WHITEPAPER · VERSION 2.1 · JUNE 2026

ARCHON

Verifiable DevTools for Mantle

AI-assisted smart-contract auditing and receipt-calibrated gas optimization, with every report anchored on Mantle as a challengeable on-chain proof.

DETERMINISTIC AUDITS | RECEIPT-CALIBRATED GAS | ON-CHAIN PROOF

NETWORK	Mantle Mainnet · Chain ID 5000
PROOF CONTRACT	ArchonProofRegistry · 0xe7043e2ec95eF357FbBa3359BA2f1edb10cEAD2a · verified
AGENT IDENTITY	ERC-8004 Agent #97 · Identity + Reputation registries
SAFETY MODEL	Read-only analysis; the only on-chain write is user-approved proof logging
LIVE SURFACES	archonaudit.xyz · /docs · public proof verification · public gas leaderboard

CONTENTS

In this paper

READING CONVENTION

This document describes the live product as deployed at the time of writing. Capabilities that are planned rather than shipped are explicitly labeled as planned. That convention is itself part of Archon's design: a tool whose claims cannot be trusted in its own whitepaper cannot be trusted in its reports.

EXECUTIVE SUMMARY

The argument in one page

Smart-contract security terminates, today, in a PDF. A team pays a reviewer, receives a document, and every downstream user is asked to trust that document: its provenance, its completeness, and the competence behind it. None of those properties can be independently checked. Meanwhile the dominant cost of operating contracts on modern L2s has quietly shifted from execution to data availability — and almost no tooling measures that cost honestly, because the interfaces most tools rely on no longer reflect what chains actually charge.

Archon answers both failures with one system, native to Mantle Mainnet:

1. **An audit engine** whose findings are deterministic first and AI-explained second. Identical source produces identical findings; the model writes the human layer and never invents the evidence.
2. **A gas engine** whose savings are measured against on-chain receipt ground truth. Archon empirically rejected the legacy fee oracle after finding it under-reports Mantle's real charged DA fee by roughly 99.96%, and built a receipt-calibrated model in its place (Section 5, Table 1).
3. **A proof layer** that anchors every report as a deterministic hash on Mantle through Archon's own verified registry contract and its ERC-8004 agent identity — so anyone can re-derive the hash, re-check the record, and publicly challenge the work.

Trust the reproducible evidence, not the auditor's claim.

The thesis generalizes beyond Archon: the next generation of crypto DevTools will be judged by evidence quality. A useful tool must show what source was analyzed, under which assumptions, what was measured versus estimated, which proof belongs to which artifact, and how a third party can reproduce or dispute the result. Archon's entire architecture — queued workers, deterministic hashing, receipt calibration, on-chain anchoring, public verification and a challenge ledger — exists to keep that evidence trail intact from input to proof.

Archon is deliberately not positioned as a replacement for expert human review, formal verification, or protocol-specific audits. It is the verifiable layer underneath them: faster feedback for builders, stronger evidence trails for teams, and a public, machine-checkable record for the ecosystem.

01 · THE PROBLEM

Two unverifiable costs

1.1 — Audits are distributed as unverifiable PDFs

The dominant audit artifact is a static report that behaves like a final presentation rather than a verifiable record. A reader is expected to trust the brand on the cover. In practice, teams and ecosystems need answers to operational questions the format cannot provide:

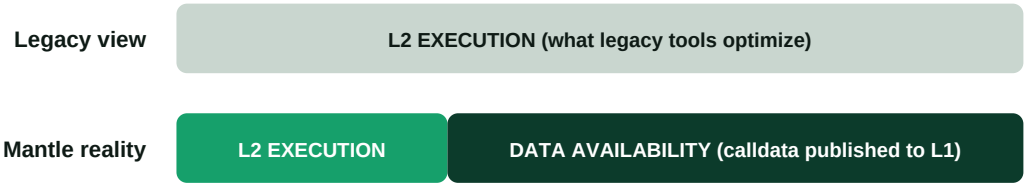
- Which exact source tree, commit, uploaded file, or verified contract was analyzed?
- Which compiler, network, and protocol assumptions were in scope?

- Which findings came from deterministic tools, which from rule engines, and which are model-assisted prose?
- Can each finding be traced to a file, line range, category, severity, and recommended fix?
- Can the report be hashed, exported, referenced, and independently re-checked later?
- Can anyone prove a public report is the same report the tool produced at a specific time?

PDFs fail these questions structurally. They can be edited, reposted, summarized incorrectly, or detached from the source that produced them. Even when the underlying analysis is honest, the delivery format destroys the evidence. Reputation accrues to auditor brands rather than to verifiable work — the opposite of the property this industry exists to provide.

1.2 — L2 cost is opaque and increasingly DA-dominated

On OP-stack-derived L2s such as Mantle, a transaction's true cost has two components: L2 execution gas and the data-availability fee charged for publishing its data. For calldata-heavy functions the DA component frequently dominates — yet nearly all optimization tooling was built for L1 execution semantics and ignores DA entirely. Developers optimize the visible fraction of their bill and ship the invisible one.



Illustrative split for a calldata-heavy function. Archon reports the per-contract split from measured data, so teams optimize the component that actually dominates their bill.

Figure 1 — Why execution-only gas tooling optimizes the wrong number on Mantle.

The problem is worse than neglect: the few tools that do estimate L1/DA fees rely on legacy oracle interfaces that no longer match what the chain charges. Section 5 quantifies this on Mantle Mainnet with real transactions: the legacy predeploy's prediction diverges from the receipt's charged fee by roughly 99.96%. A tool that trusted that interface would be wrong by three orders of magnitude while looking precise.

1.3 — These are the same failure

A security claim without reproducible evidence and a savings claim without measured ground truth are the same epistemic object: an assertion you are asked to believe. Archon's design principle is that every output — finding, patch, gas delta, report, proof — must either carry evidence a third party can independently re-derive, or be explicitly and visibly labeled as an estimate.

02 · THESIS

Verifiable DevTools: four commitments

Archon's thesis is simple: DevTools for crypto should be verifiable by default. A verifiable tool does not ask users to trust a screenshot or an opaque report; it produces artifacts that humans can inspect, machines can consume, and public infrastructure can anchor. Four commitments shape every architectural decision in this paper.

Commitment	Meaning in practice
Deterministic first	Detection is rule-based: solc compilation, Slither static analysis, Archon's Mantle rule engine, and AST-level detectors. AI explains and prioritizes findings; it never invents them. Identical input produces identical findings, which is what makes reports hashable and challengeable in the first place.
Measured, or labeled	Gas deltas come from Foundry measurement runs and on-chain receipts. Where measurement is impossible (pre-deployment analysis), the value is a calibrated estimate and is tagged as such in the UI, the API, and the stored record. The distinction is structural — it cannot be edited away in presentation.
Anchored and challengeable	Every report yields a deterministic SHA-256 hash and IPFS metadata, anchorable on Mantle. A public challenge ledger accepts scoped disputes against any artifact, referencing its on-chain proof. Verifiability without a feedback channel is theater; the challenge ledger is the feedback channel.
Read-only by default	Scanning never sends a transaction and never touches user funds. The single on-chain write in the product is explicit, user-approved proof logging — gasless via Archon's server agent, or self-custody from the user's own wallet.

How this differs from the status quo

Property	Traditional audit PDF	Generic AI scanner	Archon
Provenance of analyzed source	Stated, unverifiable	Usually absent	Hashed and recorded
Deterministic reproducibility	No	No (model-stochastic)	Yes, for all detections
DA-aware gas economics	Out of scope	Out of scope	Receipt-calibrated, per-contract split
Public verification	No	No	Hash re-derivation on a public page
Dispute channel	Email the firm	None	Public challenge ledger
On-chain identity & track record	Brand reputation	None	ERC-8004 Agent #97, append-only

Table — Property comparison. The point is not that PDFs or models are useless; it is that neither carries its own evidence.

03 · SYSTEM ARCHITECTURE

One pipeline, three artifacts

Archon is organized as a product system rather than a single analyzer. A Next.js application and a BullMQ worker run on one flat-cost VM; Postgres holds state; Redis runs the queues; Slither, solc and Foundry perform analysis and measurement; IPFS stores report metadata; and viem talks to Mantle RPC for all chain reads and the single proof write path. Heavy jobs are queued, cached by content hash, concurrency-capped and hard-timed-out. Nothing analysis-related runs serverless — a deliberate cost-discipline decision that keeps unit economics flat and failure modes debuggable.

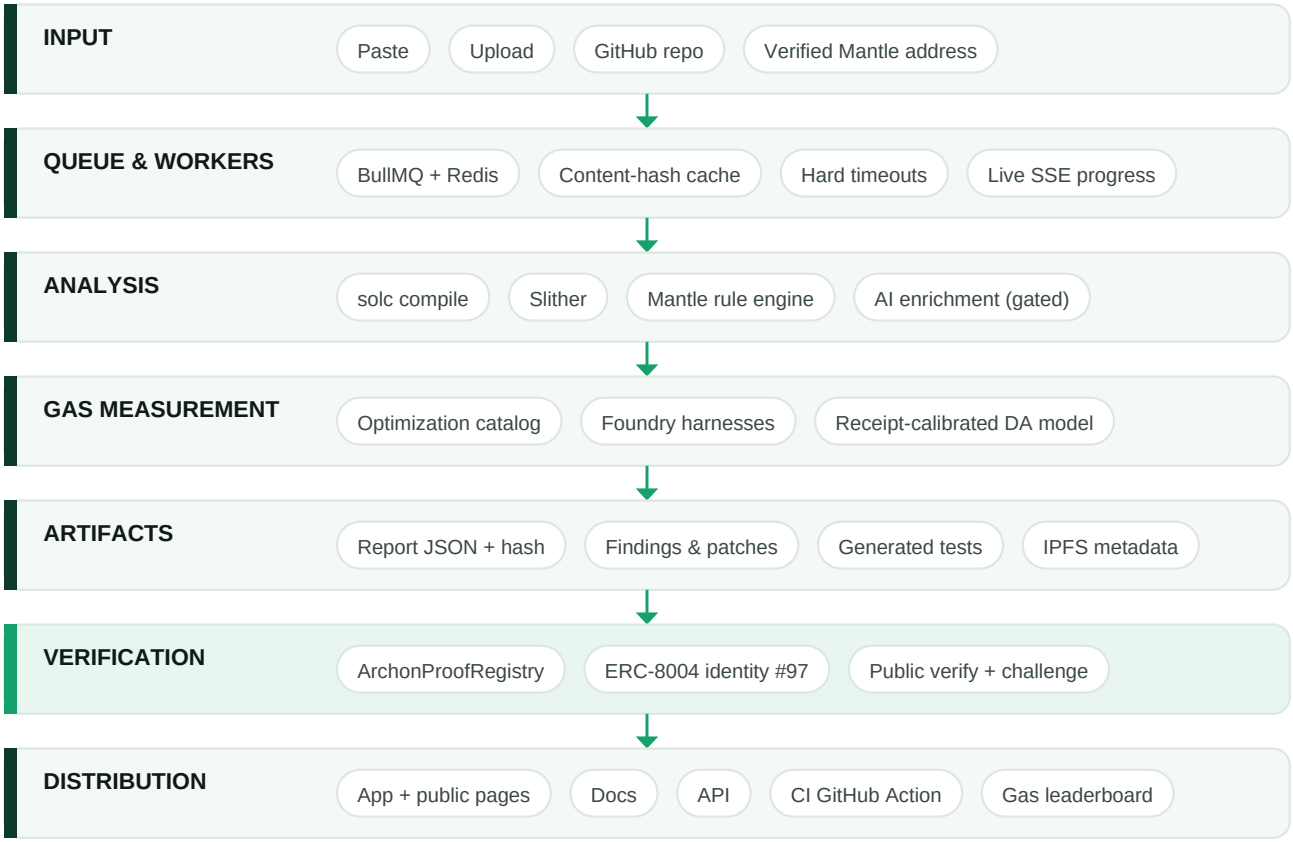


Figure 2 — The seven layers. Every layer can improve independently: gas pricing can be recalibrated without touching the proof model; the proof registry can evolve without breaking report exports.

The execution pipeline

Stage	What happens	Failure behavior
1 · Code parse	Normalize pasted source, uploads, a GitHub repository, or verified source fetched for a Mantle address; extract contracts, pragmas, imports; record scope metadata and the canonical source hash.	Structured error; nothing downstream runs on unparseable input.
2 · Static analysis	solc compilation as a truth gate, then Slither detectors plus Archon's deterministic rule set, producing findings with file/line anchors, categories, and confidence.	Compile failures surface clearly; unresolved imports degrade to reduced-mode AST rules with a visible banner.
3 · Mantle context fetch	Read-only chain context: bytecode, ABI, balances, protocol fingerprints, registry metadata.	Context marked unavailable; analysis continues without it.
4 · Protocol rule engine	Mantle-specific checks against known protocol surfaces (mETH, cmETH, USDY, Aave V3, Merchant Moe, Agni) and Mantle-relevant patterns such as sequencer-timing assumptions.	Rules skip cleanly when surfaces are absent.
5 · AI reasoning	gpt-4o-mini enriches deterministic findings with impact, exploit narrative, recommended fix and confidence — schema-validated, batched, and per-request time-bounded.	Timed-out or invalid batches fall back to deterministic templates; the pipeline never blocks on the model.
6 · Test generation	Foundry test harnesses mapped to findings; gas-diff harnesses for optimization claims; Mantle fork mode supported.	Tests are optional artifacts; generation failure does not block the report.

Stage	What happens	Failure behavior
7 · Report assembly	Canonical report JSON, deterministic SHA-256 hash, IPFS metadata, optional on-chain anchor.	Hashing is pure; assembly errors are loud, never silent.

Table — Pipeline stages. Findings stream to the client over server-sent events as each stage completes; the same spine, with specialized stages, backs both the audit engine and the gas engine.

04 · THE AUDIT ENGINE

Signal over commentary

The audit engine's job is a high signal-to-noise report a developer can act on the same day — and whose every claim can be traced back to a rule, a line, and a source hash.

4.1 — Input discipline

Credibility begins before analysis. Archon validates that input is Solidity, identifies contract names from the compiler's AST (never from raw-text heuristics), detects the pragma, builds a temporary compilation workspace with dependency remappings for common libraries (OpenZeppelin v4 and v5, solmate, solady, forge-std), and records the canonical source hash. Compilation is treated as a truth gate: if source cannot compile under a compatible compiler, Archon fails clearly rather than inventing findings on top of unparseable code. Exact pragmas are matched exactly; range pragmas accept newer compatible compilers.

4.2 — Deterministic detection

Detection combines Slither's analyzer suite with Archon's own rule engine. Coverage includes reentrancy and external-call ordering, access-control and tx.origin misuse, unchecked transfers and low-level calls, oracle freshness and timestamp-sensitive settlement, slippage enforcement, unbounded iteration and storage patterns that become operationally expensive, and Mantle-protocol-specific checks. Every finding carries severity, category, title, file and line range, source snippet, confidence, rule provenance, and remediation guidance. Cross-report triage deduplicates recurring findings into unique issues so repeated scans refine rather than multiply noise.

4.3 — AI as the communication layer

AI enrichment sits strictly on top of deterministic evidence. The model writes the human layer — why a finding matters on Mantle specifically, a plausible exploit scenario, a recommended fix — and its output is schema-validated on the way in. Enrichment runs in bounded batches with per-request timeouts; a failed or slow batch falls back to deterministic templates for exactly those findings, and the report ships. The model can improve understanding; it is never allowed to erase or replace the audit trail, and it is cached by content hash so identical bytecode never pays for the same explanation twice.

4.4 — Outputs

A completed audit produces a risk score with severity counts, an executive summary, the full finding list with anchors and fixes, optional generated Foundry regression tests mapped to findings, exportable canonical JSON, and proof metadata ready for anchoring. A good automated report helps a developer decide what to fix next; a great one also makes clear where the evidence came from, what the system did not prove, and how to reproduce the analysis path.

WHAT A REPORT IS

Archon reports are risk intelligence with provenance — not a certification that a contract is safe. That boundary is stated in the product, in the API, and on every public proof page.

05 · THE MANTLE GAS ENGINE

Receipt-calibrated DA pricing

5.1 — Three classes of gas evidence

Archon refuses to flatten all gas claims into one confident-looking number. Every figure in a gas report belongs to one of three classes, and the class is displayed wherever the figure appears:

Class	Definition	Source
Measured	A patch or scenario was compiled and executed; the delta was observed.	Foundry gas reports and harnesses; on-chain receipt l1Fee where a real transaction exists.
Estimated	A deterministic, receipt-calibrated model priced the change; no full measurement exists for this exact shape.	Calibrated DA model + static execution profiling; assumptions displayed inline.
Unpriced	The change improves code quality or theoretical behavior; Archon claims no quantified saving.	Rule catalog metadata.

Table — Evidence classes. A product that labels every suggestion as a precise saving will mislead users; Archon is designed to stay useful while saying "this is an estimate."

5.2 — Why Archon rejected the legacy oracle

The conventional approach to L1/DA pricing calls the gas-price-oracle predeploy's getL1Fee(bytes). Before trusting it, Archon validated it empirically: for real Mantle Mainnet transactions, the oracle's prediction was compared against the l1Fee actually charged in each transaction receipt.

Transaction (Mantle Mainnet)	Receipt l1Fee (charged)	Oracle getL1Fee (predicted)	Divergence
0x82d995...088ef	699,231,354,481,640 wei	313,344,079,825 wei	99.955%
0xb9ce87...1a7c5	6,874,261,528,561,290 wei	2,361,496,520,609 wei	99.966%

Table 1 — Oracle prediction vs. on-chain ground truth. The legacy predeploy under-reports Mantle's real charged DA fee by roughly 2,500x. Mantle receipts expose the richer post-migration fee fields (l1Fee, l1GasUsed, l1GasPrice, l1BaseFeeScalar, blob fee fields, daFootprintGasScalar, operator fees) that the legacy interface cannot reproduce. The full evidence is recorded in a public ADR in the repository.

5.3 — The receipt-calibrated model

Archon therefore prices DA from receipt ground truth. Where a real transaction exists, the charged l1Fee is read directly from its receipt — a measured value by definition, used as-is on leaderboards and deployed-contract views. For pre-deployment analysis, Archon maintains a calibrated model derived from recent live receipts' fee parameters, validated against the same ground truth, refreshed periodically, and always labeled as calibrated. The reporting model is explicit rather than hidden:

```
total_cost_per_call = (l2_gas_used x l2_gas_price) + da_fee
```

```
saving_per_call = (l2_gas_saved x l2_gas_price) + da_fee_saved
annual_saving_usd = saving_per_call x calls_per_year x MNT_USD
```

Reports expose every assumption behind these terms: the call-volume assumption, the MNT/USD assumption, the L2 gas-price source, the DA model mode (receipt-measured, calibrated, or degraded), and the calldata byte profile. Changing an assumption recomputes the figures; it never silently changes their evidence class.

5.4 — A worked example

Suppose Archon identifies a safe optimization on a frequently called function that saves 1,200 L2 gas per call and reduces DA cost by a calibrated 8,000 wei per call; the active L2 gas price is 30 gwei; the team's stated volume is 250,000 calls per year at an assumption of 1 MNT = 1 USD. The L2 saving is 1,200 x 30 gwei = 36,000 gwei per call; combined with the DA delta and annualized, the report shows roughly 9 USD per year. At 25 million calls per year the same patch is worth roughly 900 USD per year. The optimization did not change — the traffic did. This is exactly why Archon reports assumptions instead of pretending one universal savings number exists, and why deployment-footprint effects (bytecode size, calldata shape) are recorded separately from per-call execution savings.

5.5 — Detect, patch, prove

The deterministic optimization catalog covers storage packing and SLOAD caching, calldata reduction (the dominant DA lever), custom errors replacing revert strings, immutable and constant promotion, bounded unchecked arithmetic, loop hygiene, redundant initialization, visibility and comparison micro-patterns, and event data shape. Every opportunity ships as a concrete before/after patch with a safety note and confidence; "validate patch" compile-checks the suggestion and prepares a downloadable patched file; per-finding Foundry gas tests can prove the delta. Archon never edits a user's repository or deploys anything.

06 · THE PROOF LAYER

Identity, anchor, challenge

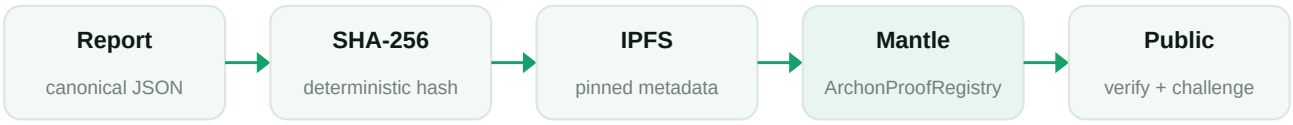


Figure 3 — The proof lifecycle. Each arrow is reproducible by a third party: the hash from the report, the metadata from IPFS, the record from the chain.

6.1 — ArchonProofRegistry

Archon's primary anchor is its own minimal, permissionless, verified contract on Mantle Mainnet:

```
logAuditProof(bytes32 reportHash, string metadataURI, uint8 riskScore, uint256 agentId)
```

The function records the deterministic report hash, the IPFS metadata URI, and the AI-derived risk score, keyed idempotently per report hash (a ProofAlreadyExists guard prevents duplicates). Because the registry is permissionless, both anchoring modes work without restriction: gasless, where Archon's server agent submits and pays; and self-custody, where the user's wallet signs and pays and is recorded as the logging address. Anchoring is, precisely, the publication of an AI inference result on-chain — the hash and the risk score are

outputs of the analysis pipeline.

6.2 — ERC-8004 identity

Archon is registered as Agent #97 in the official ERC-8004 Identity Registry on Mantle, with its agent file served from archonaudit.xyz/.well-known/archon-agent.json. The Reputation Registry records cross-agent feedback where author and subject differ. Archon's own report anchors live in ArchonProofRegistry precisely because ERC-8004 correctly forbids self-feedback — a constraint Archon discovered on-chain and designed around rather than worked around. Validation Registry integration is intentionally deferred until an official Mantle Mainnet address is published in the canonical ERC-8004 repository; shipping against an unofficial address would violate the nothing-fake rule.

6.3 — Public verification

Anyone — wallet or no wallet — can open a public report page, re-derive its canonical hash from the stored metadata, and compare it with the on-chain record. The verification surface shows the stored hash, the re-derived hash, and the match result side by side. External verifiers can consume the metadata endpoints without using the app at all.

6.4 — The challenge ledger

Verification should not be a one-way announcement. Archon includes a scoped public challenge ledger: anyone can submit a challenge against an audit report, a specific finding, a gas report, or an individual optimization. The challenge is recorded with target identifiers, rationale, optional evidence, status, timestamp, and a deterministic challenge hash; where the challenged artifact has an on-chain proof, the challenge references it. A challenge does not rewrite the original report and does not presume the challenger is correct — it creates a durable, hash-addressable record of dispute that travels with the artifact. Future versions may anchor challenge hashes on-chain and add staking; the current design deliberately favors a stable proof flow over deadline-driven contract surface.

07 · TRUST AND VERIFIABILITY MODEL

Exactly what is guaranteed

Archon's trust model has three layers — deterministic evidence, assisted interpretation, and public anchoring — and is designed around transparent limits. The table below is the contract Archon makes with its users.

Claim class	What is guaranteed	What is explicitly not guaranteed
Findings	Deterministic detections are reproducible from the same source; every finding carries rule provenance and line anchors; AI prose is confidence-scored and subordinate to evidence.	Completeness. Static analysis misses logic bugs; AI explanations can be wrong and should be reviewed.
Gas figures	Measured values come from Foundry runs and on-chain receipts; estimates are receipt-calibrated and labeled; all assumptions are displayed and adjustable.	Universality. Savings depend on traffic, fee conditions, and contract shape; fee models evolve as chains evolve.
Proofs	An anchor proves a specific report existed, with a specific hash and metadata, at a specific time, logged by a specific address, attributable to Agent #97.	Correctness of conclusions. A proof is provenance, not omniscience — the challenge ledger exists for disputes.

Claim class	What is guaranteed	What is explicitly not guaranteed
Safety	Analysis is read-only; Archon never holds funds, never executes user transactions, never stores wallet secrets in the browser; sign-in is a free SIWE signature.	Production-readiness of scanned code. Deployment decisions remain the team's responsibility.

THE BOUNDARY, STATED PLAINLY

The wrong claim is: "this contract is safe because a proof exists." The right claim is: "this report, with this hash and this metadata, was produced by this system at this time — and here is how to check it, and where to dispute it." Security tools that overclaim create false confidence; Archon treats honesty as a product feature.

08 · PLATFORM AND INTEGRATIONS

Where the artifacts live

Reports become more valuable when they are embedded in the places developers already work. Archon ships as a platform of surfaces around the same artifact model: human-readable reports for teams, machine-readable JSON and a signed verdict for integrations and agents, CI comments and fix PRs for pull requests, public leaderboard and address pages for comparison and discovery, continuous monitoring for deployed contracts, and on-chain references for verification. The table below is the current surface map.

Surface	Status	What it does
Application (archonaudit.xyz/app)	Live	Audit Studio, Gas Optimizer, findings, generated tests, Cost Guard telemetry, on-chain proof management, workspace settings; SIWE-gated; scans are read-only.
Sentinel	Live	Continuous monitoring of deployed Mantle contracts: bytecode, proxy-implementation, and admin-change drift detection on a scheduled worker; auto re-scan and finding-diff on drift; audit-freshness scoring; alerts via in-app and webhook. Each re-scan is anchorable, building a contract's audited -> drifted -> re-audited timeline.
Verified Build Attestations	Live	Deterministic source-to-deployed-bytecode matching (exact / metadata-aware / mismatch), anchored through ArchonProofRegistry. Closes the audit loophole: proves an audit covers the bytecode actually running on Mantle.
GitHub App + Autofix	Live	On pull requests, posts an updating check with new findings and an L2/DA gas diff; for safe, compile- and test-validated patches, opens an Archon fix PR. Policy via archon.config.json (severity gate, gas threshold).
Gas Observatory	Live	Public receipt-calibrated view of Mantle DA economics: effective DA cost per calldata byte, the execution-vs-DA split over recent blocks, and a standing oracle-vs-receipt divergence tracker (embeddable).
Agent Trust API + MCP server	Live	A signed, machine-grade contract verdict endpoint (recoverable to Agent #97) and an MCP server exposing scan, verdict, gas-report and proof-verify tools, so any AI agent can use Archon as its security sense.
Address intelligence pages + badges	Live	A permanent public security page for any Mantle contract (audit timeline, attestation status, Sentinel freshness, gas profile, challenges) plus embeddable shields-style badges; indexed for SEO.

Surface	Status	What it does
Public verification pages	Live	Report and proof pages with hash re-derivation, open to anyone without a wallet.
Public gas leaderboard	Live	Completed gas reports ranked by efficiency score and realized annual savings, with explicit sample labeling and links to the underlying reports.
Documentation (/docs)	Live	Product principles, audit and gas guides, proof verification, architecture, API reference, this whitepaper.
CI GitHub Action	Live	Gas optimization on pull requests with Mantle-aware L2/DA diff comments and configurable regression thresholds.
Public API	Live (expanding)	Scan, report, proof, gas, verdict and leaderboard endpoints with an OpenAPI reference; API keys live, webhooks expanding.
VS Code extension	Live (Open VSX)	Inline diagnostics, quick-fix code actions, and a per-function gas lens; available on Open VSX, with the VS Code Marketplace listing pending publisher verification.
Cloud provider layer	Live (pluggable)	AI enrichment and artifact storage run behind one interface with OpenAI, ELFA and Tencent Cloud (Hunyuan / COS) adapters; ELFA and OpenAI active, Tencent adapters built and ready pending credentials.
Staked challenges	Design (ADR 0014)	On-chain challenge-bond economics specified and reviewed; intentionally not deployed, behind a deliberate on-chain hard gate.

Table — Surfaces and status. Planned items are labeled planned here for the same reason they are labeled planned in the product.

09 · BUSINESS MODEL AND GTM

Built to outlive the hackathon

Archon's wedge is that security and cost are the two line items every Mantle team already pays for — and Archon quantifies both in dollars per year, with proof. Distribution follows the developer workflow rather than interrupting it: the first useful result requires no procurement, the CI product makes the tool part of the merge path, and the public surfaces compound into reputation.

Tier	Who it serves	Mechanics
Free	Individual builders	Capped-depth scans, public proof verification, public leaderboard. The leaderboard doubles as organic acquisition: every ranked contract links to a real Archon report.
Pro	Teams shipping on Mantle	Deep scans, full gas measurement runs, generated test suites, anchored proofs, workspace history; per-seat with metered heavy compute.
CI / API	Protocols and platforms	The GitHub Action gates pull requests on gas regressions; the public API embeds scan-and-verify into external pipelines. CI is the retention engine — a tool inside the merge path does not get churned.
Ecosystem	Mantle programs and partners	Sponsored audit campaigns, gas-optimization drives, public reputation surfaces built on anchored history.

On-chain reputation is the long-term moat. As reports, proofs, CI runs and accepted improvements accumulate against Agent #97, Archon becomes a reputation-bearing agent in the Mantle ecosystem: protocols can evaluate not just a single report but the public, append-only history of an agent's verified outputs

— and, eventually, other agents can do the same programmatically. The cost structure stays deliberately flat (one VM, queued and cached compute, AI only for explanation), so unit economics improve with scale: identical bytecode never pays for analysis twice.

10 · ROADMAP

From product to protocol

Horizon	Direction
Now (live)	Audit and gas engines on Mantle Mainnet; proof anchoring in both modes; public verification, leaderboard and challenge ledger; documentation, CI Action and GitHub App with autofix, generated tests; Sentinel continuous monitoring; verified build attestations; the Gas Observatory; the signed Agent Trust API and MCP server; public address intelligence pages and badges; the VS Code extension; and a pluggable cloud-provider layer (OpenAI/ELFA live, Tencent ready pending credentials).
Next	Reliability hardening across upload/repo inputs; webhook expansion and API-key tiers; deeper protocol packs; Tencent Cloud (Hunyuan inference, COS artifact storage) activated on provisioned credentials; VS Code Marketplace listing on publisher verification; richer Sentinel alert policies.
Later	ERC-8004 Validation Registry integration when an official Mantle address ships; staked challenges (ADR 0014) if approved; cross-agent attestations (auditor agents reviewing each other's anchored work); team workspaces and ecosystem dashboards.

The end state is an audit-and-optimization layer whose outputs are consumed not only by humans but by other agents — which is exactly what verifiable, machine-checkable reports make possible, and exactly what ERC-8004 identity was designed for.

11 · SECURITY, LIMITATIONS AND DISCLAIMERS

Honest boundaries

Archon is a developer security tool, not a guarantee of safety. Automated scans can miss vulnerabilities, misunderstand business logic, or overstate impact. AI-generated explanations should be reviewed by humans. Gas recommendations should be tested against the target codebase and deployment process. Proof records are provenance records, not certifications of correctness.

Archon is read-only by default: it analyzes source, produces reports, suggests patches, and prepares proof metadata. Users remain responsible for applying fixes, reviewing changes, running their test suites, and deciding whether code is production-ready. For high-value protocols, Archon should complement — never replace — expert review, formal methods, fuzzing, invariant testing, economic analysis, operational security review, and incident response planning.

Fee models are bounded. Mantle gas conditions, DA pricing, compression behavior and RPC data evolve; Archon exposes assumptions and refreshes calibration because cost estimates are decision support, not immutable truth. Public reputation surfaces are designed against metric-gaming: truthful labeling, real links, clear sample markers and transparent limitations are favored over inflated claims, and the leaderboard deduplicates by source hash.

Nothing in this document is investment, legal, or security advice.

12 · REFERENCES AND APPENDIX

Deployed references

Appendix A — On-chain and product references

Reference	Value
ArchonProofRegistry	0xe7043e2ec95eF357FbBa3359BA2f1edb10cEAD2a (verified on MantleScan)
Registry deploy transaction	0xb9ce87de86b212b91eb64012bbdab91014373da1f6d960470b340e1991a1a7c5
Example proof (logAuditProof)	0x82d99588e5f1bff33d618743025d598445493032637de25844a67aa8e88088ef
ERC-8004 Identity Registry	0x8004A169FB4a3325136EB29fA0ceB6D2e539a432
ERC-8004 Reputation Registry	0x8004BAa17C55a88189AE136b182e5fdA19dE9b63
Archon agent	#97 · agent file at archonaudit.xyz/.well-known/archon-agent.json
Application	https://archonaudit.xyz · documentation at /docs
Source repository	https://github.com/Franlinozz/Archon (Foundry workspace under contracts/)

Appendix B — External references

- ERC-8004: Trustless Agents — eips.ethereum.org/EIPS/eip-8004
- Mantle documentation — docs.mantle.xyz
- Solidity documentation — docs.soliditylang.org
- Slither static analyzer — github.com/crytic/slither
- Foundry — book.getfoundry.sh

Archon — Verifiable DevTools for Mantle · Whitepaper v2.1 · June 2026 · archonaudit.xyz