RUGAZE_

WHITEPAPER · V1.0

Evidence-based Solana transaction analysis.

What a transaction structurally does, and what confirmed on-chain evidence shows — never a guess dressed up as certainty.

Document	Version	Audited state	Tests
RUGAZE_WHITEPAPER.pdf	1.0	HEAD 5555ec1	851 / 851 passing

01
EXECUTIVE SUMMARY

What RUGAZE is, in one page

RUGAZE is a transaction *understanding* engine for Solana. Given a transaction, it decodes the instructions, interprets what actually executed on-chain (including inner CPI calls), computes real balance and authority-change evidence, and returns a structured, evidence-backed report — every fact labeled by how it was established, never presented as more certain than it is.

It is not a simulator, not a blocklist, and not a single yes/no safety gate. It is deliberately narrower and more honest than any of those: RUGAZE reports what a transaction is structurally capable of doing, backed by a specific instruction index, account, and decoded field for every claim — and it says **UNKNOWN** when the evidence doesn't support a confident answer, rather than guessing.

The single most important distinction in this document is between what RUGAZE already does and where it is headed. The two are easy to blur in casual conversation about the product, so this whitepaper draws the line explicitly, here, before anything else:

TODAY — DEPLOYED

RUGAZE analyzes an **already-confirmed** Solana transaction and explains what happened, using structured, on-chain evidence. This is real, tested, and live in production right now.

FUTURE DIRECTION — NOT DEPLOYED

RUGAZE is working toward understanding what a transaction **may do before** a user signs it. This is the documented next direction — it is not scoped, not approved, and not available today. See §19.

No user is protected before signing by RUGAZE today. Every capability described as "current" in this document operates on a transaction that has already executed and been confirmed on-chain.

Deployed today

A production Cloudflare Worker API, live at a public URL, with a real free tier, a real paid tier, and 851/851 tests passing at the audited commit.

What it analyzes

Already-confirmed, already-executed Solana transactions — static structure, CPI evidence, balance deltas, authority/ownership transitions, and Token-2022 extension state.

How to check any of this

The real on-chain transaction cited throughout this document is independently verifiable on public block explorers RUGAZE does not control. See the verification page immediately following this one.

This document distinguishes four maturity states throughout, using the badges below. A reader should never need to guess which one applies to a given claim.

● DEPLOYED ● BUILT, NOT DEPLOYED ● PLANNED ● FUTURE / EXPLORATORY

BEFORE YOU CONTINUE

Verify this yourself — don't take our word for it

Every material claim in this document can be checked independently, without trusting this PDF, a screenshot, or an internal log. Where a claim cannot currently be independently verified by a stranger, that is stated plainly below rather than implied otherwise.

1 · Live production endpoint

RUGAZE's API is real and live in production today. Its current public address is a temporary Cloudflare-assigned URL, not a permanent RUGAZE-owned one — this document intentionally does not publish it while that migration is in progress, and will be updated with the permanent address once it exists. This is a publishing decision about this document, not a claim that the service is unavailable.

2 · A real, confirmed on-chain transaction

The `/analyze` evidence in §12 of this document is not a constructed example. It is a real transaction, independently inspectable on two public Solana block explorers — both will show the identical signature, slot, and success status shown in this document.

```
4EdazRxje6cswMyf2AzVh3yGEAuyH6Fz8USEtFiJoHgfBuYf3AuHSBCmhjqVWhRE8BF7wTJeQzPF5yjV7dkv3A23
```

explorer.solana.com/tx/4EdazR…3A23 · solscan.io/tx/4EdazR…3A23

Independently re-confirmed while preparing this revision: Solana Explorer shows `Status: Success`, `Slot: 440,755,408`, matching this document's own evidence exactly (screenshot, §A.2). Solscan's own bot-verification page blocked this document's automated check — a standard human browser check most visitors pass in seconds; it is not a sign the link is wrong.

3 · The public API contract

The free tier is real and self-serve: a caller POSTs an email to `/signup`, receives a bearer key, then POSTs a transaction signature to `/analyze` with that key. This is the exact, current shape of the public contract — not a simplified illustration:

```
$ curl -X POST <RUGAZE production endpoint>/signup \
    -H "Content-Type: application/json" \
    -d '{"email":"you@example.com"}'
# -> {"apiKey":"rugaze_<your own key>","tier":"free","monthlyLimit":10}

$ curl -X POST <RUGAZE production endpoint>/analyze \
    -H "Authorization: Bearer <your own key>" \
    -d
'{"signature":"4EdazRxje6cswMyf2AzVh3yGEAuyH6Fz8USEtFiJoHgfBuYf3AuHSBCmhjqVWhRE8BF7wTJeQzPF5yjV7
dkv3A23"}'
```

The endpoint address is intentionally omitted from this revision (see §1 above) — every other line here, including the exact request/response shape and the real transaction signature, is the genuine public contract, reproducible as soon as the permanent address is published.

BEFORE YOU CONTINUE, CONT'D

Source code, and what is — and isn't — public today

> STATED PLAINLY
>
> RUGAZE's source repository is **not currently publicly accessible**. This document does not link to it and does not claim a stranger can inspect commit history, source files, or the internal milestone registry today. Where this document cites a file path or commit hash, it is describing the internal evidentiary basis for a claim — not asserting that path is publicly browsable.

What *is* independently checkable by anyone, right now, without any special access: the real on-chain transaction referenced throughout this document, on two block explorers RUGAZE does not control.

Claim → verification table

CLAIM	HOW TO VERIFY	PUBLIC LOCATION	STATUS
Real transaction analysis	Inspect the signature on Solscan/Explorer — matches this document's own evidence exactly	Solana Explorer, Solscan	● VERIFIABLE
Production API is live	Not link-verifiable in this revision — permanent address not yet published	Withheld pending permanent domain	● REAL, URL WITHHELD
Public landing page & pricing	Not link-verifiable in this revision, same reason	Withheld pending permanent domain	● REAL, URL WITHHELD
Free signup works	Not link-verifiable in this revision, same reason	Withheld pending permanent domain	● REAL, URL WITHHELD
Test suite result (851/851)	Not independently checkable — source repository is private	Internal only	● NOT PUBLIC
Source code / commit history	Not independently checkable — repository is private	Internal only	● NOT PUBLIC

This table will be updated as the permanent domain and, separately, the source repository become public. Overstating what is verifiable today would defeat the purpose of this section — so would understating it: these claims are true today, only the clickable proof is pending.

02
THE PROBLEM

Signing is a trust decision made with almost no information

Every Solana transaction a user signs grants some set of capabilities — to move funds, reassign an authority, approve a delegate, or invoke another program. Wallet UIs typically show none of this in a form a human can actually evaluate. The decision to sign is made on trust in the source of the request, not on evidence about what the transaction itself can do.

This is not a hypothetical problem. The pattern below recurs across real, independently reported incidents. These are cited to establish that the underlying risk is real — **not as a claim that RUGAZE would have caught any of them.** RUGAZE did not exist at the time of these incidents, and no retroactive claim is made here.

INCIDENT	MECHANISM	SOURCE
Drift Protocol, ~$270M at risk (reported April 2026)	Durable-nonce abuse — two transactions, four slots apart, granted an attacker admin control. No code bug, no stolen key.	CoinDesk
TOCTOU ("time-of-check to time-of-use") drainers	Programs that behave benignly under `simulateTransaction` and maliciously at real execution time.	Blockaid
Token-2022 extension abuse	Conservatively estimated at $50M+ in Q1 2026 losses via `PermanentDelegate`/`TransferHook` misuse.	Industry reporting (dev.to)
"Crypto Copilot" malware	Injected hidden transfer instructions into Raydium swaps, silently skimming SOL per trade since June 2025.	Industry reporting

Full citation trail: `docs/PRD.md:56–61`, `docs/THREAT_MODEL.md:224–227`. These sources include the same explicit caveat repeated here.

What each of these has in common: the transaction's *capability* was legitimate-looking, decodable, and in most cases fully explicable from the instruction data alone — if something had actually decoded and explained it before the user signed.

03
WHY SOLANA TRANSACTION RISK IS STRUCTURALLY HARD

This isn't a UI problem. It's a parsing and evidence problem.

Four structural facts about how Solana transactions actually work make this materially harder than "just show the instructions":

Version ambiguity

Legacy and v0 transactions are structurally different. A parser that assumes legacy-only will silently mis-parse any v0 transaction it encounters — not fail loudly, mis-parse.

Address Lookup Tables

v0 transactions can reference accounts via ALTs. Resolution order must exactly match validator behavior, or every account-index reference after the static keys is wrong.

Inner instructions are runtime-only

Cross-program invocations (CPI) only exist as a consequence of actual execution — they are not visible in the transaction's own static instruction list at all.

Token-2022 extension state lives on accounts

`PermanentDelegate`, `TransferHook`, and similar extensions are account-level TLV state, not instruction data — a naive "decode the instruction" implementation misses them entirely.

Each of these is a place where a system that guesses, rather than structurally decodes and discloses uncertainty, produces a wrong answer that *looks* confident. RUGAZE's architecture (§7–9) exists specifically to make that class of error structurally difficult to produce.

Source: `docs/SECURITY_ARCHITECTURE.md:92–99`.

04–05
WHAT RUGAZE IS — AND IS NOT

A precise, bounded definition

WHAT RUGAZE IS

- Structural, evidence-based transaction analysis
- Real CPI / inner-instruction interpretation, from confirmed execution evidence
- Real balance, ownership, and authority-change evidence
- A CLI and a minimal HTTP API over one shared analysis engine
- A system that reports **capability**, evidenced and traceable to specific instruction fields

WHAT RUGAZE IS NOT

- Not a guarantee that a transaction is safe
- Not an exhaustive scam detector
- Not a custody product, wallet, or signing service
- Not an auto-fix or transaction-modification system
- Not a DEX/Jupiter swap-route interpreter (today)
- Not, today, a pre-signature "wallet firewall"
- Not a portfolio tracker or block explorer

RUGAZE reports what a transaction's instructions and confirmed execution **can do** — never what its author **means** to do. Deployment recency, unverified authorship, or an unrecognized program are disclosed as facts with their own evidence and confidence label; they are never rendered as an implied verdict like "this is suspicious." This is a deliberate, permanent scope boundary, not a current limitation to be removed later.

Source: `docs/PRD.md:9,41–44`, `docs/THREAT_MODEL.md:15–19`, `landing/index.html:272–287` (the same distinction is published live on the product's own landing page).

06
CORE PRODUCT PRINCIPLES

Six rules the entire system is built to enforce

01 · FAIL CLOSED, NEVER FAIL OPEN

An instruction, program, or account RUGAZE cannot confidently analyze is reported as `UNKNOWN` / `UNABLE_TO_ANALYZE` — never silently treated as safe.

02 · UNKNOWN IS A VALID RESULT

`UNKNOWN` is a first-class, permanent product outcome, not an error state to be minimized toward zero. Driving it down by guessing would be a direct regression of the core safety property.

03 · EVIDENCE MUST REMAIN TRACEABLE

Every finding carries a pointer to the specific instruction index, account, and decoded field that produced it. No free-floating "this looks suspicious."

04 · CAPABILITY DOES NOT PROVE INTENT

RUGAZE analyzes what a transaction *can* do, not what its author *means* to do — a deliberate, load-bearing scope boundary.

05 · SIMULATION IS EVIDENCE, NOT PROOF

Where simulation is used, it is one evidence source among several — never treated as proof of safety on its own.

06 · NO UNFALSIFIABLE CLAIMS

RUGAZE never states "100% safe," "detects every scam," or "zero false negatives" in product copy, documentation, or generated explanations.

Source: `docs/PRD.md:32–35`, `docs/SECURITY_ARCHITECTURE.md:141,181–183`, `docs/THREAT_MODEL.md:15–19`.

07
SYSTEM ARCHITECTURE

A ten-layer pipeline, built incrementally and honestly

RUGAZE's architecture separates ten distinct concerns so that "I decoded this," "I judged this," and "I simulated this" can never be silently conflated. Each layer below is labeled by its **actual** build state — not its designed state.

A	**Transaction Decoding**	Wire-format parsing, legacy + v0, ALT resolution	● BUILT
B	**Instruction Normalization**	Typed decode; unrecognized instructions kept, never dropped	● BUILT
C	**Account / State Analysis**	getAccountInfo / getMultipleAccounts reads, PDA re-derivation	● PARTIAL
D	**Deterministic Security Rules**	Family A rules + Token-2022 extension findings	● BUILT
E	**Heuristic Analysis**	Non-deterministic signals, clearly labeled if ever added	● DESIGNED ONLY
F	**Simulation**	Engine implemented and tested — not wired to any live route	● BUILT, UNUSED
G	**Simulation-Result Analysis**	Capability-vs-behavior gap detection	● DESIGNED ONLY
H	**Risk Aggregation**	Fail-closed fold across static + CPI + extension evidence	● BUILT
I	**Human-Readable Explanation**	TransactionUnderstanding translation layer	● BUILT
J	**API / SDK Interface**	Deployed Cloudflare Worker, versioned public contract	● BUILT

● Live in the deployed product today ● Implemented, tested, not yet reachable from the public API

● Architecturally designed only — no implementation exists

Source: `docs/SECURITY_ARCHITECTURE.md:9–22`, `docs/M2_ARCHITECTURE_LOCK.md:372–374,414–416`. Layer F (Simulation): confirmed this audit via direct source search — zero call sites for `simulateTransaction`/`SimulationCapable` in `worker/` or `cli/server.ts`.

08
TRANSACTION ANALYSIS PIPELINE

What actually happens to a real request

This is the real, deployed orchestration for `POST /analyze` — every stage below corresponds directly to a function in `cli/analyze.ts`'s `analyzeSignature()`, the same engine used by both the CLI and the production Worker.

01 **Transaction Signature**
Caller supplies a base58 Solana signature

↓

02 **RPC Fetch**
`getTransaction` against the configured provider — real, confirmed on-chain data

↓

03 **Transaction Decode**
Legacy/v0 parsing, ALT resolution, instruction normalization

↓

04 **CPI / Inner Instruction Interpretation**
Real inner-instruction evidence from the transaction's own confirmed `meta`

↓

05 **Balance Evidence**
Real SOL + SPL token deltas from confirmed pre/post balances — not simulated, not estimated

↓

06 **Authority / Ownership Evidence**
Decoded `SetAuthority`/`Approve`/`Assign` transitions

↓

07 **Token-2022 Extension Analysis**
Bounded, gated `getMultipleAccounts`/`getAccountInfo` reads for extension state

↓

08 **Fail-Closed Risk Fold**
Three-guard `computeOverallRiskLevel()` — see §9

↓

09 **Optional: TransactionUnderstanding**
Plain-language translation, opt-in via `understand: true`

Source: `cli/analyze.ts` (orchestration), `worker/index.ts` (production wiring). Every stage listed here is confirmed live — none is aspirational.

09
RISK MODEL AND FAIL-CLOSED DESIGN

The exact logic, not a marketing summary

RUGAZE uses six severity levels — `SAFE`, `LOW_RISK`, `WARNING`, `HIGH_RISK`, `CRITICAL`, and `UNKNOWN` — and a separate confidence axis (`DETERMINISTIC` down to `UNKNOWN`), so that *how sure* RUGAZE is is never collapsed into *how bad* a finding is.

The three-guard fold (production code, unmodified)

This is a direct representation of `cli/analyze.ts:204–209` (`computeOverallRiskLevel()`), reproduced here as a diagram, not simplified.

REAL PRODUCTION EVIDENCE — FAIL-CLOSED BEHAVIOR OBSERVED LIVE ● VERIFIED

```
POST /analyze · real mainnet signature, slot 440755408
overallRiskLevel: "UNKNOWN"
reason: "CPI (inner-instruction) evidence is present but not fully resolved —
at least one inner instruction could not be decoded/evaluated. Per the
R-1 fix, this forces overallRiskLevel to UNKNOWN even when
staticAnalysis.level alone would read SAFE."
```

Captured live against production, this audit. A transaction with zero deterministic findings still correctly reports UNKNOWN rather than SAFE, because one piece of evidence was incomplete — exactly the guarantee §6 describes, observed in a real response, not asserted.

09.1
THE EVIDENCE MODEL

Three states, never collapsed into one

Every fact RUGAZE reports is labeled by how it was established. These are evidence states, not confidence-in-marketing-copy:

VERIFIED

Directly observed from confirmed on-chain execution data (e.g. real pre/post balances from `getTransaction.meta`).

INFERRED

A decoded instruction's own declared value (e.g. "this instruction sets the new authority to X") — not independently cross-checked against a fresh state read in this version.

UNAVAILABLE

The data source needed to answer could not be read (pruned by RPC retention, a failed fetch, an unresolvable account) — disclosed as a gap, never silently omitted.

Authority, ownership, and delegate-transition evidence today is **INFERRED**, not VERIFIED — RUGAZE reports what a decoded instruction declares, not a confirmed fresh state read proving it landed. This is disclosed as a known limitation (§17), not hidden inside a generic "confidence score."

Source: `README.md:1–7,115–121`.

10

TRANSACTION UNDERSTANDING ● DEPLOYED

Translating evidence into plain language, without inventing anything

Opt-in via `understand: true` on `POST /analyze`, `buildTransactionUnderstanding()` is a pure function over an already-computed report — it adds zero new findings, zero new RPC calls, and zero new severity logic. It organizes existing evidence into six sections, and every line carries an `evidenceRef` pointing back into the source report, so a line can be mechanically traced to the exact evidence that produced it.

You Send / You Receive

Real balance deltas for the signer's own accounts.

Permissions Granted

Delegations and approvals the transaction creates.

Authorities & Control Changes

Mint/freeze/owner authority reassignments.

Where Funds Go

Other accounts that gained value — explicitly not claimed as a proven causal pairing.

Other Effects

Account lifecycle facts (creation/closure) and remaining findings.

Risk / Analysis Status

The overall risk level plus a completeness signal, stricter than any single underlying flag.

REAL PRODUCTION RESPONSE — POST /ANALYZE, UNDERSTAND: TRUE ● VERIFIED


```
{
  "youSend": { "status": "ITEMS" },
  "youReceive": { "status": "ITEMS" },
  "permissionsGranted": { "status": "UNDETERMINED" },
  "authoritiesControlChanges": { "status": "UNDETERMINED" },
  "whereFundsGo": { "status": "ITEMS" },
  "otherEffects": { "status": "ITEMS" },
  "riskAnalysisStatus": { "overallRiskLevel": "UNKNOWN", "fullyResolved": false }
}
```


Note the `UNDETERMINED` sections above — the translation layer honestly reports "not fully checked," not a false "NONE_FOUND," exactly matching the certainty model in `cli/understanding.ts:14–28`.

11
TOKEN-2022 ANALYSIS

Extension-level coverage — and one item not yet live

Token-2022 extension state lives on *accounts*, not instructions (§3) — RUGAZE performs bounded, gated state reads specifically to surface it. Coverage is not uniform across extensions; each item below is labeled individually.

EXTENSION	RULE ID	STATUS
PermanentDelegate	A-06	● DEPLOYED
TransferHook	A-07	● DEPLOYED
MintCloseAuthority	A-08	● DEPLOYED
TransferFeeConfig	A-09	● IMPLEMENTED · NOT PUSHED / NOT DEPLOYED

A-09 is implemented and committed to a local commit, but has not been pushed to the main branch or deployed to production. It must not be read as a live capability of the deployed product from this document. This is the milestone registry's own recorded state, preserved here exactly.

A disclosed limitation that applies to A-06/A-07 specifically: Tier-2 confirmed active-delegate-exercise (A-06) and the active-hook execution path (A-07) have never been observed on real mainnet to date — both remain test-fixture-proven, not real-world-confirmed. This does not expire; it is a permanent note attached to the milestone, not a temporary caveat.

Source: `docs/MILESTONE_REGISTRY.md:53–54,69`, `m2/rules/token2022Extensions.ts:1–85`.

12

API AND PRODUCT INTERFACE ● DEPLOYED

Three routes, one shared engine, a locked public contract

The production Worker exposes exactly three routes: `GET /health`, `POST /signup`, `POST /analyze`. The public response shape is protected by a dedicated regression-lock test suite (`tests/contract/apiPublicContract.test.ts`) that fails the build if a field is silently added, removed, or renamed.

GET /HEALTH — REAL PRODUCTION RESPONSE, CAPTURED THIS AUDIT ● VERIFIED

```
$ curl .../health
{"status":"ok","rpcConfigured":true}   HTTP 200
```

POST /ANALYZE — REAL MAINNET SIGNATURE, CONDENSED FOR READABILITY ● VERIFIED

```
$ curl -X POST .../analyze -H "Authorization: Bearer <key>" \
    -d '{"signature":"4Edaz...v3A23"}'

status: "OK"     onChainOutcome: "SUCCESS"     slot: 440755408
overallRiskLevel: "UNKNOWN"   HTTP 200
```

19 top-level fields, matching the locked public contract exactly. Full response condensed here for space; the raw evidence file is preserved alongside this document's source.

This document deliberately shows real request/response pairs, not a schema description — every value above is from a live call made against production during this audit, not a constructed example.

13

SIGNUP, ACCESS & RELIABILITY INFRASTRUCTURE ● DEPLOYED

Free and paid access, and the integrity work underneath it

Free tier

Self-serve, 10 analyses/month per key, enforced live in production (11th request returns a real 429).

Developer tier

₹1,900/month (real, working Razorpay checkout), 1,000/month, applied by manual key upgrade after payment confirmation — self-serve billing is not yet built.

F-1 — Signup integrity, deployed this cycle

Duplicate signups for the same normalized email are rejected without ever disclosing the existing key:

POST /SIGNUP, SECOND CALL WITH AN ALREADY-REGISTERED EMAIL ● VERIFIED

```
{"error":"an API key already exists for this email. There is no
recovery flow yet -- if you've lost your key, see the contact
instructions on the landing page."}   HTTP 409
```

Captured live against production this audit. No `apiKey` field is present in a 409 response, by design and by test.

This includes a case-insensitive match (`Foo@Bar.com` and `foo@bar.com` are treated as the same identity) and does not retroactively protect accounts created before this change — a disclosed, permanent limitation, not a hidden gap.

Source: `worker/usage.ts`, `evidence/F1_SIGNUP_INTEGRITY_PUBLIC_CONTRACT/`, commit `4755e897`.

14
PRIVACY AND SECURITY PRINCIPLES

Minimal by design, not by omission

No password / account system

An API key is an opaque bearer token. There is no login, no session, no stored password.

Minimal stored data

The key-value store holds exactly: email, tier, and creation timestamp per key — nothing else.

No key recovery yet

A lost key cannot currently be recovered self-serve — disclosed directly in the API's own error text.

Read-only analysis

RUGAZE never custodies funds, never signs, and never holds any credential belonging to the transaction being analyzed.

Source: `worker/usage.ts`, `evidence/F1_SIGNUP_INTEGRITY_PUBLIC_CONTRACT/EVIDENCE_INDEX.md`.

15
RELIABILITY AND INFRASTRUCTURE

Provider-agnostic by construction, single-endpoint today

RUGAZE's Solana RPC transport (`m2/chain/rpcProvider.ts`) is a thin, hand-rolled standard JSON-RPC client — no provider-specific SDK, no proprietary API anywhere in the codebase. Every method call uses documented, standard Solana JSON-RPC. Switching RPC providers requires changing exactly one configuration value, not a code change — demonstrated directly this cycle when the production endpoint was rotated to Helius with zero source changes.

Timeout

10-second bound via `AbortController` on every RPC call.

Retry policy

Up to 2 retries, exponential backoff, **only** for transient transport failures — a well-formed error response is never retried.

Disclosed limitation: exactly one endpoint is configured at any given time. There is no automatic failover to a second provider today. This is a known, accepted gap at RUGAZE's current traffic volume, not a hidden one.

Source: `m2/chain/rpcProvider.ts:53–73,235–320`; this cycle's RPC infrastructure audit and production verification.

16
CURRENT IMPLEMENTATION STATUS

Every major capability, one table

CAPABILITY	STATUS
Static decode + Family-A deterministic rules (M0)	● DEPLOYED
CPI / inner-instruction interpretation, balance evidence (M2)	● DEPLOYED
Token-2022: PermanentDelegate, TransferHook, MintCloseAuthority	● DEPLOYED
Token-2022: TransferFeeConfig (A-09)	● COMMITTED, NOT DEPLOYED
Fail-closed risk aggregation (R-1 fold)	● DEPLOYED
TransactionUnderstanding plain-language layer	● DEPLOYED
Public Worker API (health / signup / analyze)	● DEPLOYED
Public landing page + real pricing	● DEPLOYED
Signup integrity (F-1)	● DEPLOYED
Public API contract regression lock	● DEPLOYED
Provider-agnostic RPC transport (Helius-configured)	● DEPLOYED
Simulation engine (SimulationCapable)	● BUILT, NOT WIRED TO PUBLIC API
Pre-signature transaction protection	● FUTURE DIRECTION, NOT YET SCOPED
DEX / Jupiter route interpretation	● DEFERRED, NOT BUILT
Self-serve billing	● NOT BUILT — MANUAL FULFILLMENT TODAY

Authoritative source for this entire table: `docs/MILESTONE_REGISTRY.md`, HEAD `5555ec1c7cd47fa187866e68a86ca075dfefea0a`. Test suite: 851/851 passing, verified at the same commit.

17
KNOWN LIMITATIONS

Stated plainly, not buried in a footnote

- **Authority/ownership evidence is declared-value only.** RUGAZE reports what a decoded instruction says the new value is, not a fresh state read confirming it landed.
- **No Jupiter/DEX swap-route interpretation.** Deliberately deferred until real customer demand justifies it.
- **No self-serve payment yet.** The Developer tier is real and enforced; upgrade is manual after payment confirmation.
- **Single RPC endpoint, no automatic failover.** A sustained outage of the configured provider degrades the service; retries only cover brief, transient failures.
- **Usage counting is not atomic.** A Cloudflare KV read-then-write race is a known, low-severity, disclosed condition at current traffic volume.
- **Duplicate-email protection is not retroactive.** Keys created before this protection existed are not backfilled into the new index.
- **28.6% real-corpus coverage.** Against a real, collected transaction corpus, 6/21 return a definitive SAFE and 15/21 correctly return UNKNOWN — a deliberately un-inflated number, not chased upward by guessing on unverified program interfaces.
- **No pre-signature protection today.** Everything RUGAZE analyzes has already been confirmed on-chain before analysis begins.

Source: `README.md:113–144`, `docs/M1_FREEZE.md:82–103`, `docs/MILESTONE_REGISTRY.md` per-milestone notes.

Where this is going, without inventing a schedule

The companion **RUGAZE Roadmap** document covers this in full, phase by phase. In summary: five phases are complete (foundation research, the static engine, the stateful engine, the product surface, and this cycle's foundation hardening). The next named direction is **pre-signature transaction analysis** — moving from "here is what happened" to "here is what may happen before you sign."

WHAT EXISTS TODAY	WHERE RUGAZE IS HEADING
Post-hoc analysis of an already-confirmed, already-executed transaction. Real evidence, real production traffic, zero pre-sign capability.	Pre-signature analysis, using the simulation engine that already exists at the code layer but has no defined product/API contract yet. Requires a dedicated audit and blueprint before any implementation begins.

This is not an active implementation milestone. No formally approved next milestone currently exists in the project's own governance records. The direction is documented; the design is not.

Exploratory, long-term (no committed scope)

- DEX/aggregator-aware interpretation (Jupiter, Raydium, Orca)
- External reputation/registry integration for program identity
- A public SDK and wallet/dApp design-partner integrations
- Batch/portfolio scanning
- A security-team/exchange/custodian-focused tier

Source: `docs/MILESTONE_REGISTRY.md:87`, `docs/ROADMAP.md:135–141`, and this project's own roadmap audit conducted this cycle.

20
CONCLUSION

A real, running system with disclosed edges

RUGAZE today is a working, tested, deployed engine that turns a Solana transaction signature into evidence-backed structural analysis — not a pitch for what it will eventually do. Every claim in this document traces to a specific file, test, commit, or live production response, and every gap is stated as a gap, not smoothed over.

The next real step is not a new feature announcement; it is an audit. Pre-signature protection is the documented product mission RUGAZE hasn't reached yet, and this document has tried as hard as possible not to blur that line.

RUGAZE. Evidence-based Solana transaction analysis.

A
APPENDIX — EVIDENCE INDEX

What was proven, how, and where it lives

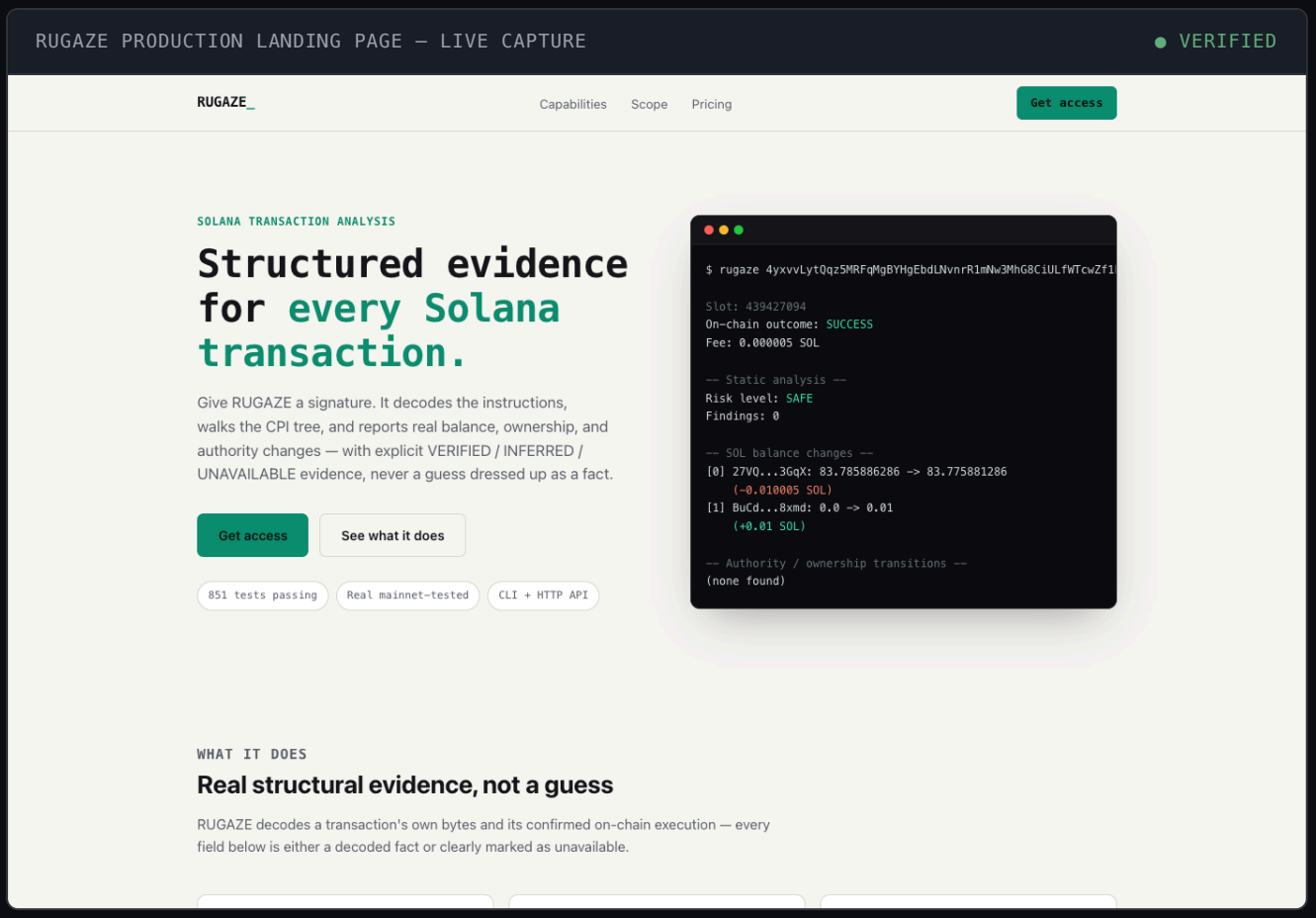
WHAT WAS PROVEN	HOW VERIFIED	STATUS	SOURCE
Static engine coverage & honesty discipline	Real-corpus run, 21 real transactions	VERIFIED	`docs/M1_FREEZE.md`
Token-2022 extensions A-06/07/08 live in production	Production evidence package	VERIFIED	`evidence/TOKEN_2022_EXTENSIONS/`
A-09 implemented but not deployed	Registry deployment field, git log	VERIFIED	`docs/MILESTONE_REGISTRY.md`, commit `dd33822`
Understanding layer additive, byte-compatible	Structural equality, two real post-deploy calls	VERIFIED	`evidence/B1C_WORKER_API_DEPLOYMENT/`
Landing page live, real pricing, no placeholders	Real browser screenshot + direct HTML fetch	VERIFIED	`evidence/L1_LANDING_PAGE_DEPLOYMENT/`; screenshot §A.1
Real on-chain transaction independently confirmed	Cross-checked directly against Solana Explorer	VERIFIED	Screenshot §A.2; explorer.solana.com
Signup integrity (F-1) live in production	Real 201/409 calls, case-variant duplicate	VERIFIED	`evidence/F1_SIGNUP_INTEGRITY_PUBLIC_CONTRACT/`
Helius RPC rotation, zero code change	`git diff --stat` empty across rotation	VERIFIED	this cycle's RPC audit + production verification
851/851 tests passing at audited commit	`npm test`, re-run this audit	VERIFIED	HEAD `5555ec1`
Simulation engine unused by any live route	Direct source grep, zero call sites found	VERIFIED	`worker/`, `cli/server.ts`

All commit hashes, file paths, and evidence directories above refer to the RUGAZE source repository at HEAD `5555ec1c7cd47fa187866e68a86ca075dfefea0a`. Live production responses shown throughout this document were captured during the preparation of this document, against the real deployed API, using freshly-issued API keys and freshly-sourced real mainnet transaction signatures — not constructed examples.

A.1
APPENDIX — LIVE PRODUCTION SCREENSHOT

The public landing page, captured live

Captured directly from RUGAZE's live production landing page during preparation of this document — a real screenshot of the real, deployed product, not a mockup. Its address is intentionally not published in this revision (see the verification section near the front of this document).

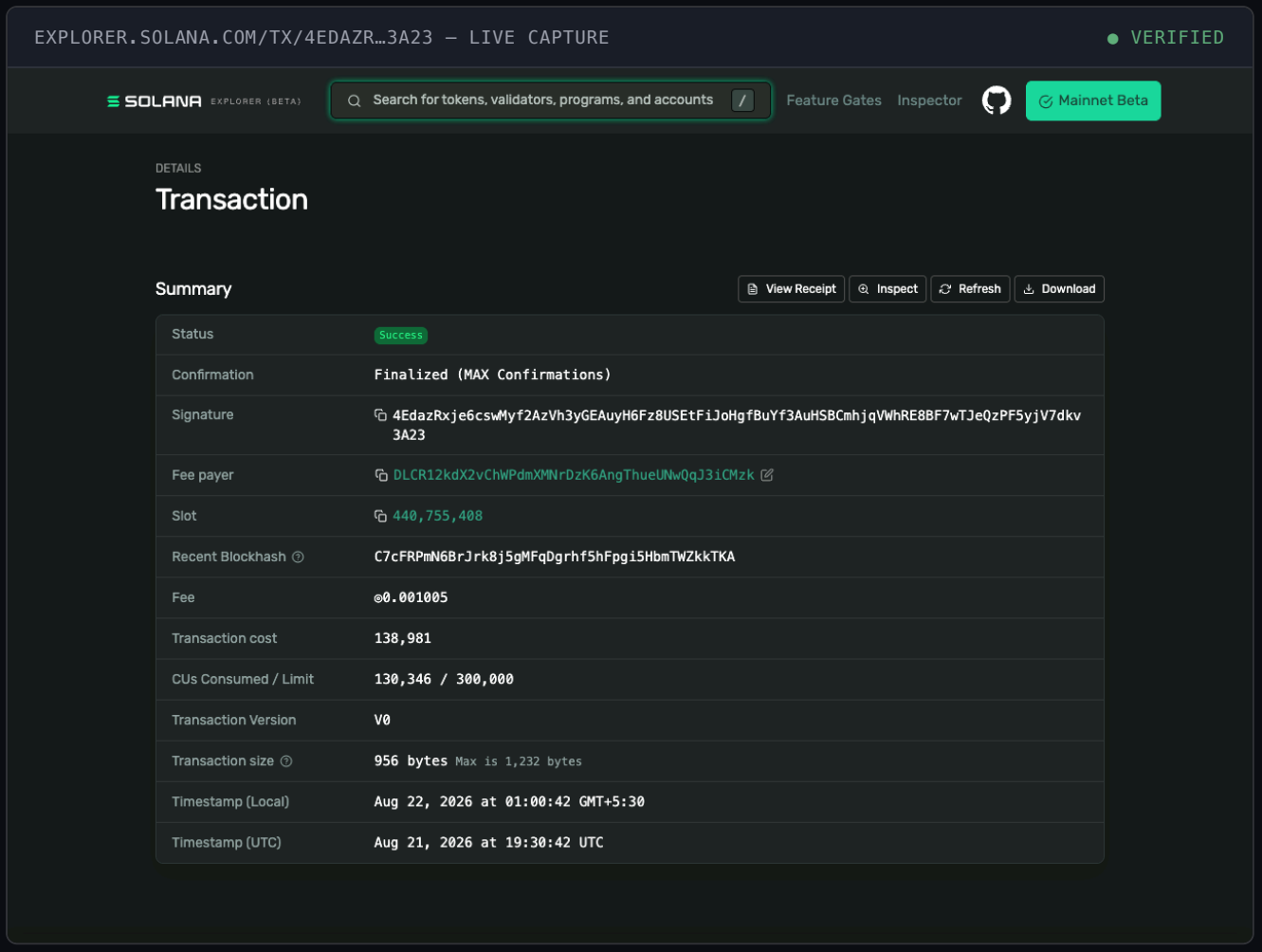


A.2

APPENDIX — INDEPENDENT BLOCK EXPLORER CONFIRMATION

The same transaction, on a source RUGAZE does not control

Solana Explorer (`explorer.solana.com`), operated independently of RUGAZE, showing the exact signature cited in §12 and the verification page at the front of this document. Status, slot, and signature match this document's own evidence exactly.



Captured during preparation of this document. Any reader can reproduce this exact check themselves — the link is provided on the verification page near the front of this document.