Same Code, Same Access, 3x the Findings

A whitebox penetration testing benchmark: Vector vs Aikido across ten production applications

zauth research

Abstract

We compared two AI penetration testing tools, Vector by zauth and Aikido, running both in whitebox mode with full source code access against ten production applications built on the Lovable platform. Both tools scanned the same ten targets. We normalized every finding to a single root cause, scored each under CVSS v3.1 with strict banding, and applied one identical adjudication rubric to both tools, including to Aikido's findings, which ship without CVSS vectors.

Under identical rules, Vector identified 227 distinct vulnerabilities to Aikido's 71, a 3.2x difference on the same code with the same access. Restricting to substantive findings and excluding low-severity configuration hygiene, the difference is 4.7x. On nine of the ten applications, Vector found critical or high severity flaws that Aikido did not report. The advantage is concentrated in breadth of coverage: Vector surfaces the long tail of authorization, validation, and access-control defects that comprehensive source analysis reveals, and that a less exhaustive pass leaves unreported.

This paper documents the methodology in full so the results are reproducible, including the CVSS vector we assigned to every Aikido finding from its published evidence, listed in Appendix B.

1. Introduction

Software is now built by people who do not write code. AI development platforms have collapsed the distance between an idea and a running application: a founder, a designer, or a student describes what they want in plain language, and a working product appears, deployed to a live URL, backed by a real database, accepting real users within hours. This is a genuine advance. It is also a security problem of a kind the industry has not had to reckon with before.

The people building these applications are not security engineers, and nothing in the tools they use requires them to think like one. Authentication, authorization, tenant isolation, input validation, and secret management are handled implicitly, generated by a model that optimizes for a working demo rather than a defensible system. The result reaches production without anyone having asked whether one user can read another's data, whether an unauthenticated request can reach a privileged function, or whether the client can be trusted with a decision the server should be making. The answer, across the applications we tested, is frequently that they

cannot.

Traditional security tooling does not reach these builders. A manual penetration test starts in the low thousands of dollars per application and takes weeks, priced and paced for enterprises with security teams and compliance obligations. The developer who assembled an app over a weekend and shipped it to a few hundred users is not going to commission one. The gap between how fast these applications are built and how slowly they can be secured is the gap this class of tooling exists to close.

Our prior benchmark, "Vibe-Coded and Vulnerable", established two things. First, that AI-generated applications carry serious, exploitable vulnerabilities at scale, not as occasional lapses but as a systematic property of how they are built. Second, that Vector's black-box scanner, operating with no source access at all, matched or exceeded a whitebox competitor that had every line of the code. That result raised an obvious question. Black-box testing is the harder position; the scanner sees only what an outside attacker sees. What happens when both tools are given the easier position, full source access, the advantage the industry markets as decisive?

This paper answers that question. We ran Vector in whitebox mode against ten production applications and ran Aikido, the whitebox pentesting platform Lovable integrated into its own product, against the same ten with the same access to the same code. Whitebox testing is the standard against which pentest tools are sold, and for good reason: source access is a real advantage, the most complete view of an application a tool can be given. This benchmark holds that advantage constant for both tools and asks what each does with it. Given the same source and the same access, the two produced results that differ by a factor of three to four. Source access is real. It is only worth what the scanner can turn it into.

---

2. Methodology

2.1 Targets

We selected the ten applications described above, each built on Lovable and backed by a Supabase stack. The ten targets are the same set used in our prior benchmark. Both tools scanned all ten.

All ten targets are fixed benchmark applications and were confirmed unchanged between the Aikido scans, dated March through May 2026, and the Vector scans, run July 10, 2026. Both tools analyzed identical code on every target.

2.2 Both tools run whitebox

Both tools were configured with full source code access. This is a like-for-like comparison of whitebox capability, not a comparison of black-box against white-box.

2.3 Normalization

Findings were normalized to root cause before counting. Where multiple reported findings share a single underlying cause and a single remediation, they collapse to one normalized finding. For example, a disabled row-level-security policy that exposes several database tables is one misconfiguration with one fix, counted once, regardless of how many tables it exposes. Informational findings are excluded entirely. Findings reported by both tools are counted once and attributed to both. Every normalization is logged so the raw-to-normalized mapping is auditable.

2.4 Scoring

Every finding was scored under CVSS v3.1. The numeric score is computed from the vector; the severity band is derived from the score, using strict thresholds: Critical at 9.0 and above, High from 7.0 to 8.9, Medium from 4.0 to 6.9, Low from 0.1 to 3.9. Scores and bands are not assigned independently of the vector at any point.

2.5 Adjudicating Aikido's findings

Aikido's reports do not contain CVSS vectors or numeric scores. They ship proprietary severity labels only. To place both tools on one scale, we assigned a CVSS v3.1 vector to each of Aikido's 71 findings from its own description, business-impact, and reproduction text, using the same scoring conventions applied to Vector's findings:

Unauthenticated cross-user data reads and mass enumeration were scored High. IDOR reads of private data were scored High. IDOR writes to non-critical fields were scored High. AI credit-drain, rate-limiting, brute-force, weak-password, and verbose-error findings were scored Medium. Transport cipher, security-header, and cookie-attribute findings were scored Low.

The vector assigned to each Aikido finding is published in Appendix B so any reader can verify the adjudication against Aikido's own evidence.

2.6 Fund-movement rule

One rule applies uniformly to both tools. A finding that allows an unauthenticated actor to move or manipulate funds, including payouts, balances, wagers, and treasury operations, carries a high availability impact for fund integrity and bands as Critical. This rule promoted four Vector findings and two Aikido findings, all on the same betting-platform target. It is applied identically to both tools; the critical count of each tool depends on where this threshold sits, but the threshold is the same for both.

---

3. Results

3.1 Headline comparison

Both tools, whitebox, same ten applications, identical scoring:

	Critical	High	Medium	Low	Total findings
Vector (whitebox)	18	45	147	17	227
Aikido (whitebox)	4	13	28	26	71
Ratio	**4.5x**	**3.5x**	**5.3x**		**3.2x**

Vector found 3.2x the total vulnerabilities on the same code with the same source access.

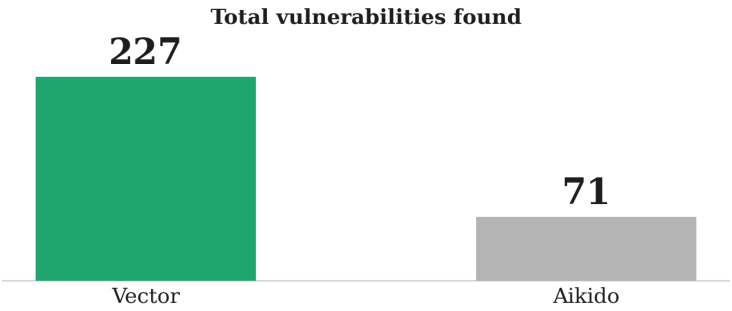
The gap is not confined to one severity band. Vector reported more findings than Aikido at every level: more than four times the criticals, more than three times the highs, and more than five times the mediums.

3.2 Substantive findings

Of Aikido's 71 findings, 26 are low-severity configuration hygiene: transport cipher suites, security headers, and cookie attributes. These are findings of a live-infrastructure probe, not of source analysis, and including them in the total works in Aikido's favor. Excluding the low band and comparing only substantive findings, Critical through Medium:

	Substantive findings (C + H + M)
Vector (whitebox)	210
Aikido (whitebox)	45
Ratio	**4.7x**

On the findings that represent exploitable application risk rather than configuration baseline, the gap widens to 4.7x.

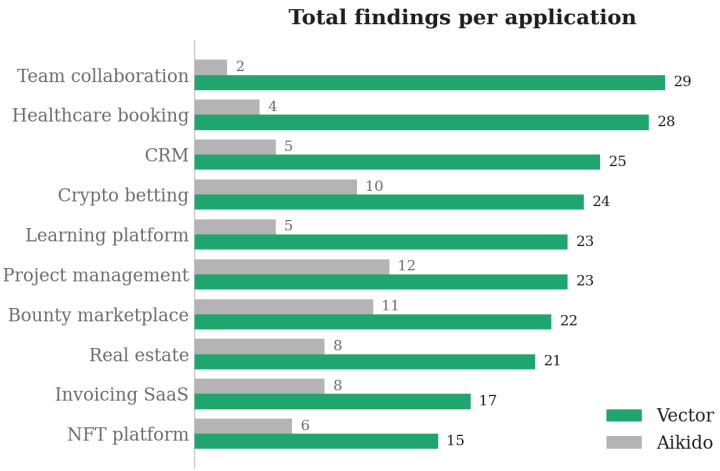


Total vulnerabilities found by each tool, whitebox mode, across the same ten applications.

3.3 Per-target breakdown

Target	Vector (C/H/M/L)	Aikido (C/H/M/L)
Bounty marketplace	0 / 7 / 13 / 2	0 / 3 / 4 / 4
Real estate	1 / 2 / 17 / 1	0 / 1 / 4 / 3
Project management	0 / 6 / 17 / 0	0 / 3 / 6 / 3
Invoicing SaaS	2 / 3 / 11 / 1	1 / 2 / 2 / 3
NFT platform	0 / 0 / 13 / 2	0 / 1 / 2 / 3
Learning platform	0 / 3 / 20 / 0	0 / 1 / 2 / 2
Healthcare booking	2 / 8 / 16 / 2	0 / 0 / 1 / 3
Team collaboration	2 / 6 / 16 / 5	0 / 0 / 2 / 0
Crypto betting	11 / 4 / 7 / 2	3 / 2 / 2 / 3
CRM	0 / 6 / 17 / 2	0 / 0 / 3 / 2
Total	**18 / 45 / 147 / 17**	**4 / 13 / 28 / 26**

Vector matched or exceeded Aikido's finding count in every band on every target.



Total findings per application. The gap holds across all ten targets, not one or two outliers.

---

4. Representative findings Aikido did not report

This section gives one detailed example and then a summary of nine more, one per application. In several cases Aikido was reading the exact file that carried the flaw and reported a lesser issue in the same function while missing the one that mattered.

4.1 A worked example: custodial wallet keys left in the open

The crypto betting platform holds custody of user funds. It generates a Solana wallet for every user and stores that wallet's private key in its database, so the safety of every balance rests on the keys being encrypted properly and unreadable by strangers. Both assumptions failed, in the same file. The wallet is created by an edge function that encrypts the private key like this:

```
encoder.encode(masterKey.padEnd(32, '0').slice(0, 32))
```

If the encryption key is shorter than 32 characters, the code silently pads it with zeros, turning a short secret into a mostly known key. The encrypted result is written to a profiles table that carries this rule:

```
CREATE POLICY "Profiles are publicly readable" ON public.profiles FOR SELECT
USING (true);
```

USING (true) makes every row readable by anyone holding the public API key, which ships in the browser. Any visitor can therefore read every user's encrypted Solana private key, and the only thing protecting those keys is the padded value above. Aikido's own report describes a race condition inside this very function, so its testers were reading it. They did not report the padded key or the public exposure of the keys the function produces.

4.2 Nine more, one per application

Each row below is the single clearest vulnerability Vector reported and Aikido did not, with the exact location in source shown as evidence.

Application	Vulnerability Aikido did not report	Severity	Evidence in source
Team collaboration	Account password is the user's public wallet address, full account takeover	Critical	`password: wallet_${walletAddress}_${chain}`
Healthcare booking	Any doctor can reassign a medical record to another patient	Critical	`medical_records UPDATE checks doctor_id, not patient_id`
Invoicing SaaS	has_role() ignores company, grants cross-tenant admin	Critical	`WHERE user_id = _user_id AND role = _role (no company)`
Bounty marketplace	Any user can insert themselves as project admin	High	`self-referential NOT EXISTS in admin INSERT policy`
Real estate	Storage "admins" policy checks only the bucket name	High	`WITH CHECK (bucket_id = 'property-images')`
Learning platform	Instructor-controlled iframe source, stored XSS on students	High	`<iframe src={lesson.video_url} />`
CRM	Server fetches a user-stored URL, server-side request forgery	High	`fetch(sub.endpoint) with no validation`
Project management	Unauthenticated endpoint returns full user rows	High	`service-role.from("marketplace_gigs").select("*")`
NFT platform	Admin panel mounted with no authentication guard	Medium	`<Route path="/admin" element={<Admin/>} />`

Table 3. One representative missed finding per application. Every item was confirmed against the application's source and against Aikido's own finding list.

5. Analysis

5.1 Where the advantage comes from

The gap is not Vector reporting low-severity noise. Vector found more at every level that matters. On the same code, it surfaced more than four times the critical vulnerabilities Aikido reported and more than three times the high-severity ones. Beneath those, it mapped the full population of authorization gaps, missing server-side validation, insecure direct object references, and access-control defects present across every table, column, edge function, and policy in the codebase, where Aikido reported a fraction. On the invoicing application alone, Vector reported eleven distinct medium-severity authorization and validation defects where Aikido reported two. Source access put every line of the code in front of both tools. Finding what is exploitable in it is a different problem, and it is the one the numbers measure.

5.2 The range of what Vector found

Vector's findings run the full span of application security failures. It surfaced broken access control, row-level-security policies that were absent, permissive, or missing a tenant filter; missing authentication on privileged endpoints that performed real actions without checking the caller; insecure direct object references that let a client select records belonging to other users; database keys and secrets exposed to the browser; injection and validation gaps where unchecked input reached a query or a sink; and the transport and header hygiene that fills the low-severity band. Surfacing all of it meant examining every table, every function, and every policy in each codebase rather than a sample of them, which is the difference the count reflects.

5.3 The whitebox advantage, realized

Source access is supposed to buy exactly this breadth. A tool with the code can enumerate every database policy and test each one, read every edge function and check its authentication, and trace every client-supplied identifier to the query it feeds. Whether a scanner actually does this is what separates the two results. Vector's coverage reflects an exhaustive pass over the source; Aikido's reflects a partial one. Both had the same material. One reported what was in it, and the other reported a fraction.

5.4 Case study: cross-tenant workspace takeover

Our prior benchmark identified the invoicing application as the single target where the competitor's whitebox scan found a critical that Vector's blackbox scan did not: a cross-tenant workspace takeover via a mutable `profiles.company_id` field, scored 9.1, reachable only with source access. In this benchmark, Vector's whitebox scan independently identified that exact vulnerability, scored 9.9. It also found a second critical on the same target that Aikido did not report: a `has_role()` security-definer function missing its company_id filter, letting administrative privilege bleed across tenants. The one finding that source access previously gave the competitor is now found by both tools, and Vector finds more around it.

6. What the results mean

The headline is a comparison between two tools, but the finding underneath it is about the applications. Ten production systems contained 227 distinct vulnerabilities between them, including exposed credentials, unauthenticated administrative access, cross-tenant data exposure, and, in a healthcare system, the ability to rewrite another patient's records. None of it was planted. These are the systems people are shipping, and the people shipping them are not equipped to find flaws of this severity on their own.

That is the gap this class of tooling exists to close, and it has to close it where the application actually lives: at the speed and price a builder can absorb, without a six-week engagement. For that tooling, the question that decides its worth is whether it surfaces everything an application is exposing or only a fraction of it. The three-to-four times gap in this benchmark is a measure of exactly that difference.

Source access remains a genuine advantage, and whitebox testing remains the right posture for assessing an application whose code you hold. This benchmark does not argue otherwise. It argues that the advantage is inert without a scanner that exhausts it, and that on the same code, with the same access, exhausting it is what produced three times the findings.

7. Limitations

This benchmark covers ten applications on a single development platform and backend stack. Results may differ on other stacks.

Aikido's findings were adjudicated to CVSS from their published evidence rather than from vectors Aikido provided, because Aikido does not publish vectors. We assigned those vectors conservatively and publish every one in Appendix B; a reader applying a stricter or looser threshold may reach a different absolute critical count for either tool, but the relative comparison holds across thresholds because the same threshold is applied to both.

Configuration-hygiene findings, transport ciphers and security headers, are surfaced by live-infrastructure probing rather than source analysis. Aikido reports these and Vector's whitebox scan does not, which is why the two tools' low-severity bands are not directly comparable and why we report a substantive-findings comparison alongside the total.

These figures reflect a stricter and fully documented CVSS adjudication of the competitor's findings than our prior benchmark. Where the two papers differ on the competitor's absolute severity counts, the per-finding adjudication in Appendix B is the reconciling reference: every score there is derived from Aikido's own published evidence and can be recomputed independently.

8. Conclusion

Given the same source code and the same access, Vector found roughly three times the vulnerabilities of a leading whitebox competitor, and nearly five times as many when restricted to substantive application risk. Whitebox testing works, and full source access is exactly the advantage it is sold as. But an advantage is only worth what the tool can do with it. The access was identical; the coverage was not, and coverage is what this benchmark measures.

Appendix A: Vector normalized findings by target

Normalized finding (representative title per root-cause cluster), severity band, and CVSS v3.1 base score. Full vectors are published in the individual Vector scan reports.

Bounty marketplace

Finding	Severity	CVSS
Stored XSS via javascript: URI confirmed in browser - clickable link in pending submissions	High	8.7
LLM system-prompt injection via unconstrained role field in ai_messages INSERT	High	8.5
Any authenticated user can self-promote to admin on a memberless/orphaned project	High	8.2
USING(true) RLS policy on bounty_payments exposes all wallet addresses and tx signatures	High	7.7
Any project member (contributor role) can inflate task bounty_amount and bypass review	High	7.7
Global profile enumeration: all user wallet addresses and email_verified status exposed	High	7.7
Admin/manager can submit and self-approve their own work - no self-review guard	High	7.7
No server-side validation on bounty_payments (amount positivity, submission linkage)	Medium	6.5
Submission re-approval double-spend: no state-machine guard + no bounty_payment	Medium	6.5
No UNIQUE(task_id, contributor_id) constraint on submissions - unlimited re-submission	Medium	5.4
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
CORS Wildcard Origin on Auth Endpoint	Medium	5.3
SECURITY DEFINER RPCs (has_role, is_project_member, project_role) allow any	Medium	5.0
Activity log forgery: any authenticated user can inject fake global audit entries	Medium	5.0
No server-side non-negative / upper-bound check on projects.budget_tokens	Medium	4.3
Unbounded comment body - no server-side length constraint	Medium	4.3
No length or format constraints on profiles.display_name / avatar_url / wallet_address	Medium	4.3
Bounty payment tx_signature generated with Math.random()	Medium	4.3
Client-side-only one-active-claim gate bypassed via direct API - contributor holds	Medium	4.3
Email verification bypass via self-PATCH of profiles.email_verified=true	Medium	4.3
Unfiltered exception message returned to client on 500 errors	Low	3.1
AI rate-limit TOCTOU: concurrent requests all pass the quota check before any usage event is	Low	3.1

Real estate

Finding	Severity	CVSS
Open redirect in email verification flow via unvalidated emailRedirectTo	Critical	9.3
Any authenticated user can upload arbitrary files to public property-images storage bucket	High	8.5
Public profiles table leaks registered user data to unauthenticated callers	High	7.5
Edge Function`chat` Callable Without User Authentication	Medium	6.5
CORS wildcard on unauthenticated AI chat endpoint - any origin can burn API credits	Medium	6.5
CSV formula injection via chat endpoint saveLeadData - malicious formulas inserted into	Medium	6.1
Vite CVE-2025-62522: server.fs.deny bypass via backslash on Windows - dev server	Medium	5.3
RLS Disabled - Anon Key Reads All `properties` Rows	Medium	5.3
has_role() SECURITY DEFINER RPC accepts arbitrary _user_id - unauthenticated admin	Medium	5.3
Weak Password Policy - 6-Char Password With No Complexity Accepted	Medium	5.3
Email Verification Disabled - Accounts Active Without Confirmation	Medium	5.3
Anonymous leads INSERT accepts attacker-controlled fields (status, source, property_id)	Medium	5.3
LeadForm: email format and field length validated client-side only; trivially bypassed	Medium	5.3
Internal environment-variable name disclosed in error response	Medium	5.3
No rate limiting on unauthenticated chat endpoint - unlimited AI credit consumption	Medium	5.3
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
PostgREST filter injection via unsanitized marketplace search parameter - tautology	Medium	5.3
Prompt injection extracts complete AI system prompt including all internal property	Medium	5.3
user_favorites INSERT policy does not enforce property active status - users can favorite	Medium	4.3
Admin route guarded only by client-side React state - isAdmin flag readable	Medium	4.3
No server-side lower-bound constraint on property price; integer fields accept negative	Low	2.7

Project management

Finding	Severity	CVSS
Stored XSS: javascript: URI in Notification link Field Rendered as Clickable Anchor	High	8.7
KanbanBoard and CoPilotChat rendered unconditionally on auth:no route, leaking task data	High	8.6
IDOR: All Authenticated Users Can Enumerate Private Draft Projects of Other Users	High	7.7
BOLA: Any Authenticated User Can Insert/Modify Tasks in Any Project (No Ownership Check	High	7.7
Unauthenticated Access to Marketplace Data via ai-match Edge Function (Service Role Key	High	7.5
Unauthenticated IDOR: Full Private Project Data + Tasks Exposed via project-copilot Service	High	7.5
Prompt injection via unvalidated message role field	Medium	5.3
No per-element type or size validation on userSkills array	Medium	5.3
Raw exception messages returned to callers in all edge function error handlers	Medium	5.3
Wildcard CORS (Access-Control-Allow-Origin: *) on All Unauthenticated Edge Functions	Medium	5.3
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
Unauthenticated AI Gateway Proxy - Credit Drain via global-chat Edge Function	Medium	5.3
No Rate Limiting on Login Endpoint - Credential Brute-Force Possible	Medium	5.3
Weak Password Policy - 6-Char Password With No Complexity Accepted	Medium	5.3
Email Verification Disabled - Accounts Active Without Confirmation	Medium	5.3
Unbounded `position` integer written to tasks with no range constraint	Medium	5.0
event_participants SELECT USING(true): any authenticated user can enumerate all	Medium	5.0
Identity Spoofing: Arbitrary user_name in Marketplace Gigs Accepted Without Validation	Medium	5.0
Admin Panel Protected by Client-Side Role Check Only - No Server-Side Enforcement	Medium	5.0
max_participants Accepts Invalid Values (Zero, Negative, Unbounded) - No DB CHECK	Medium	4.3
Event Capacity Limit Enforced Client-Side Only - Direct API Allows Over-Registration	Medium	4.3
Missing Role Restriction on Event Creation: Any Student Can Create Fake Hackathons	Medium	4.3
Project Owner Can Self-Approve Own Project, Bypassing Admin Workflow	Medium	4.3

Invoicing SaaS

Finding	Severity	CVSS
Mass Assignment on profiles.company_id enables full cross-tenant takeover	Critical	9.9
has_role() security-definer function lacks company_id filter - admin privilege bleeds across	Critical	9.6
Stored XSS via invoice pdf_url field: javascript: URI in Download PDF link exfiltrates session	High	8.7
Payments can be recorded with arbitrary tx_signature and amount - no on-chain verification;	High	7.7
Unauthenticated send-invoice-email: any caller can exfiltrate any invoice	High	7.5
Client RLS bypass: invoice client can zero totals and forge payment status without making	Medium	6.5
Unauthenticated process-invoice-email-retries: any caller can trigger bulk email retry	Medium	6.5
No server-side payment verification: arbitrary tx_signature and amount accepted,	Medium	6.5
Client-side computed invoice totals trusted by server - payment amount derived from	Medium	6.5
invoice_items mutable after invoice creation with no status gate - retroactive financial	Medium	6.5
No Rate Limiting on Login Endpoint - Credential Brute-Force Possible	Medium	5.3
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
Missing company-membership check: client-role callers bypass storage RLS	Medium	5.0
Authenticated company member can insert delivery-log rows with arbitrary	Medium	4.3
Direct PostgREST insert bypasses 4,000-character chat message limit	Medium	4.3
System prompt injection via direct PostgREST chat_messages INSERT with role='system'	Medium	4.3
Company name accepts empty string - no server-side minimum length constraint	Low	2.7

NFT platform

Finding	Severity	CVSS
Vite dev server bound to all interfaces enables CVE-2026-39365 path traversal	Medium	6.5
DOM XSS: dangerouslySetInnerHTML in ChartStyle built from unencoded caller props	Medium	6.1
Create Page Fully Accessible and Submittable Without Wallet Connection	Medium	5.3
Admin Panel Fully Accessible Without Any Authentication	Medium	5.3
Admin Link Unconditionally Rendered in Global Navbar for All Unauthenticated Visitors	Medium	5.3
Uncontrolled title field: value silently dropped, empty title never rejected	Medium	5.3
build:dev script enables componentTagger, embedding source file paths in production	Medium	5.3
robots.txt Has No Disallow - Admin Panel Indexed by All Search Engines	Medium	5.3
Internal Lovable Platform Project UUIDs and Commit SHA Exposed in Public HTML	Medium	5.3
Lovable Platform Session Token Exposed in localStorage Without HttpOnly Protection	Medium	5.3
Dashboard Wallet Gate Is UI-Only - No Server-Side Enforcement or Redirect	Medium	5.3
autoConnect:true Silently Re-Establishes Wallet Connection Without User Gesture	Medium	4.7
Vite dev server serves raw TypeScript source files at literal /src/** paths	Medium	4.3
Missing Content-Security-Policy Header	Low	3.4
TOCTOU race on wallet-switch: stale SOL balance from prior wallet displayed for current	Low	2.6

Learning platform

Finding	Severity	CVSS
Stored XSS: instructor-controlled lesson.video_url injected into unsandboxed<iframe src>	High	8.7
Courses UPDATE policy has no WITH CHECK - instructor can reassign instructor_id to any	High	7.7
RLS Disabled - Anon Key Reads All `profiles` Rows (Sensitive Columns Exposed)	High	7.5
NULL profile wallet silently skips the wallet-ownership guard in verify-nft-certificate	Medium	6.5
quiz_attempts: Client-Controlled score and passed Fields - No Server-Side Grading	Medium	6.5
Certificate Self-Issuance: No Course Completion or Enrollment Check in RLS	Medium	6.5
lesson_progress INSERT Has No Enrollment Check - Fake Course Completion Without	Medium	6.5
Profile wallet_address Accepts Arbitrary Value Without Proof-of-Ownership	Medium	6.5
CVE-2026-39365: Vite dev server bound to all interfaces enables network-reachable path	Medium	5.3
chat-assistant: unbounded JSON body parsed before rate-limit check; no messages array	Medium	5.3
Internal env-var name leaked in 500 response when LOVABLE_API_KEY is not set	Medium	5.3
Chat-Assistant Rate Limit Bypass via X-Forwarded-For Header Rotation	Medium	5.3
Live Supabase Anon JWT Committed to Repository (.env not in .gitignore)	Medium	5.3
Chat-Assistant Edge Function Requires No Authentication - Unauthenticated AI API Access	Medium	5.3
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
No Rate Limiting on Login Endpoint - Credential Brute-Force Possible	Medium	5.3
Student can self-mark NFT as on-chain verified via direct PostgREST PATCH	Medium	4.3
Enrollment UPDATE has no WITH CHECK - student can rebind course_id to any course	Medium	4.3
lesson_progress UPDATE RLS missing WITH CHECK - student can transfer/forge	Medium	4.3
quiz_questions SELECT exposes correct_index to enrolled students - full answer key	Medium	4.3
claim_attempts INSERT has no FK constraint or ownership check on certificate_id, allowing	Medium	4.3
Enrollments INSERT Has No Published Check - Self-Enrollment in Draft Courses	Medium	4.3
Raw exception messages (including internal RPC error details) returned in HTTP 500 body	Medium	4.3

Healthcare booking

Finding	Severity	CVSS
IDOR: Doctor can create medical records for any appointment/patient they did not treat,	Critical	9.6
Medical record update policy has no column restriction - doctor can reassign patient_id	Critical	9.6
IDOR: Doctor can forge NFT certificates for appointments they never conducted - INSERT	High	8.5
IDOR: Doctor can reassign NFT certificate patient_id to any arbitrary user	High	8.5
Session JWT + Refresh Token Stored in localStorage - Vulnerable to XSS Token Theft	High	8.0
Any doctor can upload files to any path in medical-attachments - no path/patient ownership	High	7.7
IDOR: Attachment file_path not bound to caller's uploaded objects - cross-patient file	High	7.7
Wrong user promoted to doctor: full_name ILIKE search used instead of email lookup	High	7.7
IDOR: Doctor permanently retains read access to former patient profiles after the therapeu-tic	High	7.7
nft_certificates.status is unconstrained free text - any string accepted on INSERT and PATCH	High	7.7
Patient Can Self-Complete Appointments via Unrestricted PATCH - No Column-Level	Medium	6.5
Patient Can Insert Appointment with Arbitrary Status - No Server-side Status Validation	Medium	6.5
Any doctor can delete any file from the medical-attachments private storage bucket	Medium	6.3
Missing Content-Security-Policy and X-Frame-Options Headers	Medium	6.1
Raw Error.message returned to client in catch-all handler	Medium	5.3
Avatars bucket public-read policy has no `to authenticated` - unauthenticated users read	Medium	5.3
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
Unauthenticated Access to AI Edge Function - Free Credit Drain	Medium	5.3
Rate-Limit Bypass via Forged JWT Sub Claim - Unverified JWT Decode	Medium	5.3
No Rate Limiting on Login Endpoint - Credential Brute-Force Possible	Medium	5.3
Business-logic: overlapping availability slots enable double-booking within same	Medium	4.3
availability_exceptions PATCH with check only on doctor_id - doctor can flip is_available	Medium	4.3
No server-side lower-bound on consultation_fee or years_experience in doctors table	Medium	4.3
Mass Assignment: Users Can Overwrite created_at Timestamp and Store Invalid Wallet	Medium	4.3
Missing end_time > start_time constraint on availability_exceptions + nullable times when	Medium	4.3
slot_duration_minutes ≤ 0 causes infinite loop in every patient's browser	Medium	4.1
nft_certificates published to Realtime with REPLICA IDENTITY FULL - full row (metadata_uri,	Low	3.5
ProtectedRoute null-role bypass: authenticated user with role=null skips allow-list	Low	3.1

Team collaboration

Finding	Severity	CVSS
Wallet-Auth: Deterministic Password Enables Full Account Takeover Without Signature	Critical	10.0
Open redirect / auth-token exfiltration via attacker-controlled emailRedirectTo derived	Critical	9.3
IDOR: Any Authenticated User Can Self-Join Private Channels and Read All Messages	High	8.5
IDOR: messages UPDATE policy lacks channel-membership check - users can edit messages	High	7.7
All User Profiles Readable by Any Authenticated User - Mass Data Enumeration	High	7.7
Stored XSS via SVG Upload to Public Avatars Bucket - No MIME Type Restriction	High	7.6
IDOR: Cross-Tenant Task Injection - Any Authenticated User Can Assign Tasks to Other Users'	High	7.4
AI Chat Endpoint Accepts User-Controlled system Role Messages - Prompt Injection	High	7.3
Privilege Escalation: Admin Can Create Invitations with super_admin Role	Medium	6.8
Latent DOM XSS via dangerouslySetInnerHTML in ChartStyle with unsanitized config	Medium	5.4
No Rate Limiting on Login Endpoint - Credential Brute-Force Possible	Medium	5.3
Weak Password Policy - 6-Char Password With No Complexity Accepted	Medium	5.3
Email Verification Disabled - Accounts Active Without Confirmation	Medium	5.3
Unbounded `context` and `tone` fields forwarded to AI gateway without size validation	Medium	5.3
Unbounded `data` field and unvalidated`type` key forwarded to AI gateway	Medium	5.3
Internal error messages from Supabase admin API forwarded to client	Medium	5.3
No Rate Limiting on Unauthenticated AI Endpoints - Unbounded Credit Consumption	Medium	5.3
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
Notifications INSERT WITH CHECK (true): any authenticated user can spoof notifications	Medium	5.0
Arbitrary External URL Accepted in avatar_url - IP Tracking of All Authenticated Users	Medium	5.0
Deals INSERT allows cross-ownership contact_id/company_id references causing silent	Medium	4.9
IDOR: contacts INSERT does not verify that company_id is owned by the caller - cross-user	Medium	4.3
messages.content: no server-side length constraint allows unbounded message payloads	Medium	4.3
Deals Table Accepts Negative Values - No Server-Side Bounds Check	Medium	4.3
Team invitation expiry not enforced server-side; duplicate invitations not prevented	Low	3.8
Activities INSERT allows cross-ownership FK references enabling cascade-delete of another	Low	3.5
wallet-auth: listUsers() called without pagination - returning users locked out in large	Low	3.5
Unrestricted File Upload to Files Bucket - Malware/Phishing Payload Delivery via Signed URLs	Low	3.5
PostgREST filter-string injection via CRM contact search	Low	3.1

Crypto betting

Finding	Severity	CVSS
COMPLETE EXPLOIT CHAIN: Unauthenticated Treasury SOL Drain via Bet Fabrication +	Critical	10.0
IDOR: Unauthenticated Actor Can Set Any User's Balance to Arbitrary Value	Critical	10.0
Unauthenticated payout-winners and close-expired-markets Edge Functions Accessible	Critical	10.0
IDOR: Any Unauthenticated User Can Resolve Any Market to Any Outcome	Critical	10.0
IDOR: Any User Can Forge Payout Records - Status and tx_signature Writable Without Auth	Critical	10.0
Unauthenticated Admin Account Creation with Hardcoded Credentials	Critical	10.0
JWT verified via local getClaims() decode instead of server-side getUser(), enabling	Critical	10.0
Supabase Anon Key and Project URL Committed to Repository (.env)	Critical	10.0
Business Logic: Bets Accepted on Resolved Markets - Guaranteed Risk-Free Winning Bets	Critical	10.0
RLS Disabled - Anon Key Reads All `chat_messages` Rows (Sensitive Columns Exposed)	Critical	9.3
Mass Assignment: Bets INSERT Accepts Arbitrary wallet_address and time_weight=9.9999	Critical	9.1
give_tokens Accepts Negative Amount - Admin Can Drain Any User's Balance	High	8.7
AES-GCM encryption key zero-padded when shorter than 32 bytes, weakening custodial	High	7.7
TOCTOU double-spend: unauthenticated concurrent payout triggers send real SOL twice	High	7.5
Profiles Table Publicly Readable Including Encrypted Solana Private Keys	High	7.5
Wildcard CORS (Access-Control-Allow-Origin: *) on All Edge Functions Including Admin	Medium	6.1
Edge Function`ai-assistant` Callable Without User Authentication	Medium	5.3
Weak Password Policy - 6-Char Password With No Complexity Accepted	Medium	5.3
Email Verification Disabled - Accounts Active Without Confirmation	Medium	5.3
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
Missing Content-Security-Policy Header	Medium	4.7
Market `duration` has no server-side bounds - zero, negative, or multi-decade markets	Medium	4.3
Math.random() used to decide financial market outcomes, enabling manipulation	Low	3.7
Realtime filter injection: unsanitised URL param interpolated into PostgREST filter string	Low	3.1

CRM

Finding	Severity	CVSS
TOCTOU Race in First-Signup Admin Bootstrap Trigger Allows Two Concurrent Registrations	High	8.7
leads RLS allows arbitrary client_user_id / assigned_to values - privilege escalation	High	7.7
Tasks RLS does not restrict assigned_to/created_by on INSERT/UPDATE and allows client role	High	7.7
Over-Broad RLS Policy Exposes All Profiles (including wallet_address) to Every sales_rep	High	7.7
IDOR: interactions UPDATE policy does not validate the new lead_id - any interaction can be	High	7.7
Stored SSRF: Attacker-Controlled Push Endpoint URL Fetched Server-Side	High	7.7
Client role can create or modify deal amounts on their linked lead (insufficient RLS)	Medium	6.5
Critical DB Misconfiguration: SECURITY DEFINER Functions Lack EXECUTE Grant - Entire	Medium	6.5
CVE-2026-53571: Vite dev server bound to all interfaces with no server.fs.deny, enabling	Medium	5.6
JWT identity accepted from local decode (getClaims) - no server-side signature	Medium	5.3
No Rate Limiting on Unauthenticated send-push-reminders: Unlimited Service-Role DB	Medium	5.3
No Rate Limiting on Login Endpoint - Credential Brute-Force Possible	Medium	5.3
Unauthenticated Endpoint Leaks Aggregate Internal Task & Subscription Statistics	Medium	5.3
Verbose PostgREST Error Messages Leak Internal Schema Details	Medium	5.3
Interaction INSERT: created_by not restricted to auth.uid() - client role can forge	Medium	5.0
task_reminder_state INSERT Enables Cross-Tenant Task UUID Enumeration via FK Error	Medium	5.0
IDOR: Client user reads all internal sales interactions on their lead	Medium	4.3
No non-negative constraint on deals.amount allows negative deal values	Medium	4.3
leads.value has no DB-level bounds check; client-side Zod min(0) is bypassable	Medium	4.3
snoozed_until accepts arbitrary far-future timestamps, permanently suppressing push	Medium	4.3
Unvalidated occurred_at allows arbitrary backdating or future-dating of CRM interaction	Medium	4.3
No DB-level enforcement of completed/completed_at relationship in tasks table	Medium	4.3
crm-chat Edge Function Leaks Internal Error: "userClient.auth.getClaims is not a function"	Medium	4.3
Race condition (TOCTOU) bypasses AI chat rate limit (20 msg/hour)	Low	3.1
Service Worker Open Redirect: Absolute URL in Push Payload Bypasses Origin Check	Low	2.6

Appendix B: Aikido finding adjudication

Aikido's reports contain no CVSS vectors. Every finding below was assigned a CVSS v3.1 vector from Aikido's own published description, business-impact, and reproduction text, using the scoring conventions in Section 2.5. The computed score and resulting band follow. Findings marked as collapsed are duplicate reports of the same root cause, counted once per Section 2.3.

Target	ID	Finding	Aikido label	Assigned CVSS vector	Score	Band
Crypto betting	PT-1	Unauth create-admin edge fn, hardcoded creds, full admin takeover	Critical	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H	10.0	Critical
Crypto betting	PT-2	Unauth create markets / insert bets / resolve / trigger payouts	Critical	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:H/A:H	10.0	Critical
Crypto betting	PT-3	Race in generate-wallet mints 1000 credits per concurrent call	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:L	7.1	High
Crypto betting	PT-4	Unauth REST arbitrary market creation and deletion	High	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N	7.5	High
Crypto betting	PT-5	Unauth close-expired-markets triggers resolution and payouts	High	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:H/A:H	10.0	Critical
Crypto betting	PT-6	Unauth ai-assistant edge fn, arbitrary AI credit consumption	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L	5.3	Medium
Crypto betting	PT-7	Weak password policy accepts '123456'	Medium	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N	4.8	Medium
Crypto betting	PT-8	Auth cookie missing HttpOnly/Secure/SameSite	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Crypto betting	PT-9	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Crypto betting	PT-10	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Invoicing SaaS	PT-1	Cross-tenant invoice disclosure via mutable profiles.email	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N	7.7	High
Invoicing SaaS	PT-2	Recipient updates invoice status via direct PATCH	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:H/A:N	7.7	High
Invoicing SaaS	PT-3	Cross-tenant workspace takeover via mutable profiles.company_id	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N	9.6	Critical
Invoicing SaaS	PT-4	Unauth process-invoice-email-retries worker	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L	6.5	Medium
Invoicing SaaS	PT-5	Recipients access owner-only delivery audit data	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N	4.3	Medium
Invoicing SaaS	PT-6	Sensitive cookie missing Secure/HttpOnly/SameSite	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Invoicing SaaS	PT-7	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Invoicing SaaS	PT-8	Weak TLS cipher suites accepted (duplicate)	Low	collapsed to prior finding		

Target	ID	Finding	Aikido label	Assigned CVSS vector	Score	Band
Invoicing SaaS	PT-9	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
NFT platform	PT-1	Unauth cross-wallet read/write in dashboard tables via anon API	High	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:H/A:N	8.6	High
NFT platform	PT-2	Non-atomic marketplace purchase forces 'sold' without transfer	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N	4.3	Medium
NFT platform	PT-3	Unauth ai-generate edge fn, AI credit abuse	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L	5.3	Medium
NFT platform	PT-4	Auth cookie lacks HttpOnly/Secure/SameSite	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
NFT platform	PT-5	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
NFT platform	PT-6	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Bounty marketplace	PT-1	Contributor alters task bounty and completion state	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:H/A:N	7.7	High
Bounty marketplace	PT-2	Authed profile enumeration via permissive access control	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N	7.7	High
Bounty marketplace	PT-3	Non-manager contributors read others' pending submissions	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N	7.7	High
Bounty marketplace	PT-4	Contributor-controlled review state on submission creation	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N	4.3	Medium
Bounty marketplace	PT-5	Cross-user activity_log read access	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N	4.3	Medium
Bounty marketplace	PT-6	Single-active-claim control bypass	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N	4.3	Medium
Bounty marketplace	PT-7	Race in ai-chat hourly limiter allows >20/hr	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:L	4.3	Medium
Bounty marketplace	PT-8	Account enumeration via differential resend responses	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Bounty marketplace	PT-9	Cross-user ai_messages write poisoning	Low	CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:L/A:N	3.1	Low
Bounty marketplace	PT-10	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Bounty marketplace	PT-11	Weak TLS cipher suites accepted (duplicate)	Low	collapsed to prior finding		
Bounty marketplace	PT-12	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Project management	PT-1	Cross-account task IDOR via Kanban API	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:H/A:N	7.7	High
Project management	PT-2	Authed users read others' draft projects	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N	7.7	High
Project management	PT-3	Student bypasses admin approval by setting status	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N	4.3	Medium
Project management	PT-4	Authed IDOR exposes others' closed marketplace gigs	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N	4.3	Medium

Target	ID	Finding	Aikido label	Assigned CVSS vector	Score	Band
Project management	PT-5	Unauth project-copilot leaks RLS-protected data	High	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	7.5	High
Project management	PT-6	Authed IDOR leaks event participant user IDs	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N	4.3	Medium
Project management	PT-7	Event participant cap bypass via direct inserts	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N	4.3	Medium
Project management	PT-8	Unauth ai-match public backend AI invocation	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L	5.3	Medium
Project management	PT-9	Weak password policy allows '123456'	Medium	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N	4.8	Medium
Project management	PT-10	Unauth Realtime subscription leaks UUIDs on DELETE	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Project management	PT-11	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Project management	PT-12	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Learning platform	PT-1	Unauth AI proxy in chat-assistant, credit abuse	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L	5.3	Medium
Learning platform	PT-2	Unauth profile enumeration via profiles endpoint	High	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	7.5	High
Learning platform	PT-3	Verbose Supabase REST errors disclose DB details	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	Medium
Learning platform	PT-4	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Learning platform	PT-5	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Healthcare booking	PT-1	Ineffective rate limiting in medibook-chat, AI abuse	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L	5.3	Medium
Healthcare booking	PT-2	Account enumeration via differential signup responses	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Healthcare booking	PT-3	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Healthcare booking	PT-4	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Team collaboration	PT-1	Unauth ai-analyze edge fn proxies Lovable AI	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L	5.3	Medium
Team collaboration	PT-2	Unauth ai-writer edge fn, arbitrary AI usage	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L	5.3	Medium
CRM	PT-1	Unauth invocation of privileged send-push-reminders	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L	6.5	Medium
CRM	PT-2	Internal authz details exposed via raw DB errors	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	Medium
CRM	PT-3	Verbose internal error disclosure in crm-chat	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	Medium
CRM	PT-4	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low

Target	ID	Finding	Aikido label	Assigned CVSS vector	Score	Band
CRM	PT-5	Weak TLS cipher suites accepted (duplicate)	Low	collapsed to prior finding		
CRM	PT-6	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low
Real estate	PT-1	No brute-force protection on password login	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	5.3	Medium
Real estate	PT-2	CSV formula injection in admin leads export	Medium	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N	5.4	Medium
Real estate	PT-3	Unauth profile enumeration via REST profiles table	High	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	7.5	High
Real estate	PT-4	Unauth access to /functions/v1/chat, AI credit drain	Medium	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L	5.3	Medium
Real estate	PT-5	Weak password policy accepts trivial password	Medium	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N	4.8	Medium
Real estate	PT-6	User enumeration via distinct signup errors	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Real estate	PT-7	Weak TLS cipher suites accepted	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	3.7	Low
Real estate	PT-8	Missing or misconfigured security headers	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	3.1	Low