WORKING PAPER · MORESALAMANDER STUDIOLABS

The deterministic scaffold: a case study in compounding architecture

How one invariant — the model proposes, a deterministic gate decides — generalized across ten checkpoints without ever being rewritten, and became the substrate its own catalog now runs on.

Author — 0ne29 · Compiled by — Claude (Sonnet 5) · Period covered — 2026-01 to 2026-07-28

This paper traces one architectural idea across ten checkpoints of real, shipped work: the claim that reliability in an LLM system should live in a deterministic verification boundary, never in the model's judgment. The evidence for this claim is not that it sounds coherent — it is that the same invariant was re-implemented independently, under increasing stakes and across domains that share no code: a personal learning system, a story generator, a real-time heist simulation, a hosted teach-yourself-to-code reader, a 30-project physics curriculum, an autonomous-org framework, a crypto-opportunity hunter, a meta-org of hunters, a scoped Manager-engine extension, and — new in this revision — the session in which that autonomous-org framework stopped being nine separate instances of the pattern and became the single substrate all of them now sit on, under one name: Entropy AI. What follows is a synthesis of that record, compiled from persisted session memory and live repository state rather than raw transcript, with dates, test counts, and live-run costs cited wherever they were recorded — and with the places the pattern does not uniformly hold, or is not yet built, named plainly rather than smoothed over.

I	II	III	IV	V	VI
MyMaestro → myAIstro 01–05/26	story / House 05–06/26	Veritas 06/09–10	build-it	taichi-academy 07/14	Crypto Hunter 07/18–21

VII	VIII	IX	X
Opportunity 07/24	trust tiers scoped	Education Agency scoped, 07/25–26	Entropy AI, assembled 07/27–28

00 A methodological note

This account is reconstructed from dated memory checkpoints written at the end of past sessions, plus live repository state read directly at compilation time, not from a complete transcript archive. Where a figure is cited — a test count, a dollar cost, a date — it is because it was recorded or verified at the time, not because it was reconstructed after the fact. The paper's job is to trace the shape of the thesis's evolution, not to certify the current state of any one repository beyond what was actually checked while writing this revision.

a correction to this note, made honestly

The previous revision of this paper had no persisted source — it was authored as HTML inside a session's private temp scratchpad, printed to PDF, and the scratchpad was gone by the next session. This revision's own source (`paper_source.html`) is committed to this repository for exactly that reason: a paper arguing that verification requires a durable, checkable record should not itself depend on a session's temp directory surviving. See §12, "the paper's own source did not survive its own standard," for the fuller note.

01 The claim

The Deterministic Scaffold: an LLM proposes; it never decides. A separate, non-LLM gate — code that runs a test, checks a domain registry, diffs a claim against a ground-truth document, enforces a hard numeric floor — is the only path from "proposed" to "trusted." The corollary that recurs at every checkpoint below, phrased slightly differently each time: the grader can't be the thing it grades.

```
LLM proposes
a candidate claim
```

→

```
deterministic gate
checks against a real source of truth
```

→

```
trusted
or rejected, with evidence
```

02 Checkpoint I — the origin

two failures, January–March 2026, before myAIstro existed

The methodology did not begin as a methodology. It began as two separate, unrelated failures, two months apart, that only later turned out to be the same lesson from two directions. Building the Build It Publisher (a six-stage n8n workflow, Jan 2026) forced the discipline: named stages, one responsibility each, explicit data flow — the shape that later re-emerged as NDJSON event streams across nearly every project in this paper. MyMaestro, the first study tool, hallucinated — plausible wrong content into a learning aid, worse than nothing — and the fix was architectural: verification has to be a load-bearing layer, not a soft warning at the UI. Pipeline shape + verification at every boundary = the Deterministic Scaffold. Neither failure is speculative (commit log and file listing: Appendix A.1).

myAIstro is not the earliest instance in this history — it is the first project built after both lessons had landed: a five-stage ingestion pipeline feeding a validation hard-gate that must pass before anything persists, and three separate local models by role — one summarizes, one (chosen for its 128K context window) handles quiz generation and advisory queries, and a third judges, deliberately kept separate from both generators so the grader is never grading its own work. Its Classroom subsystem contains at least three more independent instances of the same discipline, shaped differently by what each risk actually requires:

Mechanism	LLM's role	The gate
CHECK	proposes options	exact-match check, 0 or 100, no partial credit
TYPING_PRACTICE	excluded entirely	code quoted verbatim from the SOT's own source

RAISE-HAND	answers a question	grounding report; 409 refusal if nothing to ground in

load-bearing distinction

None of these is "the" gate pattern — each is the pattern shaped by what the specific risk actually requires. The invariant held across this paper is fixed: an LLM's output is never self-certifying. The implementation is not fixed, and should not be.

03 Checkpoint II — the rule survives domain transfer

my-AI-story (shipped 2026-05-31, 88 tests) & The House Always Wins (begun 2026-06-08)

Two near-simultaneous instances, in domains that share nothing. My-AI-story: no episode is persisted or spoken until a pure-Python verifier confirms it matches the series bible. The House Always Wins pushed the rule into a real-time simulation — a per-frame pipeline (assess → propose → verify → commit → log) where three named, pure verifier functions are the only path from a crew's proposed action to a committed one: "the model proposes; Python disposes." This checkpoint also produced a deliberately engineered floor of irreducible uncertainty — trained crews win roughly ninety percent of the time, never one hundred, because a gate that lets skill hit certainty stops being suspenseful. The core simulation engine was originally built months before this project existed by name; the artifact outlived that conversation entirely.

04 Checkpoint III — naming the substrate

Veritas · P0–P4 · 2026-06-09 to 2026-06-10 · 13 → 31 tests

Veritas extracted the pattern and named it as a reusable primitive: a provider-agnostic model boundary, a `Gate` abstraction whose own docstring states the whole thesis in one line — *"The LLM proposes; the gate decides."* P2 sharpened this further: acceptance requires at least one hard gate to pass and every hard gate to agree. The clearest live proof of the thesis working as designed came from an ordinary run: `llama3.1:8b` shipped a correct `factorial()`; the QA agent — itself an LLM, therefore honestly a soft gate — wrongly flagged it. The system recorded the dissent and did not block the hard-verified code. P3 added a second claim: the organization learns from its own failures through deterministic recall alone, no weight update — `test_org_avoids_repeating_its_own_failure` is a real regression test (Appendix A.3).

05 Checkpoint IV — off the laptop

build-it · live at build-it-mauve.vercel.app

Build-it is the first public, hosted instance — a calm "Lego manual for code" reader for absolute beginners, built by the same adult-beginner who is its own first audience. The deterministic reader, the step-by-step verifier, and even a fully deterministic setup interview all work with zero API key. The one place an LLM appears only ever voices a verdict a deterministic check already reached. The thesis generalizes not just across domains but across how much infrastructure is available to run it.

06 Checkpoint V — scale

taichi-academy · all 30 projects shipped 2026-07-14

A thirty-project GPU-simulation curriculum, each built reference-first: a verified implementation and its tests exist before a single lesson fragment is written, and a deterministic checker — not the model — renders every verdict while a learner hand-types code. Thirty independent applications of the same discipline is evidence about the pattern holding under repetition without drifting: a raw five-point Laplacian unstable at a given timestep, a divergence/gradient mismatch that silently halved a fluid solver's effectiveness, a bistable climate-model state that had to be tuned around — each a real bug the gate structure caught.

07 Checkpoint VI — proof under real stakes

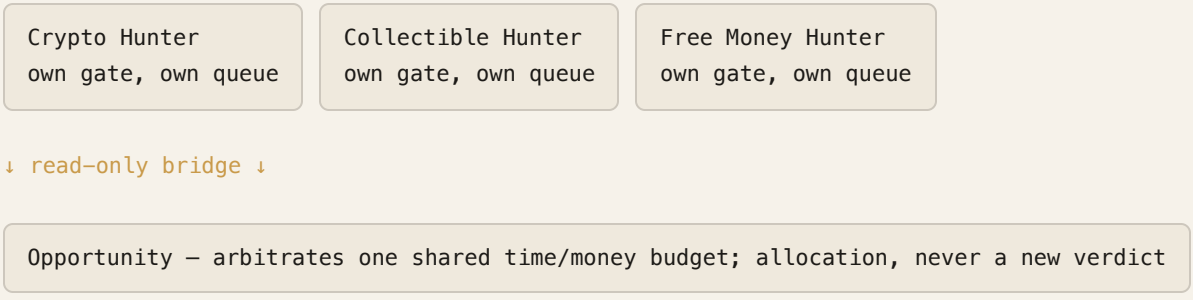
Crypto Hunter AI · Phase 0–4a · 2026-07-18 to 2026-07-21 · 15 → 47 tests

Crypto Hunter AI is the precise test of whether the thesis was a real invariant or an artifact of Veritas's own codebase — an independent, parallel implementation under real money and real adversaries, later bridged into Veritas's registry read-only, never re-judged. The gate laws here were found live, by real failures, and locked as regression tests the same day: evidence must be monotonic (a source that fails validation is excluded, never used to demote — except scam-listed domains, which hard-fail anywhere); negative evidence fails closed; re-filing under a name variant cannot evade a scam report. Cost is on the record, not estimated: $1.30 for the first full day-cycle, later roughly $0.12/day-cycle. The organization now runs on an unattended daily cron.

08 Checkpoint VII — recursion, one level up

Opportunity [Agency AI] · shipped 2026-07-24 · live-merged real data

Opportunity applies the same invariant to a new level of composition: "an agent organization of agent organizations." Crypto Hunter's reusable 80% became a shared substrate, `hunter-engine`, and two new independent engines — Collectible Hunter AI and Free Money Hunter AI — were built on it and live-validated the same week. Opportunity reads each engine's already-gated queue read-only, never re-deciding a verdict, and arbitrates one shared budget across all of them. First live run: eighteen real Crypto Hunter opportunities and five real Free Money Hunter opportunities merged into one genuine cross-domain allocation.



09 Checkpoint VIII — the substrate refines itself

2026-07-25 · scoped, not built

An explicit open question — whether the Hunter archetype generalizes into a Manager archetype that operates and grows an owned asset over time — was answered convergently from two unrelated starting points in the same session, both landing on a predicted action-executor-plus-approval-tier and metrics-feedback loop. The session also produced the four-tier trust spectrum (0·self-verifying, 1·consistency, 2·fact-bank, 3·floor+human), refining rather than repeating the gate-hard/debate-argued binary.

10 Checkpoint IX — Education Agency

scoped, 2026-07-25 to 07-26

The Manager-archetype candidate this paper's own closing suggestions (§13) recommended building first — the only one built from pieces that already exist, ship, and are tested. As of this revision it remains scoped, not built; that gap is named here rather than implied closed.

11 Checkpoint X — Entropy AI, assembled

Veritas · 2026-07-27 to 2026-07-28 · 411 → 502 tests · nine commits

Where checkpoints III–IX each proved the invariant held in a new domain or a new level of composition, this checkpoint is different in kind: it is the session in which the studio's own catalog — nine separate repositories, each independently proving the same invariant — was given one name for what all of them had been checkpoints of, and Veritas itself was rebuilt to actually behave like the substrate the closing note of the previous revision only described in prose. Four real subsystems shipped, each independently gated and tested, none of them a new domain — all of them Veritas's own substrate, deepened:

The Collector — Entropy's own DataHub, fed by admission

Every engine in the catalog already kept its own DataHub-shaped store (`hunter_engine.store.DataHub`, crypto-hunter's own copy, Opportunity's `OpportunityHub`). The Collector reads across all of them read-only and re-admits a record into Veritas's own SOT only if it passes a *structural* admission gate — was the crossing legitimate (evidence exists, is well-formed) — never re-judging whether the underlying claim is true, which is already each source engine's own job, one level down. This is the recursive "engine of engines" diagram from §08, made literal at the data layer rather than only at the decision layer.

All three Hunter engines, one generalized bridge

Collectible Hunter AI and Free Money Hunter AI joined Crypto Hunter as full Veritas orgs — not by copy-pasting Crypto Hunter's bridge a second and third time, but by extracting `orgs/hunter_engine_bridge.py`'s `run_hunter_engine()` once and pointing three thin registry entries at it. The same DRY-vs-copy-paste choice this repo had already made once for its four presets (§07's `_run_grounded_preset`), applied again rather than repeated as a special case.

API Key Tracker — a security boundary, not a demo

Inventory and rotation metadata live in Veritas's own SQLite; real secret values live only in the macOS Keychain, entered exclusively via a CLI using `getpass` — never echoed, never a shell-history argument, never a browser round-trip even to localhost. This is a genuine new trust-tier-3 (hard floor) instance in the sense of §09's spectrum: an irreversible, outward-facing class of action (credential handling) given machinery proportional to its stakes, designed before any code touched a real key, not after a leak.

Second-brain deepening, and a caught bug worth keeping in the record

Three additions to Veritas's own institutional memory (`engine/memory.py` 's `MemoryStore` — already, independently, the same shape as myAIstro's SOT, not an import of it; see §12's convergence note): a grounding check (`engine/grounding.py`) verifying a memory record's body actually stays anchored to what its own `informed_by` provenance claims informed it; an Obsidian vault export (`engine/memory_export.py`) making the memory graphable; and a recall-preview endpoint surfacing what `format_lessons()` would inject into a new proposer's prompt, previously invisible. Verification caught a real bug before it shipped: the first vault-export pass reported `files_written: 221` but only 50 files existed on disk — records sharing an identical title (routine; many runs produce a "[storyboard] by storyboard-artist-agent") were silently overwriting each other's files. Fixed by keying filenames on the record's own id, not its title, and confirmed 221 written = 221 on disk afterward. This is the same category of evidence §07 cites for Crypto Hunter's gate laws: a real failure, caught, fixed, and kept in the record rather than quietly smoothed over.

`named, not built — the choreography model`

A fifth thread from this session is explicitly *not* shipped: the proposal that engines become ephemeral, per-run Docker containers — "organized chaos by trust," a fireworks show where Entropy AI is the choreographer, not the pyrotechnics. Reasoned through carefully (does an ephemeral container interfere with a DataHub-shaped store? no, provided the store is bind-mounted rather than living in the container's own layer — compute is ephemeral, storage is not), but zero lines of Docker-orchestration code exist yet. Included here for the same reason §10's Manager archetype was included while still scoped: a thesis's honest edge includes what it has reasoned through but not yet built, named as such.

12 The wider catalog — where the pattern holds, and where it honestly doesn't

The checkpoints above are not the whole studio; they are the load-bearing ones. Several other shipped tools carry the identical shape — AccessGuard's deterministic axe-core scan with an LLM only voicing the branded report, amigurumi-ai's crochet-geometry engine validating every spec before an LLM interview can touch it, my-AI-scene's per-boundary verifier gates, my-AI-beats' CLAP-gated audio sections. SprinklerPro3D's quality-gate is worth naming specifically because it is currently failing on all five test yards — an honest, unresolved result, not a cherry-picked success.

Others in the catalog — Particle Life, universe-forge — are real-time physics and generative art, verified stable over long runs, but carry no LLM-proposal-versus-gate split at all; there may be no LLM in the loop to

distrust. Counting them as instances of this specific thesis would be the overclaim this paper is trying to avoid elsewhere.

13 What "provable" means in this record

Every checkpoint above is cited with something falsifiable: a locked regression test born from a real bug, a dollar figure from an actual live run, a dissenting verdict left on the record, or an independent re-derivation landing on a predicted structure. None of it rests on an assertion that the architecture "feels right."

a new instance of the strongest evidence class

§04's Gate abstraction and §02's Classroom subsystems are this paper's strongest evidence — independent re-derivation, sometimes retroactive to a target that didn't yet exist. Checkpoint X adds a fresh instance of exactly that class, found by directly reading the code rather than asserted: five separate persistence stores across this catalog — `hunter_engine.store.DataHub`, crypto-hunter's own copy of it, Opportunity's `OpportunityHub`, and Veritas's own `CollectorStore` and `KeyTrackerStore` (both built this session) — converged on the identical shape without being mechanically copy-pasted: plain SQLite, no ORM, schema-on-init, and — load-bearing, not incidental — per-call connections rather than one held instance. That last detail was independently re-discovered as a bug fix in this very session (a `sqlite3.ProgrammingError` under FastAPI's threadpool, fixed by switching `CollectorStore` to per-call connections, matching the pre-existing `AccountStore` / `QuotaStore` convention it hadn't been built to match on purpose). Five stores landing on the same shape, one of them re-deriving the same fix for the same underlying reason without copying it, is the same category of evidence as §04's Gate docstring generalizing across codebases that share no code.

what would not count

A design that sounds elegant, a pattern applied uniformly "for consistency," or a finding manufactured to make a review look thorough. The record above deliberately excludes anything that could not survive being asked "grounded in what, exactly?"

14 Limitations and open threads

- The persistence-layer name is still not uniform across the lineage — Veritas calls it `MemoryStore`, the Hunter family calls it `DataHub` — stated plainly rather than smoothed into false consistency, even after §13's finding that the shape underneath both names converged independently.
- The Manager archetype (Career, YouTube, Commerce) remains scoped at the architecture level only — no code exists yet for any of the three, unchanged since the previous revision.
- The choreography model (§11) is reasoned through, not implemented — zero Docker orchestration code exists; the claim that a bind-mounted DataHub survives an ephemeral container cleanly has not been tested against a real container run, only argued from the existing file-based read-after-run contract.
- Veritas's hosted wedge has every primitive except billing, which remains explicitly deferred pending a real payment integration — unchanged since the previous revision.

- The four-tier trust spectrum (§09) has still not been stress-tested against a domain nobody in this thesis has thought about yet — the same open item as the previous revision, named again rather than allowed to quietly drop out because it wasn't re-addressed.
- SprinklerPro3D's quality-gate is, as of its last recorded state, failing on all five test yards — included deliberately, since a paper arguing for grounded evidence cannot omit its own unresolved failures.
- **The paper's own source did not survive its own standard, until now.** The previous revision existed only as a rendered PDF; its HTML source lived in a session's private temp scratchpad and was gone by the time this revision began. That is precisely the kind of non-durable, unverifiable state this paper argues against everywhere else — and it was true of the paper itself for one full revision before anyone (including the author of this sentence) noticed. Fixed this revision by committing `paper_source.html` to the repository as the actual, durable source of truth. Named here, not quietly corrected, because a paper about verification failing its own standard once is itself evidence worth keeping on the record rather than erasing.

15 On the collaboration itself

This paper is also an artifact of how it was produced. It was written by the user (one29) and an instance of Claude, working from a shared session, and the division of labor held a consistent shape across every checkpoint: the user supplied the vision, the direction, and — repeatedly — the correction. This revision adds one more instance to that record, worth stating plainly rather than only implying: the user's own architectural reframe this session — "Entropy is exactly what it is called, organized chaos by trust," describing engines as disposable Docker containers choreographed like a fireworks show — is the actual origin of §11's choreography model. Claude's contribution was reasoning through whether that model interfered with the existing DataHub contract (it doesn't, provided storage is bind-mounted) and naming the convergence with established orchestration patterns (Airflow, Argo, CI) as independent validation, not the origin of the idea itself. The user also corrected a real imprecision mid-session: an earlier framing suggested Veritas's second brain needed "myAIstro's second brain" imported into it; the user pointed out, correctly, that Veritas's own `MemoryStore` already independently is a second brain, tested, and only needed further engineering — not an integration. §13's DataHub-convergence finding exists because that correction was taken seriously enough to go verify, rather than politely agreed with and dropped.

16 Closing note

Veritas was never the top of this stack — it is the substrate every level sits on. What the previous revision of this paper could only describe in prose — "the discipline that was always the how becomes, here, the what" — this revision can point to as shipped code: a Collector that makes the recursive "engine of engines" diagram literal at the data layer, three Hunter engines unified behind one bridge instead of three copies, a key tracker that treats credential handling with the trust-tier machinery its stakes actually require, and a second brain that checks its own memory trail rather than merely accumulating it. Entropy AI is not a tenth checkpoint sitting beside the other nine — it is what checkpoint X's own subject line says plainly: assembled. The thesis became the product; this revision is the record of the product starting to behave like one.

17 Appendix A — independent evidence

Exhibits only, no narration. Every commit hash and test count below was read directly from the repository or run live during this compilation. Where the underlying record has a real gap, that gap is named rather than filled. A.1–A.8 are carried over unchanged from the previous revision (re-verified, not re-run, for this update); A.9 is new.

A.1 · the two founding failures, verified

```
$ git -C ~/my-maestro log --format="%h %ad %s" --date=short
bfc0d77 2026-03-13 feat: initial commit
08ebdb1 2026-03-13 Initial commit from Create Next App
# Both commits are scaffolding. No feature commit exists.

$ ls ~/Downloads | grep n8n
n8n-build-it-publisher.json          # Jan 2026, per the studio's own account
```

A.2 · myAIstro — Classroom commits

```
be40354  classroom: TYPING_PRACTICE beat — verbatim-code typing drill
2432548  highlights (h2 fix): split multi-line green highlights into per-line mastery
goals
fde0cc8  highlights (h2 mvp): user-authored "this matters" markers feed Classroom
mastery_goals
3712add  H0 verification + tightened mastery binding (path A)
e123914  mastery goals (h0): deterministic per-lesson "what to master" field
```

A.3 · Veritas — P0–P4 commits + live test run (as of the previous revision)

```
5049021  Phase 4 — the Hub (local-first): control plane over the real engine
65dedae  Phase 3 — the org learns: failure retrieval informs new work
7577fed  Phase 2 — the cast: Security, QA, Validation, and hard/soft honesty
4feae34  Phase 1 — the first honest run: goal -> spec -> code, gated
4c81041  Phase 0 — the spine: Artifact, Gate, Memory, Run
def test_org_avoids_repeating_its_own_failure(tmp_path):   # tests/test_learning.py:79
$ .venv/bin/pytest -q
411 passed, 8 skipped (needs a running Docker daemon), 1 warning in 43.07s
```

A.4 · Crypto Hunter AI — gate-law commits, test names, live run

```
9bc3225  gate v1.1: monotonic evidence semantics + RDAP registrable-domain fix
d82b776  phase 1: scam detection + scoring rubric + ranking + outcome tracking
6bd996f  phase 2: daily mission (profile-driven, deterministic) + org dashboard
0cfb9e5  provider abstraction: OpenAI build-mode + two gate laws captured live
42886c3  phase 3: the debate goes live + 4 new scout beats + Veritas org-ification
b6f510c  phase 4a: on-chain sense (keyless) + daily autonomous run
```

```
def test_evidence_is_monotonic():                                    # tests/test_gate.py
def test_scam_taint_is_the_monotonicity_exception(legit_spec):       # tests/test_gate.py
$ .venv/bin/pytest -q
70 passed in 0.30s   # re-run 2026-07-28; was 47 passed at the previous revision
```

A.5 · Opportunity — meta-layer commit + live run

```
2c67b92  The meta-layer, v1: read-only bridges + deterministic cross-engine allocation
def test_shared_budget_is_genuinely_shared():          # tests/test_mission.py:19
def test_picks_across_engines_by_score():               # tests/test_mission.py:5
def test_empty_engines_produce_an_honest_empty_allocation():  # tests/test_mission.py:47
```

A.6 · build-it & taichi-academy — commits

```
build-it:
  d5f84c3  Add the setup interview — the front door for true beginners
  3f2efe4  Finish the page-aware helper (chat phase 2)
taichi-academy:
  212ba4a  30-universe-sandbox: direct N-body gravity — galaxies, black holes, collisions
  a575620  29-earth-simulator: energy-balance climate with seasons, ice caps, a biosphere
```

A.7 · The House Always Wins — JS test suite (file listing, verified present)

```
game/test/core.test.mjs  game/test/detection.test.mjs  game/test/guards.test.mjs
game/test/mechanics.test.mjs  game/test/navigation.test.mjs  game/test/obstacles.test.mjs
game/test/staff.test.mjs
```

A.8 · documented example failures (the gate catching something real)

- Veritas P2: the QA agent wrongly flagged a correct `factorial()`; recorded as a soft dissent, did not block the hard-verified code.
- Crypto Hunter, live day one: a scam-listed domain and an unresolvable RDAP report both had to be handled as hard-fail and reject-not-shrug respectively.
- taichi-academy project 01: a raw five-point Laplacian was unstable at the lesson's chosen timestep — fixed with a nine-point stencil, documented as a teaching beat.

A.9 · Checkpoint X — commits, test counts, and the caught bug (new this revision)

```
$ git -C ~/MoreSalamander/veritas log --oneline -9
7b0e7f7  Deepen Veritas's own second brain: memory-trail grounding, Obsidian vault export,
recall preview
b415862  Clicking a Hunter engine card opens the real app, via Launchpad's live status
e163140  Give Prompt Studio its own first-class section
e16c850  Default the shell hero to a from-below view; relabel as Entropy AI powered by
Veritas Dynamics
dbce176  Reclassify the Hunter engines as engines, not Labs
a9a69d6  Give Collector evidence the same "Explain it to me" treatment as the engines
```

```
f02051e  Make Collector pending items expandable to their full record
781ff3d  Add the API key tracker: inventory + Keychain-backed vault, no secret ever in-app
70b444c  Register collectible-hunter and free-money-hunter as Veritas orgs
3a47cb3  Add the collector: Entropy's own DataHub, fed by admission

$ git -C ~/MoreSalamander/hunter-engine log --oneline -1
41f7942  Add usage_log key_id attribution + fix missing .env gitignore entry

$ .venv/bin/pytest -q   # ~/MoreSalamander/veritas, 2026-07-28
502 passed, 8 skipped (needs a running Docker daemon), 1 warning in ~40s
# was 411 passed at the previous revision — 91 new tests from this checkpoint alone

# the caught bug, kept in the record rather than silently fixed:
$ curl -s -X POST localhost:8099/api/memory/vault/sync
{"vault_path": "...", "files_written": 221}
$ curl -s localhost:8099/api/memory/vault/status
{"vault_path": "...", "exists": true, "file_count": 50}   # BUG: 221 != 50, filename
collisions
# fixed by keying vault filenames on the record's id, not its title; re-verified:
{"vault_path": "...", "files_written": 221}
{"vault_path": "...", "exists": true, "file_count": 221}  # fixed: 221 == 221
```

Compiled from: project_myaistro, project_myaistory, project_housealwayswins, project_buildit, project_taichi_academy, project_veritas, project_cryptohunter, project_opportunity_agency, project_scaffold_recursion, project_trust_tiers, project_datahub_hackathon, project_moresalamander, project_entropy_ai — session memory plus live repository state, MoreSalamander StudioLabs, 2026-07-28.