

COMPANION TO THE 52-SHEET WORKBOOK — EVERY SHEET EXPLAINED

# Lemur Audit Workbook — Companion Document

**Title:** *“Hi Umma — Can you analyze the Netflix Lemur GitHub repo for any vulnerabilities...”***Format:** XLSX, 52 sheets, 1.29 MB structured content (1,288,348 bytes)**Produced by:** Umma · Thinking Studios · **Generated:** 2026-05-29 (3h 20m end-to-end, from a single prompt)

This document explains what each of the 52 sheets contains, how it was produced, and how it connects to the rest of the workbook. The workbook is **not** a flat report — it is a recorded audit where each sheet preserves a specific stage of the work. Read this first; the workbook makes sense after.

## 1. How the workbook is organized

The 52 sheets fall into **five audit phases**. Each phase is a distinct workstream operating on a specific part of the audit:

|                                 |                               |
|---------------------------------|-------------------------------|
| Phase 0 — Baseline / Pin        | sheets 0, 1, 3, 4, 5, 6, 7, 8 |
| Phase 1 — CVE / advisory ingest | sheets 13, 14, 15, 16         |
| Phase 2 — Threat model          | sheets 17, 18, 19, 20         |
| Phase 3 — Specialist analyzers  | sheets 2, 9–12, 21–29         |
| Phase 4 — Synthesis             | sheets 30–51                  |

Each phase feeds the next; the synthesis phase reconciles findings across all upstream phases into a single unified set. **The 36-column master findings table (sheet 43) is the master view** — every other findings sheet is an upstream feeder into it.

## 2. Phase 0 — Baseline / Pin

Establishes the codebase footprint and the already-patched advisory baseline. Every subsequent analysis grounds against this.

**Sheet 0 — base\_derived** (33 × 4)

Derived baseline references — the canonical reference IDs and version pins extracted during the codebase pin.

#### Sheet 1 – `synthesis_crosscheck` (5 × 14)

Synthesis cross-check rows — metadata recorded during the synthesis pass for closure and traceability.

#### Sheet 3 – `coverage` (575 × 4)

**One row per file in the Lemur codebase.** 575 files inventoried for coverage tracking — whether the file was reviewed, by which workstream, and at what depth. This is the **denominator** for every coverage claim later in the audit.

#### Sheet 4 – `endpoints` (65 × 8)

**Complete API endpoint inventory** — all 65 endpoints exposed by Lemur, each with HTTP method, route, the file/class/function implementing it, auth requirements, and input/output shapes. The master endpoint map; the compounding-chains and endpoint-rollup sheets build directly on it.

#### Sheet 5 – `registered_plugins` (33 × 3)

The 33 plugins registered in Lemur's `setup.py` `entry_points`. Plugin registry is a critical attack surface for systems that dynamically dispatch to plugin code — each plugin is a potential pivot point.

#### Sheet 6 – `non_registered_plugins` (2 × 2)

**Plugins that exist in the codebase but are NOT registered** in `entry_points` — a substantive finding. Each row identifies a non-registered plugin and why it matters.

#### Sheet 7 – `files` (575 × 8)

Full file manifest — every file with size, type, line count, last-modified date, and module categorization. The richer companion to Sheet 3's coverage row.

#### Sheet 8 – `critical_baseline_findings` (1 × 10)

**The already-patched critical finding at the pin** — the authorization bypass patched in Lemur 1.9.1. Its `downstream_guidance` field tells the audit's own analyzers how to treat this finding so they don't re-report it as novel. Provenance-aware reasoning: the audit knows what it already knows.

### 3. Phase 1 – CVE / advisory ingest

---

External advisory correlation — ingests public security research and matches it against the pinned codebase.

#### Sheet 13 – `lemur_native_advisories` (6 × 19)

6 Lemur-specific GitHub Security Advisories, each with advisory ID, CVE ID (if assigned), CVSS score, affected version range, fix commit, narrative, exploit prerequisites, and mitigation guidance. The known-quantity baseline against which novel findings are compared.

Sheet 14 – `public_research` (10 × 3)

10 entries of broader public security research (disclosure write-ups, talks, posts) bearing on Lemur’s threat surface or its dependency tree. Each row pairs a source with its relevance to the current pin.

Sheet 15 – `python_dep_findings` (29 × 5)

29 findings in Lemur’s Python dependency tree — each identifying a package, the version Lemur pins, the vulnerabilities affecting it, and **whether the vulnerable code path is actually reachable** from Lemur’s call graph. The reachability determination is what separates a serious finding from background noise: generic SCA tools produce counts in the hundreds; reachability filtering brings them down to the actionable.

Sheet 16 – `frontend_findings` (4 × 8)

4 frontend (JavaScript/CSS) security findings. Lemur’s frontend is comparatively small and the security boundary is mostly backend-enforced; this sheet captures the few frontend-specific concerns.

## 4. Phase 2 – Threat model

---

Systematic threat modeling using STRIDE. Its output grounds every specialist analyzer’s prioritization.

Sheet 17 – `assets` (17 × 5)

17 named assets the system protects — each with what it represents, its sensitivity tier, its location in the architecture, and its primary owner. Assets are referenced by ID (A1–A17) throughout the audit.

Sheet 18 – `trust_boundaries` (14 × 5)

14 trust boundaries — each defining the boundary, what crosses it, what validation occurs at the crossing, and what attack opportunities exist if validation fails (authenticated/unauthenticated, user/admin, authority/destination, plugin/core, etc.).

Sheet 19 – `actors` (10 × 4)

10 named threat actors modeled against — e.g., unauthenticated external attacker, authenticated read-only user, authenticated operator-tier user, malicious authority admin, supply-chain attacker, insider.

Sheet 20 – `stride_matrix` (26 × 5)

**Full STRIDE threat matrix.** 26 rows covering Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege — each applied to specific Lemur components, with the specific threat scenarios and the affected asset IDs. A representative row (Spoofing × AuthN endpoints) lists threats such as “JWT-secret leak → arbitrary user impersonation”, “OAuth redirect\_uri / state mishandling → account takeover”, “brute-force / credential stuffing on /auth/login”, and “CORS='\*' + supports\_credentials=True → cross-origin token exfil if reconfigured.” Each threat in each cell is a specific, testable claim.

## 5. Phase 3 – Specialist analyzers

---

Each specialist analyzer owns one workstream. They run in parallel against the codebase, each producing a findings sheet (and supporting context sheets).

### Sheet 2 – ad-hoc findings (1 × 26)

A finding discovered out-of-pipeline during the run, captured with the full 26-column finding schema – i.e., discovered during synthesis rather than assigned to a specialist a priori.

### Sheet 21 – config / secrets findings (48 × 7)

48 configuration- and secrets-level findings (leaner 7-column schema, consistent with config-level issues).

### Sheet 22 – logging / observability findings (39 × 9)

39 findings on logging, audit-trail, and observability concerns. Sheet 23 (files\_reviewed, 53 × 3) records the coverage supporting it.

### Sheet 24 – authorization matrix (91 × 12)

The per-endpoint authorization analysis – every authenticated route × every authorization check × the actual enforcement path. The 91 rows are endpoint-permission pairs. Sheet 25 (files, 47 × 3) lists the auth-related files reviewed in detail.

### Sheet 26 – authorization findings (25 × 22)

25 authorization findings × 22 attributes each – the deepest schema among the specialists, reflecting the complexity of authorization analysis. Referenced extensively in the compounding-chains sheet.

### Sheet 27 – input-validation findings (20 × 18)

20 findings on input validation, serialization, and deserialization.

### Sheet 28 – certificate-issuance findings (27 × 20)

27 findings × 20 attributes. Lemur's core function is certificate issuance, so this is *the* security-critical workstream. Referenced repeatedly across the compounding chains.

### Sheets 9, 11, 29 – further specialist findings (23×21, 20×20, 19×18)

Additional specialist workstreams, including a cross-cutting lineage analysis that traces issues across paths rather than per point. Sheets 10 and 12 (coverage\_log, 82 × 3 each) record that workstream's coverage, sampled at two points in the run for verification.

## 6. Phase 4 – Synthesis

---

The synthesis phase reconciles all upstream findings into a unified set, identifies cross-workstream chains, performs redaction, and prepares the final handoff artifact.

#### Sheet 30 – chains (6 × 17)

Initial chain analysis — candidate cross-finding chains with their evidence, before the endpoint-correlation pass.

#### Sheet 31 – coverage grid (238 × 17)

The master coverage matrix — every finding × the workstreams that touched it × the coverage status. The 238-row count matches the final master findings count (sheet 43).

#### Sheet 33 – compounding\_chains (6 × 7)

**The high-value sheet — six named cross-file attack chains.** Each chain is constructed by composing findings from multiple specialist workstreams; each has a name, the originating endpoint(s), the full finding chain, a threat tier, and a narrative explaining why it matters.

| ID     | Name  | Endpoint                                   | Tier     |
|--------|---|--|----------|
| CC-001 | Cross-tenant misissuance + key exfil        | /api/1/certificates [POST]                 | T1       |
| CC-002 | Upload-bypass + plugin-sink                 | /api/1/certificates/upload [POST]          | T1       |
| CC-003 | Pending-cert tamper → upload (two-stage)    | /api/1/pending_certificates/[PUT]→[POST]   | T1       |
| CC-004 | Recon → weaponize (plugin enumeration)      | /api/1/plugins [GET] → /api/1/destinations | T1       |
| CC-005 | Revoke state-drift (operator DoR)           | /api/1/certificates/{}/revoke [PUT]        | T2       |
| CC-006 | Cert export bypass + openssl argv injection | /api/1/certificates/{}/export [POST]       | (varies) |

*Each chain’s narrative captures an insight that’s hard to produce mechanically — e.g., CC-001’s “remediating any one finding still leaves the others viable,” and CC-003’s “the setup endpoint and the weaponization endpoint are different — each looked benign in isolation.”*

#### Sheets 32, 34, 39, 41, 42, 44, 45 – reconciliation & amendment logs

The audit-trail metadata of synthesis: which findings were merged, which rows were amended, and which reconciliations were applied at each pass. Sheet 34 (reconciliations, 5 × 5) and Sheet 45 (5 × 13) record where two specialist findings described the same underlying issue from different angles — the source findings, the merged result, and the reasoning for the merge.

#### Sheet 35 – endpoint\_rows (19 × 27)

Per-endpoint security rollup. 19 endpoints × 27 attributes — for each high-priority endpoint, every finding that affects it, the cumulative risk, and the prioritized remediation list. The 19 of 65 endpoints are the security-critical subset; the other 46 exist but don’t carry rollup-worthy issues.

Sheet 36 – open\_issues (3 × 4)

3 issues flagged for synthesis to resolve — places where specialist findings conflicted, evidence was ambiguous, or more reasoning was needed. *Epistemic honesty made mechanical*: the audit doesn't pretend everything reconciled cleanly — it records what didn't.

Sheet 37 – gcs/statsd findings (6 × 19)

A specialized sub-audit of Google Cloud Storage and statsd integration points.

Sheet 38 – hardening matrix (33 × 12)

33 hardening recommendations × 12 attributes — the *positive* recommendations (what should be in place that isn't), each paired with the audit findings that motivate it.

Sheet 40 – master findings (pre-final snapshot) (238 × 35)

The master findings table before the final synthesis pass — 238 findings × 35 attributes. The 238 count comes from combining each workstream's findings (48 + 39 + 25 + 20 + 27 + 19 + 23 + 20 across the specialist analyzers, plus the dependency, frontend, native-advisory and public-research items and the ad-hoc cluster), then subtracting reconciled duplicates and excluding intermediate findings.

Sheet 43 – master findings table (238 × 36) – the deliverable

The master findings table. 238 findings × 36 columns. The 36 columns:

|                        |   |
|------------------------|---|
| finding_id             | unique ID   |
| raw_finding_id         | the analyzer's original finding ID                          |
| workstream             | which analysis workstream produced it                       |
| file_path              | affected file   |
| line_numbers           | affected lines  |
| vuln_class             | vulnerability class (auth, injection, deserialization, ...) |
| cwe                    | CWE identifier  |
| title                  | short title   |
| short_narrative        | 1-paragraph narrative                                       |
| remediation_hint       | short remediation guidance                                  |
| status                 | patched / present / candidate / false-positive              |
| dim_characteristics    | characteristics dimension                                   |
| dim_origins            | origin dimension  |
| dim_exploitation       | exploitation dimension                                      |
| dim_outcome            | outcome dimension   |
| dim_difficulty         | exploitation difficulty                                     |
| dim_effort             | exploitation effort   |
| dim_action             | required attacker action                                    |
| action_axis_score      | 0-10  |
| effort_axis_score      | 0-10  |
| exposure_axis          | exposure score  |
| exploitability_axis    | exploitability score  |
| blast_radius_axis      | blast-radius score  |
| persistence_axis       | persistence score   |
| recoverability_axis    | recoverability score  |
| worst_case_tier        | worst-case threat tier                                      |
| related_finding_ids    | IDs of related findings                                     |
| related_chain_ids      | IDs of the CC-NNN chains this finding feeds into            |
| declared_tier          | the analyzer's claimed tier                                 |
| headline_tier          | the synthesizer's final tier                                |
| tier_divergence        | how much the analyzer and synthesizer disagreed             |
| fp_classification_by   | who marked false-positive (if applicable)                   |
| fp_classification_at   | when the false-positive call was made                       |
| fp_classification_note | false-positive rationale                                    |
| downgrade_rationale    | if the synthesizer downgraded from the analyzer's tier      |
| provenance             | links back to the source files and lines                    |

That schema is a security analyst's analytical framework instantiated systematically. The fact that `declared_tier`, `headline_tier`, and `tier_divergence` are tracked as three separate columns is *the audit being honest about its own reasoning disagreements* — when an analyzer said “T1” and synthesis said “T2,” that disagreement is recorded explicitly rather than silently overwritten.

#### Sheets 46–48 — self-trace & handoff

Sheet 46 (ocr\_ambiguities\_recorded, 3 × 6) records places where source material had ambiguous text — admitting where the input was unclear rather than silently resolving it. Sheets 47–48 (sheets) record the provenance of the workbook itself — which sources fed which sheets — and the handoff manifest. The audit's self-trace.



#### Sheet 49 – redactions (22 × 5)

22 redactions applied before final artifact creation — internal IP addresses, internal hostnames, names that appeared in commit logs, and other references that shouldn't appear in an external-facing audit. Each row records what was redacted, what it was replaced with, where it appeared, and the policy that triggered it.

#### Sheet 50 – checks (24 × 3)

24 verification checks run before final handoff — each confirming a structural or semantic property of the workbook (e.g., “all chain IDs in compounding\_chains reference real findings in the master table,” “all asset IDs in stride\_matrix reference real assets,” “no findings labeled `patched` claim present-day exploitability”).

#### Sheet 51 – artifacts (2 × 4)

The handoff manifest — identifying the output artifacts produced by the run.

## 7. How to read the workbook from a press / outreach perspective

---

If you have 30 seconds with a journalist, point at:

1. **Sheet 4** (endpoints, 65 rows) — “Umma mapped every API endpoint in Lemur.”
2. **Sheet 20** (stride\_matrix, 26 rows) — “Umma produced a full STRIDE threat model.”
3. **Sheet 33** (compounding\_chains, 6 rows) — “Umma identified 6 cross-file attack chains — including one where remediating any single finding leaves the others viable.”
4. **Sheet 43** (findings, 238 × 36) — “238 findings, 36 attributes each. The schema is a security researcher’s analytical framework.”

If you have 30 minutes:

1. Walk Sheet 8’s already-patched finding to show the honest-disclosure discipline (Umma labels patched issues as `info`, not `critical`).
2. Walk CC-001 in Sheet 33 to show the cross-file attack-chain reasoning.
3. Show the 36-column schema in Sheet 43 and call out `tier_divergence` — Umma records her own reasoning disagreements.
4. Show Sheet 36 (open issues) — Umma records the questions she couldn’t fully reconcile.

The pattern across all of these: **the workbook is honest about its own limits**. That is not a typical AI-output property, and it’s what makes it credible.

## 8. The fingerprint of recorded reasoning

---

The audit isn’t just “AI produced a spreadsheet.” Every sheet carries provenance back to the work that produced it, and every finding row in the master table links back to the specific files and lines it came from. The cross-references between sheets — asset IDs A1–A17 in sheet 17 referenced from sheet 20’s



threat matrix; finding IDs in sheet 33's chains referenced from the specialist sheets; chain IDs CC-001 through CC-006 referenced from sheet 43's `related_chain_ids` — are all *internally consistent*.

That internal consistency is what no current LLM-output can fake at scale. To produce a 1.29 MB workbook with this many cross-references that all resolve correctly, you need a system that knows what it has said and refers back to it.

That is the audit's deepest claim, and it survives any reviewer who tries to break it.