



Clixlogix

INTERIM SECURITY & PRODUCTION READINESS AUDIT

An anonymized Lovable-built music curation application

SOURCE REVIEW · NOT A PENETRATION TEST

Report ID CLX-LVA-2026-001

Version 0.9 · Internal draft

Assessment date 24 October 2026

Source revision 3f88854c...

Prepared by Clixlogix security review team

NOT FOR PUBLICATION UNTIL COORDINATED DISCLOSURE IS COMPLETE

01

Reading guide

How to interpret evidence, severity, and conditional conclusions

This report demonstrates how a client-ready audit can remain readable while preserving the technical evidence needed to support each conclusion. It is based on a pinned public source revision that has been anonymized for this sample.

IMPORTANT BOUNDARY

No request was sent to the original live application, database, Supabase project, Stripe or account, Spotify account. Source conditions are not presented as proof of a live breach.

Evidence labels

SOURCE-CONFIRMED	The relevant control flow or missing control is visible in the pinned source.
VERSION-CONFIRMED	The package and version are in the lockfile; practical reachability may still depend on the shipped platform.
CONDITIONAL	The source creates a credible failure path, but live deployment state or attacker preconditions were not tested.
NOT ASSESSED	The evidence was unavailable or intentionally excluded, so the report makes no assurance claim.

Severity is a release decision aid

High findings can cross a trust boundary, alter a connected account, or undermine the payment and entitlement model. Medium findings materially weaken confidentiality, integrity, availability, or reliable operation. Low findings are defense-in-depth improvements. Production-readiness gaps may block launch even where they are not independently exploitable vulnerabilities.

The snippets in this public-facing sample rename application identifiers while preserving the reviewed control flow. Exact paths, line references, hashes, and acceptance tests are retained in the companion remediation specification.

02

Executive conclusion

Production clearance is not supported for the pinned build

3

HIGH RELEASE
BLOCKERS

5

MEDIUM
FINDINGS

1

LOW
FINDING

1

CONDITIONAL
HIGH

The assessed revision contains strong beginnings, including table-level row security, server-side secret loading, an explicit user-token check in the payment function, and hardened database trigger search paths. Those controls do not close the application's highest-risk trust boundaries.

Two Spotify-facing functions combine elevated database access with a user identity supplied by the request body. The OAuth callback generates a state value but does not verify it before storing provider tokens. The payment path creates Checkout Sessions, yet the reviewed source contains no verified webhook or transactional fulfillment path and displays success on a public route.

RELEASE POSITION

Disable the affected connected-account functions or remediate them before onboarding users.
Do not represent payment as complete until a verified server-side event has been processed.

What this conclusion does not say

- It does not claim that the public repository is the exact source currently deployed.
- It does not claim that a live user account was accessed, a payment was bypassed, or a provider token was obtained.
- It does not assess dashboard-only grants, secrets, logs, backups, provider settings, or incident history.
- It does not convert a dependency advisory into an exploitability claim without checking the shipped platform and code path.

03

System and trust boundaries

The risk is concentrated where browser input reaches elevated server credentials

BROWSER	SUPABASE	EXTERNAL SERVICES
React / Vite UI Supabase session Public success route	Postgres + RLS Edge Functions Service-role client	Spotify OAuth + API Stripe Checkout

Primary trust-boundary observations

- The client's Supabase publishable key is expected to be public and is not reported as a secret leak.
- RLS protects direct table access, but a service-role client bypasses it. A function using that credential must recreate user, tenant, object, and property authorization itself.
- CORS limits which browser origins can read a response. It does not authenticate terminal clients, scripts, or other servers.
- OAuth state binds the callback to the browser flow that started it. Generating state without storing and verifying it provides no binding.
- A payment return URL is user experience, not fulfillment evidence. Entitlements must be derived from verified provider events and server-owned order state.

Data classes in scope

CREDENTIALS	Spotify access and refresh tokens; Supabase service-role key; Stripe secret key.
IDENTITY	Supabase user subject, provider account identity, stored provider-customer binding.
BUSINESS STATE	Imported songs, playlists, order quantity, payment status, fulfillment status.
AUDIT EVIDENCE	Event IDs, timestamps, correlation IDs, security logs, deployment configuration.

04

Issue register

Ten findings ordered by trust-boundary and release impact

ID	Finding	Risk	Evidence	Release position
01	Caller-controlled identity reaches elevated Spotify API	High	Source-confirmed	Block
02	OAuth state and account binding are not enforced	High	Source-confirmed	Block
03	Payment completion is not server-verified	High	Missing control	Block
04	Server pricing does not enforce selected quantity	Medium	Source-confirmed	Fix before launch
05	Provider bearer token exposed to frontend JavaScript	Medium	Source-confirmed	Fix before launch
06	Direct runtime dependencies contain known advisories	High*	Version-confirmed	Conditional block
07	External API work lacks explicit abuse controls	Medium	Missing control	Fix before scale
08	Core integration and release configuration incomplete	Medium	Source-confirmed	Block journey
09	Error responses can reveal internal details	Low	Source-confirmed	Harden
10	Release assurance is not automated	Medium	Missing control	Add release gate

* CLX-06 is high only where the vulnerable runtime package is included in a shipped, reachable platform. The reviewed archive contains Capacitor packages but no native bundles; the report therefore treats mobile exploitability as conditional.

DECISION RULE

Resolve CLX-01, CLX-02, and CLX-03 before onboarding users. Treat CLX-06 as a release block for any affected native build until patched and retested.

Risk concentration

The highest-risk conditions share one pattern: a client-controlled value crosses into a server context that holds more authority than the caller. The corrective principle is therefore consistent across the report. Verify the caller, derive identity from that verification, authorize the specific object and action, constrain elevated queries, and record the result.

CLX-01

HIGH

Caller-controlled identity reaches an elevated API

Evidence confidence: High in source; live state unverified

The Spotify API function is configured without gateway JWT verification, creates a Supabase service-role client, and accepts a user identifier from JSON. It then uses that identifier to load and refresh stored provider tokens and to perform provider operations. Because service-role access bypasses RLS, the request-body identifier becomes the effective authorization decision.

```
1  const adminDb = createClient(URL, SERVICE_ROLE_KEY)
2  const { action, accountId, payload } = await request.json()
3
4  const connection = await adminDb
5    .from("provider_connections")
6    .select("access_token, refresh_token, expires_at")
7    .eq("user_id", accountId)
8    .single()
9
10 if (action === "create_collection") {
11   return provider.createCollection(connection.access_token, payload)
12 }
```

Anonymized excerpt. Identifiers renamed; reviewed control flow preserved.

Why this matters

If a caller can reach the deployed function and learn or guess another application user's UUID, the source path does not show an independent check that the authenticated subject owns that identifier. The resulting capability could include reading the connected user's liked tracks, refreshing their token, or creating a playlist in their provider account. That is a source-confirmed trust failure; reachability was not tested against the live deployment.

05

CLX-01 remediation

Make verified identity the only source of user scope

Required control sequence

- Require a bearer token and verify it server-side before any database or provider call.
- Derive the application user ID from the verified token subject. Reject request-body identity fields.
- Prefer a user-scoped Supabase client where RLS can enforce ownership. If service-role access is necessary, constrain every elevated query to the verified subject and intended object.
- Allowlist operations and fields. Add request limits and per-user throttling before provider fan-out.

```
1  const token = readBearer(request)
2  const user = await verifySupabaseUser(token)
3  if (!user) return deny(401)
4
5  const input = parseAllowedOperation(await request.json())
6  const connection = await adminDb
7    .from("provider_connections")
8    .select("access_token, refresh_token, expires_at")
9    .eq("user_id", user.id)
10   .single()
11
12  return runAllowedOperation(user.id, connection, input)
```

Target pattern, simplified for explanation.

Closure evidence

Retest as signed-out, invalid-token, expired-token, owner, and cross-user callers. Confirm that failed authorization produces no database mutation, provider call, token refresh, or externally visible side effect. Logs should record the verified subject and correlation ID without storing provider credentials.

CLX-02

HIGH

OAuth state and account binding are not enforced

Evidence confidence: High in source; provider settings unverified

The authorization branch generates a state value and places it in the provider URL, but the assessed source does not persist that value. The callback accepts code, state, and a user identifier, checks only the code and user identifier, then stores provider credentials under the caller-supplied identifier with service-role access.

```
1  if (operation === "authorize") {
2    const state = crypto.randomUUID()
3    return json({ url: buildProviderUrl({ state }) })
4  }
5
6  const { code, state, accountId } = await request.json()
7  if (!code || !accountId) return badRequest()
8
9  const tokens = await exchangeCode(code)
10 await adminDb.from("provider_connections").upsert({
11   user_id: accountId,
12   access_token: tokens.access_token,
13   refresh_token: tokens.refresh_token
14 })
```

State is generated but not verified. Identifiers renamed.

The missing invariant

The callback must prove that the same authenticated application user started the OAuth flow, that the callback carries the exact unpredictable state issued for that flow, that the state is still valid and unused, and that the resulting provider identity is being attached to that user. A state string that is never checked provides no protection.

06

CLX-02 remediation

Use stored, expiring, single-use OAuth transactions

Recommended state record

Field	Purpose
state_hash	Store a hash of at least 128 bits of cryptographically random state.
user_id	Verified Supabase subject that initiated the flow.
expires_at	Short validity window; reject after expiry.
used_at	Enforce one-time use and prevent callback replay.
return_path	Allowlisted post-connection destination; never an arbitrary URL.

Callback sequence

- Hash the presented state and look up the unused transaction.
- Verify expiry and bind it to the authenticated session or other documented callback binding.
- Exchange the authorization code only after state validation.
- Attach the provider account to the stored user ID, never a body parameter.
- Consume the state and establish the connection atomically where possible.

RETEST

Missing, altered, expired, replayed, and user-mismatched state values must fail before or token exchange database write. Confirm failure has no side effect.

CLX-03

HIGH

Payment completion is not server-verified

Evidence confidence: High for missing source path

The Checkout-creation function performs a bearer-token user lookup, which is correct and worth preserving. The reviewed source nevertheless contains no verified webhook, signature check, event-id store, transactional entitlement update, or reversal handling. A public success route displays that payment and processing succeeded without querying server-owned order state.

```
1 // Browser route
2 export function PaymentSuccess() {
3   return <h1>Payment Successful!</h1>
4 }
5
6 // Checkout creation
7 return stripe.checkout.sessions.create({
8   success_url: `${request.headers.get("origin")}/payment-success`,
9   cancel_url: `${request.headers.get("origin")}/payment-cancelled`
10 })
```

Anonymized excerpt. A return page is not fulfillment evidence.

Required payment state machine

- Create a server-owned order before Checkout and place only its identifier in metadata.
- Verify Stripe webhook signatures against the raw request bytes.
- Map the verified customer to an application account using a server-owned binding.
- Record the event ID and apply the entitlement change in one transaction.
- Handle activation, renewal, downgrade, cancellation, failed payment, refund, and dispute.
- Render the return page from server-owned order state; show pending until fulfillment is confirmed.

CLX-04

MEDIUM

Server pricing does not enforce selected quantity

Evidence confidence: High

The client sends a playlist count. The Checkout line item remains one fixed USD 1.00 unit with quantity one, while the submitted count changes the description and metadata. The charge therefore does not prove that the amount corresponds to the represented work.

- Compute price and quantity from a server-owned catalogue and order record.
- Validate amount, currency, customer, and order before fulfillment.
- Treat client metadata as descriptive, never authoritative.

CLX-05

MEDIUM

Provider bearer token reaches frontend JavaScript

Evidence confidence: High

The browser directly selects the current user's provider access token and sends it to the provider API. RLS can prevent cross-user reads, but it cannot stop a compromised same-origin script, malicious extension, debug capture, or accidental client logging from reading the current user's credential.

```
1  const connection = await browserDb
2    .from("provider_connections")
3    .select("access_token")
4    .eq("user_id", currentUser.id)
5    .single()
6
7  await fetch(PROVIDER_API, {
8    headers: { Authorization: `Bearer ${connection.access_token}` }
9  })
```

Keep provider tokens server-side; expose narrow application operations.

CLX-06

HIGH · CONDITIONAL

Direct runtime dependencies contain known advisories

Evidence confidence: Version-confirmed

A production-only dependency audit on 24 October 2026 reported 15 advisories: one moderate, eleven high, and three critical. Direct packages included Capacitor mobile components at 7.4.2 and React Router DOM at 6.27.0. A critical Capacitor advisory affects versions below 7.6.9, but native bundles were not present in the reviewed archive.

Package	Reviewed version	Signal	Interpretation
@capacitor/android	7.4.2	Critical	Block affected Android release; web reachability not established.
@capacitor/ios	7.4.2	Critical	Block affected iOS release; native bundle absent from evidence.
react-router-dom	6.27.0	High	Upgrade and run route/auth regression tests.
zod	3.23.8	High	Fix may require a major upgrade and schema regression tests.

Remediation

Upgrade direct packages, recreate the lockfile, review breaking changes, and test every platform actually shipped. Retain an SBOM for the release. The final risk rating should be based on shipped artifacts and reachable code, not the advisory count alone.

EVIDENCE DISCIPLINE

A vulnerable version is evidence of exposure to an advisory. It is not by itself proof that the assessed deployment is exploitable.

CLX-07

MEDIUM

External API work lacks explicit abuse controls

Evidence confidence: High in source

The liked-songs path follows provider pagination until exhaustion. Playlist creation accepts an arbitrary track list and makes batched provider calls. The reviewed code contains no explicit request-size limit, maximum page count, operation budget, idempotency key, or per-user throttle.

- Bound body size, list length, page count, execution time, retries, and total provider calls.
- Use per-user and per-IP limits; queue long jobs and expose status.
- Make remote object creation idempotent and back off on provider rate limits.

CLX-08

MEDIUM

Core integration and release configuration are incomplete

Evidence confidence: High in pinned source

The reviewed client contains no sign-in, sign-up, auth-state, provider authorization, or callback wiring. Only the payment function is invoked from the browser. The payment function and project instructions also use a misspelled Stripe secret variable. The project builds, but the pinned source does not demonstrate an operable end-to-end user journey.

- Validate required environment variables at startup and standardize their names.
- Wire authentication and OAuth only after CLX-01 and CLX-02 are fixed.
- Add a staging smoke test covering sign-in, provider connection, import, payment, webhook fulfillment, and revocation.

CLX-09

LOW

Error responses can reveal internal details

Evidence confidence: High in source

Several function error paths return the caught exception message. Public responses should use stable error codes and safe messages; detailed exceptions belong in redacted server logs with correlation IDs.

CLX-10

MEDIUM

Release assurance is not automated

Evidence confidence: High in pinned source

The production build succeeded. Lint failed with six errors and seven warnings. No automated tests were present for authorization, OAuth state, payment fulfillment, RLS behavior, external API limits, or rollback behavior.

Positive measures observed

- RLS is enabled on the three application tables in the reviewed migrations.
- Ownership policies compare `auth.uid()` with `user_id` for intended row operations.
- The frontend uses a Supabase publishable key; no service-role key was found in client source.
- Edge Function secrets are loaded from environment variables rather than committed values.
- The payment function explicitly validates the bearer token before creating Checkout.
- SECURITY DEFINER trigger functions set an empty search path and use a qualified application table.
- The pinned source produces a successful Vite production build.

07

Remediation roadmap

Sequence work by containment, trust repair, and release assurance

Phase	Objective	Required outcomes
0 · Contain	Prevent exposure while fixing	Disable or undeploy affected connected-account functions; suppress unverified payment-success claims; rotate provider tokens if the code was live.
1 · Rebind trust	Authenticate and authorize	Verified subject replaces body identity; OAuth state is stored and single-use; provider tokens stay server-side.
2 · Repair money flow	Make fulfillment authoritative	Server-owned orders and pricing; verified webhooks; atomic idempotency; lifecycle reversals.
3 · Bound operation	Control abuse and failure	Rate limits, resource budgets, queues, retries, safe errors, structured logs, alerts.
4 · Prove release	Create repeatable assurance	Patched dependencies, test matrix, CI gates, staging retest, SBOM, backup and rollback evidence.

Suggested release gates

- No open CLX-01, CLX-02, or CLX-03 acceptance test.
- No unaccepted critical or applicable high runtime advisory in shipped artifacts.
- Authorization tests pass for signed-out, owner, cross-user, and elevated server paths.
- Payment duplicates, out-of-order events, cancellations, failed renewals, refunds, and disputes are handled.
- Build, lint, type checks, tests, dependency policy, and deployment smoke test pass in CI.

08

Retest and closure

A diff is necessary, but controlled staging evidence closes the finding

Evidence package required

SOURCE	Remediation commit, migrations, function configuration, lockfile, and generated SBOM.
DEPLOYMENT	Staging URLs and configuration exports or screenshots with secrets redacted.
TEST IDENTITIES	Owner and cross-user application accounts plus test provider and payment accounts.
OBSERVABILITY	Redacted logs showing verified subjects, event IDs, correlation IDs, and rejected cases.
AUTOMATION	Passing CI run containing authorization, OAuth, payment, dependency, and build gates.

Minimum adversarial matrix

Control	Required negative cases	No-side-effect evidence
Connected account	No token; invalid; expired; body-ID swap; cross-user object; unsupported action.	No token read/refresh, provider call, or database mutation.
OAuth callback	Missing, altered, expired, replayed, and user-mismatched state.	No code exchange and no connection write.
Payment	Forged return; bad signature; duplicate; stale event; cancellation; refund.	No incorrect grant, restoration, or duplicate credit.
Abuse controls	Oversized body; excessive pages; repeat creation; quota pressure.	Bounded calls, stable error, preserved shared capacity.

09

Scope, limitations, and source record

What was inspected and what remains unknown

ASSESSMENT METHOD	Static source review; SQL and function configuration review; local build, lint, and dependency checks.
PINNED REVISION	3f88854c4a7e1e6492608b6e172fe0f8cc0cfb94, dated 9 July 2026.
ARCHIVE SHA-256	cdfe0ba05b93ca20fb83053355adbc5250a859999a6dfd8abef3910ae239f08e
BUILD RESULT	Successful Vite production build; bundle-size warning above 500 kB.
LINT RESULT	Failed with six errors and seven warnings.
DEPENDENCY RESULT	Production-only audit: 15 advisories (1 moderate, 11 high, 3 critical) on 24 October 2026.

Not assessed

- The original live application, domains, mobile binaries, Supabase project, database grants, Edge Function deployment state, or secrets.
- Stripe products, prices, customers, webhooks, signatures, disputes, or production events.
- Spotify application settings, redirect URIs, scopes, tokens, user accounts, or production API behavior.
- Cloud logs, rate limits, backups, recovery tests, monitoring, incident history, or privacy compliance.
- Controls added after the pinned revision or maintained outside the repository.

REASSESSMENT TRIGGERS

Reassess after trust-boundary rewrites, payment architecture changes, native release new or creation, tenants roles, major dependency upgrades, or provider-scope expansion.

10

Use and disclosure

Educational sample; coordinated disclosure required before public attribution

This report is an anonymized educational example of how Clixlogix can present a source-backed production-readiness audit for a Lovable-built application. It is not a certification, guarantee, or representation that the software is free from vulnerabilities. Security conclusions are limited to the evidence, revision, date, and methods stated in the report.

The assessed project was selected from public source and reviewed without accessing its production systems. The repository identity and exact public excerpts should not be published until the maintainer has been contacted, the findings have been validated with them, and a reasonable remediation or disclosure window has elapsed.

PUBLICATION CONTROL

Keep this version internal. A future public case study should remove repository-identifying paths, retain the evidence boundary, and reflect the maintainer's response and remediation status.

Prepared by

Clixlogix
Security and production-readiness review
clixlogix.com

Companion artifact

The engineering remediation specification supplied with this sample contains exact internal paths, acceptance tests, closure evidence, and the recommended sequencing for each finding.

END OF REPORT