Stracket

PUBLIC DOCUMENT

Stracket Security Document

Screening individuals and companies against PEP and sanctions lists, anywhere in the world, in seconds. This document describes the technical and organizational measures that protect the platform and the data entrusted to it.

PLATFORM Stracket KYC/AML screening and monitoring platform	DOCUMENT VERSION 1.0
LAST UPDATED 5 August 2026	CLASSIFICATION Public. May be shared with (prospective) customers.

Table of contents

01 Purpose and scope

Stracket is a screening and monitoring platform for KYC (Know Your Customer) and AML (Anti-Money Laundering) processes. Customers use the platform to record subjects, both natural persons and legal entities, and to screen them against international sanctions and watchlists and against adverse media sources, and to keep those subjects under continuous monitoring.

That workflow involves personal data of our customers' clients. Security is therefore treated as a functional requirement of the product. This document describes the technical and organizational measures that protect the Stracket platform and the data entrusted to it.

IN SCOPE

the Stracket web application (`stracket.com`), the Stracket API, the asynchronous processing layer, and the production database and file storage.

OUT OF SCOPE

the security of customer-side systems, customer end-user devices, and the internal networks from which customers access the platform.

VENDOR ASSESSMENTS

This document describes controls that are implemented and active today. It is written at the level of detail appropriate for a public document. Customers with a formal vendor assessment process are welcome to request further detail under a mutual non-disclosure agreement (see section 11).

02 Platform architecture

Stracket is a three-tier application with a clear separation between presentation, business logic and storage.

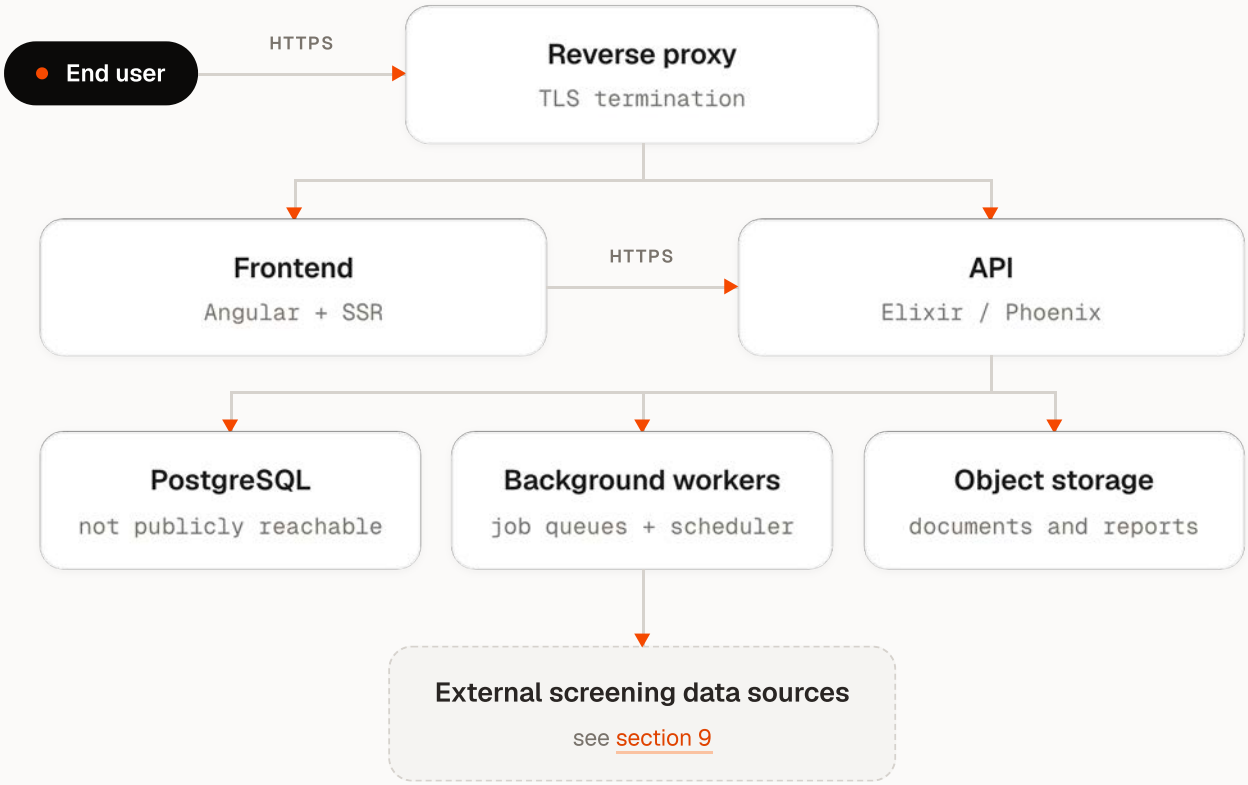

FIGURE 1 · STRACKET PLATFORM ARCHITECTURE

Frontend. A single-page application built with Angular, server-side rendered on Node.js. The frontend holds no business logic that governs access to data. It renders what the API is willing to return for the authenticated user.

API. An Elixir application on the Phoenix framework. All data access, reads and writes alike, passes through this API, which enforces authentication, authorization and validation on every request. The API surface is formally described in an OpenAPI specification.

Background processing. Screening runs, adverse media searches and recurring monitoring are executed asynchronously by dedicated job queues and a scheduler. This keeps long-running third-party lookups off the request path and makes them individually retryable and observable.

Storage. A PostgreSQL relational database holds all structured data. Uploaded documents and generated reports are stored in cloud object storage.

03 Hosting and network security

3.1 Hosting environment

The production environment is hosted with an established cloud infrastructure provider in the **United States**. The application, background workers and the PostgreSQL database all run on cloud infrastructure in that country. Uploaded files and generated documents are stored in cloud object storage (see section 8.3).

The hosting provider is responsible for the physical security of the datacenter and the underlying network, and publishes its own compliance attestations for the infrastructure layer.

Separate environments are maintained for development, testing and production. Test and development environments run on separate infrastructure and do not contain production data.

3.2 Network exposure

The production environment exposes a minimal surface to the internet. Only the ports required to serve the application are reachable: HTTPS on port 443, and port 80 solely to issue a permanent redirect to its HTTPS equivalent. All other services are closed at the firewall.

The PostgreSQL database accepts no connections from outside the host it runs on. There is no network route by which a client on the internet can reach the database directly. Internal application components are not published to the public internet and are reachable only through the reverse proxy that terminates public traffic.

3.3 Administrative access

Interactive access to production systems is restricted to the engineers who operate the platform, is limited to an allowlist of approved source addresses, and requires authentication. Access rights are reviewed when someone joins or leaves that group. There is no shared administrative account for customer-facing use, and customer accounts are not used to access customer data.

04 Encryption

4.1 Data in transit

All traffic between end users and the platform is encrypted with TLS. HTTPS is enforced: the reverse proxy terminates TLS with certificates from a recognized certificate authority, and any request arriving over plain HTTP is redirected to the HTTPS equivalent. The application additionally instructs browsers, through its Content Security Policy, to upgrade insecure requests and to block mixed content, so no sub-resource can be loaded over an unencrypted connection.

Traffic between the API and external screening providers, mail delivery and cloud storage is likewise TLS-encrypted. All integrations use HTTPS endpoints.

4.2 Data at rest

Database and application volumes reside on hosting infrastructure that encrypts data at rest at the infrastructure layer. Objects in cloud object storage are encrypted at rest by the storage provider. This encryption is applied automatically to every object as it is written and cannot be disabled.

User passwords are never stored in a recoverable form. They are hashed with a **strong, adaptive password-hashing algorithm**, using a unique per-password salt and a deliberately costly work factor. A password cannot be derived from the stored hash, and Stracket personnel cannot read or recover a user's password.

05 Identity and access management

5.1 Authentication

The platform is fully closed behind a login. There is no anonymous read access to customer data anywhere in the API. Users authenticate with an email address and password combination.

5.2 Multi-factor authentication

Multi-factor authentication is **enabled on production accounts**. After a successful password check, the login does not immediately yield a session. The platform generates a single-use verification code, delivers it to the user's registered email address, and issues session tokens only after that code is submitted correctly.

Verification attempts are limited. After a small number of incorrect attempts the challenge is invalidated and the user must start a new login. Each challenge is bound to a specific single-use verification token, so a code issued for one login attempt cannot be replayed against another.

5.3 Passwords

Passwords are validated on registration and on change, hashed before storage, and never logged or included in API responses. Changing a password requires the current password to be supplied and verified.

A forgotten-password flow issues a time-limited, single-use reset token by email to the address on file. The token, rather than the old password, authorizes the change.

5.4 Sessions

Sessions are based on cryptographically signed tokens. Access tokens expire automatically, after which the client must present a refresh token to obtain a new one.

Refresh tokens are additionally recorded server-side, which means they can be **revoked**. Logging out invalidates the refresh token immediately, so it can no longer be exchanged for new access tokens. Expired token records are removed automatically by a recurring cleanup process.

5.5 API keys

Customers integrating Stracket directly from their own systems authenticate with a workspace API key rather than a user session. Each workspace is issued its own randomly generated, high-entropy key, presented in a dedicated request header.

API-key authentication is confined to a dedicated set of integration endpoints and resolves strictly to the one workspace that owns the key. An unrecognized or missing key is rejected before any application logic runs, and the response is indistinguishable from a "not found" response so that key validity cannot be probed. Keys can be rotated on request.

06 Authorization and tenant isolation

Authorization is enforced server-side on every request. Data cannot be reached by manipulating the frontend, guessing an identifier or calling the API directly, because the frontend is not the component that decides what a user may see.

6.1 Role model

Each user account carries a role. Customer users operate strictly within the workspaces they belong to. Elevated roles exist for platform operations and are held only by the personnel who run the service.

6.2 Workspace isolation

Stracket is multi-tenant. A **workspace** is the tenant boundary, and subjects, checks, monitors, reports, documents, orders and audit records all belong to exactly one workspace.

Every request that addresses workspace-scoped data passes through an authorization layer that:

1. resolves the currently authenticated user from the session token;
2. loads the requested workspace;
3. verifies that the user is an **active member** of that specific workspace;
4. rejects the request otherwise, without disclosing whether the workspace exists.

Workspace membership is itself a first-class, revocable record. Removing a user from a workspace deactivates the membership, and inactive memberships are excluded from the membership check. All subsequent queries are scoped to the workspace resolved by this layer, so one customer cannot read, modify or even enumerate another customer's data.

6.3 Feature-level authorization

Beyond membership, individual capabilities are gated per workspace, such as which check types are enabled and whether billing features are available. Attempting to invoke a capability that is not enabled for the workspace is rejected by the API.

07 Application security

7.1 API layer

- **Input validation.** Every write operation passes through a validation layer that accepts only explicitly permitted fields, enforces required fields, validates formats such as email addresses, dates and enumerations, and enforces uniqueness constraints at the database level. Fields that are not explicitly allowed are discarded rather than silently persisted, which prevents mass-assignment.
- **SQL injection.** All database access goes through a query layer that generates parameterized queries. User input is never interpolated into SQL statements.
- **Schema-documented surface.** The API is described by an OpenAPI specification, which keeps request and response shapes explicit and reviewable.
- **Rate limiting.** Requests are rate-limited per source address at the API entry point, and clients exceeding the limit receive HTTP 429 responses. Rate limiting is evaluated against the real client address, resolved from the trusted reverse proxy.
- **Cross-origin requests.** Browser requests from other origins are restricted by an explicit allowlist containing only the official Stracket frontend.
- **Upload limits.** Request body sizes are capped, which bounds the resource cost of any single upload.
- **Error handling.** Detailed error messages and stack traces are disabled in production. API errors are returned as structured responses containing what the caller needs and nothing about the internals.

7.2 Frontend

- **Content Security Policy.** The application ships a strict Content Security Policy that denies everything by default and then permits only what the application genuinely needs. Scripts are allowed only from the application's own origin and must carry a **per-response nonce**. Plugin content and form submissions to third-party destinations are blocked outright, and the document base URI and framing ancestors are locked down. This substantially reduces the impact of any injected markup, since injected script has no valid nonce and will not execute.
- **Cross-site scripting.** Angular escapes interpolated values by default and treats values bound into the DOM as untrusted unless explicitly marked otherwise. This built-in contextual escaping, combined with the policy above, provides defence in depth against XSS.

- **Framework currency.** The frontend runs on a currently supported major release of Angular, and dependencies are updated regularly so that fixes for disclosed vulnerabilities are picked up rather than accumulating.
- **Server hardening.** The rendering server suppresses framework fingerprinting headers, serves static assets read-only, and applies cache headers that allow long-term caching only for immutable, content-hashed build artefacts.

7.3 Runtime isolation

Both the frontend and the API run as containerized releases built on minimal base images containing only the runtime libraries the application needs. A smaller image means a smaller attack surface and fewer packages to patch.

08 Data protection

8.1 What data the platform holds

CATEGORY	EXAMPLES	WHERE STORED
Customer account data	User name, email address, password hash, role, workspace membership	`PostgreSQL`
Subject data	Name, aliases, date of birth, gender, nationality, country, company name, registration number, incorporation date	`PostgreSQL`
Screening results	Sanctions and watchlist matches, adverse media findings, match assessments and final decisions	`PostgreSQL`
Documents	Subject assets, generated PDF reports, workspace logos	`Object storage`
Audit records	Who did what, to which subject, check or monitor, and when	`PostgreSQL`
Billing data	Invoice details, orders, credit bundles, bank transfer payment references	`PostgreSQL`

NO CARD DATA

Stracket does not store payment card data. The only payment method supported is bank transfer, of which the platform records a reference rather than any cardholder or account credential.

8.2 Data location

Structured data resides in the United States. Documents and generated reports reside in cloud object storage. Some screening operations necessarily transmit subject attributes to external data providers. Those flows are listed in section 9.

Customers with data residency requirements should review section 9 together with their own legal counsel, and address residency and transfer terms in their agreement with Stracket.

8.3 File storage

Documents are never served directly from the application, and the storage bucket is not publicly readable. Access is granted exclusively through **cryptographically signed, expiring URLs generated by the API**, which:

- are issued only after the caller's authentication and workspace membership have been verified;
- are valid for a short period, after which the link stops working and must be re-requested;
- cannot be forged without the signing key, which never leaves the server.

A leaked document link therefore expires on its own, and removing a user's workspace access removes their ability to obtain new links.

8.4 Audit trail

The platform maintains an append-only audit log covering the actions that matter for a KYC file: creating and updating subjects, creating, updating and completing checks, recording a final check result, webhook deliveries, and creating, updating, enabling, disabling and firing monitors.

Each audit record captures the event type, the acting user (both by reference and by name, so the record stays meaningful if the account is later removed), the affected subject, check or monitor, the owning workspace, contextual detail, and a timestamp. Audit records are exposed to customers per subject through the API, so a customer can reconstruct the full history of a file.

Records removed by a user are deactivated and excluded from all application queries, which preserves the referential integrity of historical checks and the audit trail. In a compliance product, a screening decision must remain explainable after the fact.

09 Sub-processors and third-party services

Stracket relies on a small number of third-party services to deliver the platform. Each is used for a single defined purpose, and only the data required for that purpose is transmitted. The current named list of sub-processors is provided to customers on request and under the terms of their agreement.

CATEGORY	PURPOSE	DATA TRANSMITTED
Hosting infrastructure	Production hosting, United States	All platform data, at rest
Object storage	Document and report storage	Uploaded documents, generated reports, logos
Screening data provider	Sanctions, PEP and watchlist data	Subject identifying attributes used as the search query
Web search provider	Search layer used for adverse media screening	Subject names and search keywords
Language model provider	Adverse media analysis	Retrieved article content
Public reference data	Public entity enrichment	Entity identifiers and names
Email delivery provider	Transactional email (login verification codes, password resets, monitor notifications)	Recipient email address, message content

10 Backup and recovery

The production database is backed up on a recurring schedule so that data can be restored after a failure. Backups are retained long enough to allow recovery from a point in time before an incident, rather than only from the most recent state.

The application layer is deployed as versioned, immutable build artefacts. A lost server can therefore be rebuilt from the last released version combined with a database restore, without manual reconstruction of the environment.

11 Contact

For security questions about this document, for vendor security assessments, or to report a suspected vulnerability, contact **info@conneqtid.com**.

Stracket is operated by:

Conneqtid VBA

Italiestraat 38 unit 9
Oranjestad, Aruba

Conneqtid Curaçao B.V.

Scharlooweg 102
Willemstad, Curaçao

Customers with a formal vendor assessment process may request additional technical detail under a mutual non-disclosure agreement.

A Appendix A: Technology stack summary

LAYER	TECHNOLOGY	SECURITY-RELEVANT CHARACTERISTICS
Frontend	`Angular, server-side rendered on Node.js`	Contextual output escaping by default, strict nonce-based Content Security Policy, framework fingerprinting suppressed
Reverse proxy	`nginx`	TLS termination, HTTP to HTTPS redirect, sole public entry point
API	`Elixir on Phoenix`	Process isolation and fault tolerance from the BEAM runtime, request-level authentication and authorization
Authentication	`Signed session tokens, adaptive password hashing, email one-time codes`	Revocable refresh tokens, adaptive password hashing, multi-factor login
Database	`PostgreSQL`	Parameterized queries, referential integrity and uniqueness enforced in-database, no public network exposure
Background jobs	`Persistent job queues and scheduler`	Durable, retryable, observable asynchronous processing
File storage	`Cloud object storage`	Private bucket, access only via short-lived signed URLs, encrypted at rest
Hosting	`Cloud infrastructure, United States`	Minimal exposed port surface, restricted administrative access, database not publicly reachable

This document describes the Stracket platform as of the date on the cover. It is reviewed and updated as the platform evolves. For the current version, or for answers to a specific vendor security questionnaire, please contact us using the details in section 11.