured per regulation and per country, supporting GDPR's 30-day right-to-be-forgotten window alongside TCPA's longer retention requirements for consent records.

Feature-Level Compliance Gating

Certain compliance-sensitive features require explicit administrative approval regardless of subscription plan: HIPAA compliance mode, custom integrations, and data residency configuration. This prevents accidental activation of features that carry regulatory obligations.

OWASP Top 10 Coverage

The following table maps each OWASP Top 10 (2021) risk category to the specific controls implemented in Claform. This mapping is provided to assist security reviewers in validating coverage against the most widely recognized web application security standard.

OWASP Risk	Platform Controls
A01: Broken Access Control	RBAC with 7 roles and 42 permissions, team-scoped queries, ownership verification guards, fail-secure denials
A02: Cryptographic Failures	AES-256-GCM at rest, TLS 1.2/1.3 in transit, bcrypt-12 passwords, 90-day key rotation, no hardcoded secrets
A03: Injection	Parameterized queries (TypeORM), SecureValidationPipe whitelist, RequestSanitizationMiddleware, HTML entity encoding
A04: Insecure Design	Defense-in-depth architecture, threat modeling per feature, multi-layer validation, fail-secure defaults
A05: Security Misconfiguration	Helmet CSP enforcement, HSTS preload, Permissions-Policy, no default credentials, environment validation at startup
A06: Vulnerable Components	Dependabot monitoring, npm audit in CI, pinned base images, multi-stage Docker builds excluding dev dependencies
A07: Auth Failures	bcrypt-12 hashing, TOTP MFA, 15-min token expiry, refresh token rotation, progressive lockout, session revocation
A08: Data Integrity Failures	HMAC-SHA256 webhook signing, Stripe signature verification, GCM authentication tags, audit trail for all mutations
A09: Logging & Monitoring	30+ audit actions, 20 alert types, severity classification, sensitive field redaction, SecurityMonitoringService
A10: Server-Side Request Forgery	No user-controlled URL fetching in core flows, integration URLs validated against allowlists, internal network isolated

13. Mobile SDK Security

The platform provides native SDKs for React Native, Flutter, iOS (Swift/SwiftUI), and Android (Kotlin/Jetpack Compose). All SDKs share a common security architecture derived from the TypeScript core engine, adapted to each platform's security primitives.

Encrypted Storage

An EncryptedStorageAdapter wraps each platform's secure storage mechanism (Keychain on iOS, EncryptedSharedPreferences on Android, expo-secure-store on React Native, flutter_secure_storage on Flutter) with transparent AES-256-GCM encryption. The adapter includes migration-safe fallback: if existing unencrypted data is detected, it is transparently encrypted on first access rather than requiring a destructive migration.

Biometric Authentication

Sensitive surveys can require biometric authentication (fingerprint or face recognition) before the survey loads. This gate is enforced in the loadSurvey() method of the core SDK and delegated to platform-specific biometric adapters. The biometric check is in addition to, not a replacement for, the standard API authentication.

Offline Data Protection

When surveys are completed offline, responses are encrypted before being written to local storage. A ConflictEngine with four resolution strategies (client-wins, server-wins, latest-wins, merge) handles synchronization conflicts when connectivity is restored. Background sync is triggered automatically on network reconnection via the ConnectivityAdapter. All offline responses are queued and transmitted over TLS once the device regains connectivity.

SDK Event Isolation

The SDK EventBus uses key-prefix isolation to prevent cross-survey event leakage when multiple SDK instances run in the same application. Events from Survey A cannot trigger handlers registered by Survey B, even if both share the same JavaScript runtime.

Platform Security Comparison

Each mobile SDK platform implements the same security architecture using platform-native primitives. The following table shows how each security feature is realized across platforms:

Security Feature	iOS (Swift)	Android (Kotlin)	React Native	Flutter
Secure storage	Keychain Services	EncryptedSharedPreferences	expo-secure-store	flutter_secure_storage
Biometric auth	LocalAuthentication (Face ID / Touch ID)	BiometricPrompt API	expo-local-authentication	local_auth package
Network pinning	URLSession delegate	OkHttp CertificatePinner	React Native SSL pinning	Dio with certificates
Offline encryption	CryptoKit AES-GCM	javax.crypto AES-GCM	expo-crypto	encrypt package
Background sync	BGTaskScheduler	WorkManager	Background fetch	workmanager package

14. Staff & Administrative Security

The staff portal — an internal application for platform administration — implements elevated security controls that exceed the requirements of the customer-facing application.

Security Clearance Levels

Level	Min Password	Access Token TTL	Refresh Token TTL	MFA Required
CRITICAL	32 characters	15 minutes	1 hour	Yes
CONFIDENTIAL	24 characters	30 minutes	4 hours	Yes
RESTRICTED	16 characters	1 hour	8 hours	Yes
INTERNAL	12 characters	1 hour	8 hours	Yes
PUBLIC	12 characters	1 hour	8 hours	No

Staff Password Policy

Staff passwords are subject to stricter requirements than customer passwords: a minimum length of 12 to 32 characters depending on clearance level, mandatory complexity (uppercase, lowercase, numbers, special characters), a 24-password history (preventing reuse of the last 24 passwords), a 180-day maximum age with a 7-day expiry warning, and pattern blocking that detects repeated characters, sequential numbers, keyboard patterns, and common dictionary words. Integration with a breach database flags passwords known to be compromised.

Progressive Brute-Force Protection

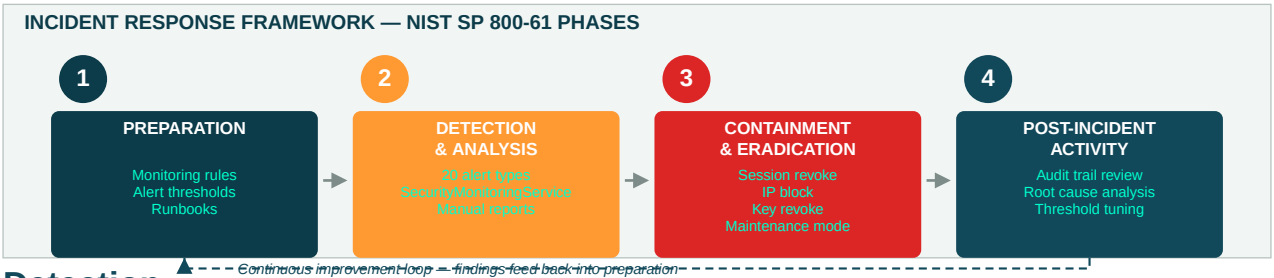
The staff portal uses a dedicated BruteForceGuard with exponential backoff. Failed authentication attempts increase the lockout duration progressively, up to a maximum of 24 hours. The guard tracks attempts by IP + endpoint combination, logs blocked attempts to the audit trail, and supports manual unblock by a CRITICAL-level administrator.

Hidden Route Configuration

The staff portal URL is not publicly discoverable. The portal path is derived from an HMAC-SHA256 hash of a secret key combined with a time-based rotation period (configurable as hourly, daily, weekly, or monthly). Both the current and previous period paths are valid simultaneously, providing a grace window during rotation. This does not replace authentication — it adds an additional layer of obscurity that prevents automated scanners from discovering the administrative interface.

15. Incident Response

Claform maintains a documented incident response plan that defines roles, escalation procedures, and communication protocols for security events. The plan follows the NIST SP 800-61 framework with four phases: preparation, detection and analysis, containment and eradication, and post-incident activity.



Detection

Automated detection is provided by the SecurityMonitoringService described in Section 9. Manual detection channels include the security@claform.com reporting address, the vulnerability disclosure program, and internal team reporting. All detection sources feed into the same alert management pipeline.

Response Capabilities

The platform supports the following containment actions, all available via API or staff portal:

- Session revocation: Immediately terminate all active sessions for a specific user or team
- API key revocation: Instantly invalidate compromised API keys with audit logging
- Account suspension: Disable user accounts with a SUSPENDED enum status that prevents authentication
- IP blocking: Block specific IP addresses at the application layer with audit logging
- Rate limit override: Dynamically adjust rate limits on specific endpoints during an active incident
- Maintenance mode: Disable non-essential operations while preserving monitoring and admin access

Notification & Communication

When a security incident is confirmed, the platform supports notification through multiple channels. The real-time WebSocket notification gateway delivers alerts to active staff portal sessions immediately. Email notifications are sent to the security team distribution list. For incidents affecting customer data, the audit trail provides the information needed to identify affected organizations, determine the scope of exposure, and generate regulatory notification reports within the timelines required by GDPR (72 hours), CCPA, and other applicable regulations.

Post-Incident Review

Post-incident reviews produce actionable findings that are tracked through resolution. The audit trail provides a forensic record of all actions taken during the incident window, including who accessed what data, which sessions were active, and which API keys were in use. This evidence chain supports both internal review and regulatory notification requirements.

The review process includes root cause analysis, timeline reconstruction from audit logs, identification of control gaps, and documentation of remediation actions. Findings from post-incident reviews feed directly into the security monitoring thresholds — if an attack pattern was detected too late, the alert thresholds are tightened. This creates a continuous improvement loop that strengthens the platform's security posture after each incident.

16. Appendix: Glossary

The following terms are used throughout this whitepaper. Definitions reflect the specific meaning within the Claform security context.

Term	Definition
AES-256-GCM	Advanced Encryption Standard with 256-bit keys in Galois/Counter Mode. Provides both confidentiality and integrity verification.
bcrypt	Password hashing function with configurable work factor. Cost factor 12 means 2^12 (4,096) internal iterations per hash.
BullMQ	Redis-backed job queue for Node.js. Used for asynchronous email, SMS, WhatsApp, and voice campaign delivery.
CORS	Cross-Origin Resource Sharing. HTTP mechanism that controls which domains can make requests to the API.
CSP	Content Security Policy. HTTP header that restricts which resources (scripts, styles, frames) a page can load.
DTO	Data Transfer Object. Typed class that defines the expected shape of a request body, enforcing validation before processing.
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral. Key exchange protocol providing forward secrecy — past sessions remain safe if the server key is later compromised.
GCM	Galois/Counter Mode. Block cipher mode that produces an authentication tag alongside ciphertext, detecting tampering.
HSTS	HTTP Strict Transport Security. Header instructing browsers to only connect via HTTPS, preventing protocol downgrade attacks.
HMAC-SHA256	Hash-based Message Authentication Code using SHA-256. Used for webhook signatures and token verification.
JIT Provisioning	Just-In-Time Provisioning. Automatic creation of a user account on first SSO login, without manual admin setup.
JWT	JSON Web Token. Compact, signed token containing user claims (ID, role, email) used for stateless API authentication.
OCSP Stapling	Server-side certificate status check that avoids the latency of clients checking certificate revocation independently.
OWASP Top 10	Industry-standard list of the ten most critical web application security risks, published by the Open Web Application Security Project.
PBKDF2	Password-Based Key Derivation Function 2. Derives encryption keys from a master password using iterated hashing (SHA-512, 100K rounds).
RBAC	Role-Based Access Control. Authorization model where permissions are assigned to roles, and roles are assigned to users.
SAML 2.0	Security Assertion Markup Language. XML-based protocol for exchanging authentication data between an identity provider and a service provider.
TOTP	Time-Based One-Time Password (RFC 6238). Generates a 6-digit code every 30 seconds using a shared secret and the current time.

TypeORM	Object-Relational Mapping library for TypeScript. Generates parameterized SQL queries, preventing SQL injection by design.

17. Conclusion

Claform's security architecture reflects the reality that survey and feedback platforms handle sensitive data by default. Whether a customer collects employee satisfaction scores, patient health assessments, or consumer preference data, the platform treats all data with the same rigor: encrypted at rest and in transit, accessible only through authenticated and authorized channels, and monitored by automated detection systems that alert on anomalous behavior.

The controls described in this whitepaper are not aspirational — they are the production system. AES-256-GCM encryption keys rotate every 90 days. Passwords are hashed with bcrypt-12. JWT access tokens expire in 15 minutes. Rate limits enforce 5 login attempts per 15 minutes. These are not configurable options that a customer might enable; they are defaults that a customer cannot weaken.

Security is a continuous practice, not a destination. We conduct regular penetration tests, maintain dependency scanning in our CI/CD pipeline, and refine our monitoring thresholds based on observed traffic patterns. We welcome security inquiries from prospective and current customers and are prepared to complete vendor security questionnaires, share SOC 2 reports under NDA, and schedule architecture walkthroughs with your security team.

Summary of Key Controls

Domain	Primary Control	Key Specification
Data at rest	AES-256-GCM field encryption	PBKDF2-SHA512, 100K iterations, 90-day rotation
Data in transit	TLS 1.2/1.3 only	ECDHE forward secrecy, HSTS preload
Authentication	JWT + refresh token pair	15-min access, rotation on use, 1 session max
MFA	TOTP (RFC 6238)	10 backup codes, 5-attempt lockout, trusted devices
SSO	SAML 2.0	DNS domain verification, signed assertions, JIT provisioning
Authorization	RBAC (8 roles, 42 permissions)	Fail-secure evaluation, ownership verification
Rate limiting	Dual-layer (Nginx + app)	Auth: 5/15min, API: 1K/min, progressive lockout
Input validation	8 validation pipes	Whitelist enforcement, recursive sanitization
Audit	43 action types	4 severity levels, sensitive field redaction, indefinite retention for critical events
Monitoring	20 alert types	Automated thresholds, false-positive tracking, dashboard
Multi-tenancy	3-layer isolation	Service + guard + API key scoping
Compliance	GDPR, TCPA, CASL, LGPD	Consent management, opt-out, quiet hours, data export/deletion
Mobile SDKs	Encrypted storage + biometric gates	5 platforms, offline encryption, event isolation
Staff security	5 clearance levels	32-char passwords, HMAC-based hidden routes, 24-password history

Next Steps for Evaluators

If you are evaluating Claform for your organization, we recommend the following steps to complete your security assessment:

- Review this whitepaper against your organization's security requirements and vendor evaluation checklist
- Request a live security architecture walkthrough — our engineering team can answer technical questions in real time
- Request our SOC 2 Type II report under NDA for detailed third-party audit findings
- Submit your vendor security questionnaire — we maintain pre-completed responses for common frameworks (SIG, CAIQ, VSAQ)
- Contact security@claform.com for any questions not covered in this document

> *For security inquiries, vulnerability reports, or to request a security review, contact security@claform.com. For responsible disclosure of security vulnerabilities, we operate a safe harbor program — researchers who act in good faith will not face legal action.*

Document version 2.1, published April 2026. This document is reviewed and updated quarterly or following significant architectural changes. The most current version is always available at claform.com/security/whitepaper.

Claform

Enterprise Survey & Feedback Platform

CONTACT

Security Team
security@claform.com

Sales
sales@claform.com

General
hello@claform.com

Website
claform.com

Version 2.1 — April 2026