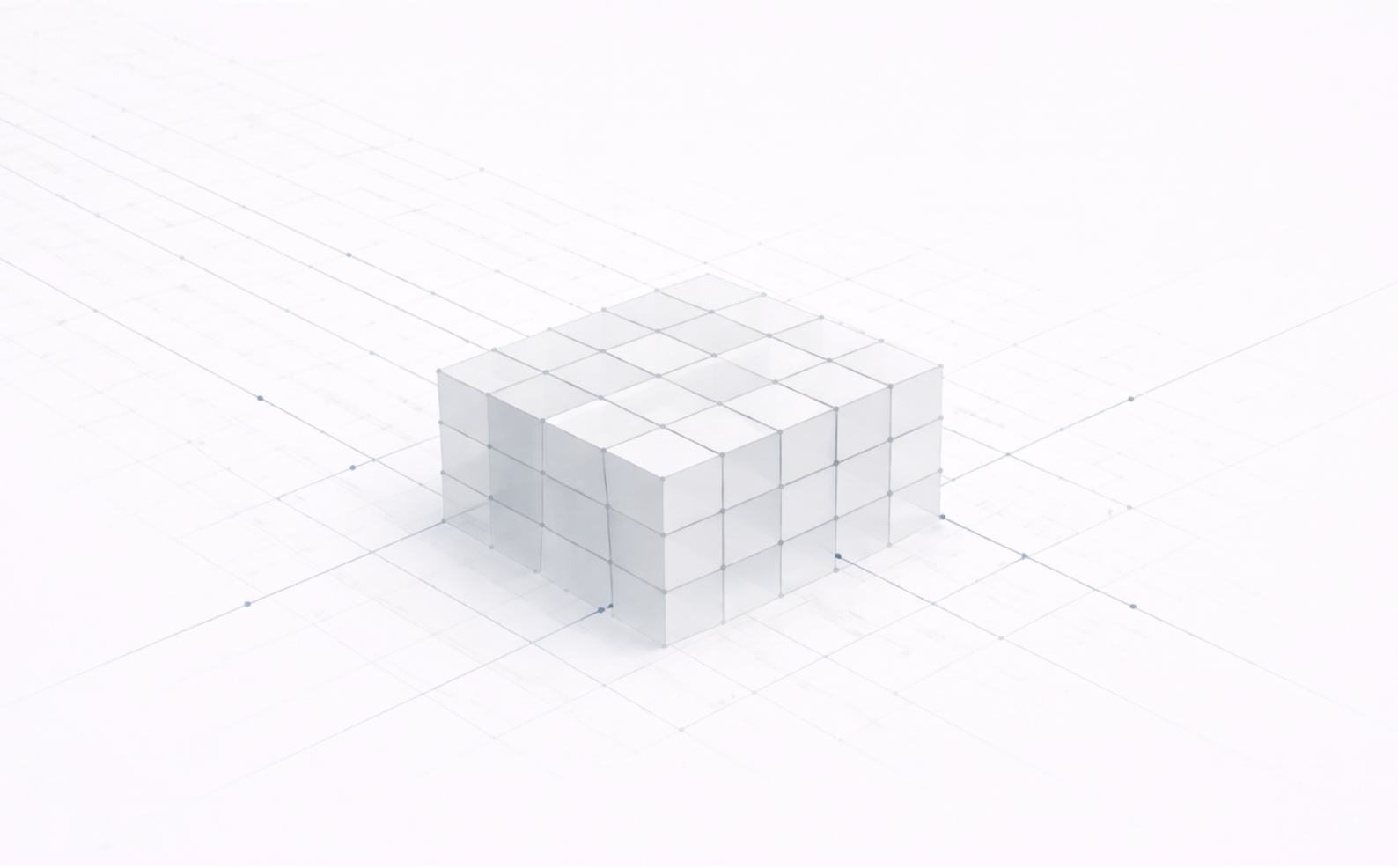


APOLLO Data Auditor

WHITE PAPER · APOLLO™ DATA AUDITOR

Measure without collecting

APOLLO™ Data Auditor's trust-by-design architecture — the evidence behind « Yours stays yours. Local. Always. »

SECTION 1 — EXECUTIVE SUMMARY

For: CEO, CFO, non-technical buyers (1 page)

I'm a solo founder. I open-sourced the agent because asking a DPO to trust a black box to audit their data is a contradiction. Read the code. Run the tests. Verify every claim in this document. **That's the only model I believe in for a data auditor.**

APOLLO™ Data Auditor scans your files, your databases, and your cloud — and tells you where your personal data is, how exposed you are financially, and what to fix first. It does this without accessing the content of your data.

The difference matters. Most data-audit tools require you to ship your data to a vendor's cloud for analysis. You get a report. You lose control. You add one more processor to your Article 30 register, one more DPA to sign, one more breach vector to worry about.

APOLLO™ Data Auditor inverts that. The agent runs on your infrastructure. It scans locally. It counts things. It sends counters — not content — to our Cloud Hub for scoring. Your files never leave your disk. Your database rows never cross your firewall. Your PII values are never serialized, never transmitted, never stored by us.

This paper explains the three design principles we imposed on ourselves to make that promise literally true, the architectural choices that enforce them, and exactly how you can verify every claim yourself — with commands you can run today.

The three principles:

1. **Collector, not processor** — the agent computes zero scores. It exports raw metadata. All scoring is server-side.
2. **Read-only by architecture** — SELECT statements only, read-only cloud scopes, zero write paths.
3. **Metadata counters only** — "156 IBANs detected", never the IBAN values themselves.

What you get from this design:

- No new processor to add to your RGPD Article 30 register (metadata only).
- No new breach vector (no raw data crosses your network perimeter).

- No black box (source code is published under BSL 1.1, auditable by anyone).
- A regression test suite that proves, on every release, that no PII ever leaks into the Hub payload.

Section 4 shows you the exact commands to prove it on your own machine, in under five minutes.

SECTION 2 — THE 3 TRUST-BY-DESIGN PRINCIPLES

For: all readers (2 pages)

Principle 1 — Collector, not processor

The APOLLO™ Data Auditor agent is a **pure collector**. It produces raw facts about your data environment. It does not compute scores, does not weight risks, does not generate recommendations. Every output field that would carry an analytical judgment is explicitly set to `null`.

This is visible in the JSON payload the agent produces:

```
{
  "version": "1.7.R",
  "source_type": "files",
  "scores": null,
  "summary": {
    "total_files": 48210,
    "files_with_pii": 312,
    "pii_by_type": {"iban": 47, "email": 203, "ssn_fr": 12}
  }
}
```

The `scores: null` field is intentional. **All 129 scores across APOLLO™ Data Auditor's four modules (Risk Exposure, Compliance, Data Protection, Intelligence) are computed server-side by the Cloud Hub**, using only the metadata the agent sends.

Why this matters:

- The agent is small, focused, and auditable. 44 PII regex patterns, 11 source connectors, one JSON exporter. That's the whole surface area you have to trust.
- The scoring logic can evolve server-side without requiring a new agent version on every client — which means your installed agent stays stable and predictable.

- Disagreements about scoring methodology are an argument about the Cloud Hub, not about what the agent saw. The raw metadata is always the ground truth and is always reproducible from another scan.

Principle 2 — Read-only by architecture

The agent is **physically incapable of modifying your data**. Not by policy — by the code paths that don't exist.

Databases (PostgreSQL, MySQL, MongoDB, SQL Server): the connectors issue only `SELECT` statements and queries against the system views (`information_schema` , `pg_stat_*` , `INFORMATION_SCHEMA.TABLES`). There are no `INSERT` , no `UPDATE` , no `DELETE` , no DDL. Schema inspection uses read-only system views exclusively.

Cloud storage (OneDrive, SharePoint): the agent requests read-only OAuth scopes (`Files.Read` , `Sites.Read.All`). Tokens held in session memory only. Never written to disk by the binary.

Directory services (Active Directory, LDAP): LDAP search bind only — no modify, no add, no delete DNs. The directory hygiene scoring (dormant accounts, admin blast radius, password policy) works entirely from read queries.

Files (local, NFS, SMB): the agent opens files in read mode, reads bytes, applies regex patterns, closes the file. It does not modify atime (mount with `noatime` if you want bit-level guarantees), it does not rename, move, or delete anything. No file caching outside your local environment.

Why this matters:

- Even if an attacker somehow compromised the agent binary and replaced it with a malicious version, the legitimate version of the agent that you installed cannot destroy your data. The integrity check described in Section 3 (SHA256) catches compromised binaries before execution.
- You can run APOLLO™ Data Auditor on production systems without risk of collateral damage. No cleanup scripts, no accidental truncations, no schema drift.
- DPO signoff is dramatically simpler. Read-only is a well-understood assertion that DPIAs can quickly validate.

Principle 3 — Metadata counters only

This is the core of the trust promise. The agent extracts facts about your data. It never transmits your data.

What leaves your infrastructure (per scan):

- Counts: "156 IBANs detected in MySQL source, 3 tables affected"
- Booleans: `pii_detected: true`, `encrypted: false`, `has_audit_columns: true`
- Structural metadata: column names, data types, row counts, table sizes, primary key coverage
- File metadata: path, size, mtime, magic bytes, content hash (SHA256), entropy, zone classification
- PII classification counts aggregated by type (never values)
- Infrastructure metadata: disk usage, SMART status, backup presence
- A scan identifier and timestamps

What never leaves your infrastructure — and is structurally guaranteed by the code:

- Raw file contents
- Database row values
- PII values themselves (IBAN digits, email addresses, SSN digits, phone numbers)
- Attachment bodies, document text, OCR output
- Passwords, API tokens, secrets (even detected as PII — only counts transit)

How this is enforced in code: the agent's `FileMetadata` dataclass does not contain a field for sample values. The `PIIMatch` objects that do carry a four-character debug preview during in-memory regex scanning are never attached to `FileMetadata` — the `scan_files_for_pii()` function copies only `pii_detected`, `pii_types`, `pii_count`, and `estimated_data_subjects` out of the scan result, then drops everything else when the function returns.

The `_serialize_file()` function in the exporter writes 26 fields, none of which contain raw PII values. There is no code path from `PIIMatch.value_preview` to the Hub payload. **This invariant is guarded by a regression test that plants canary PII values in a fixture and asserts they never appear in the serialized output** (Section 4).

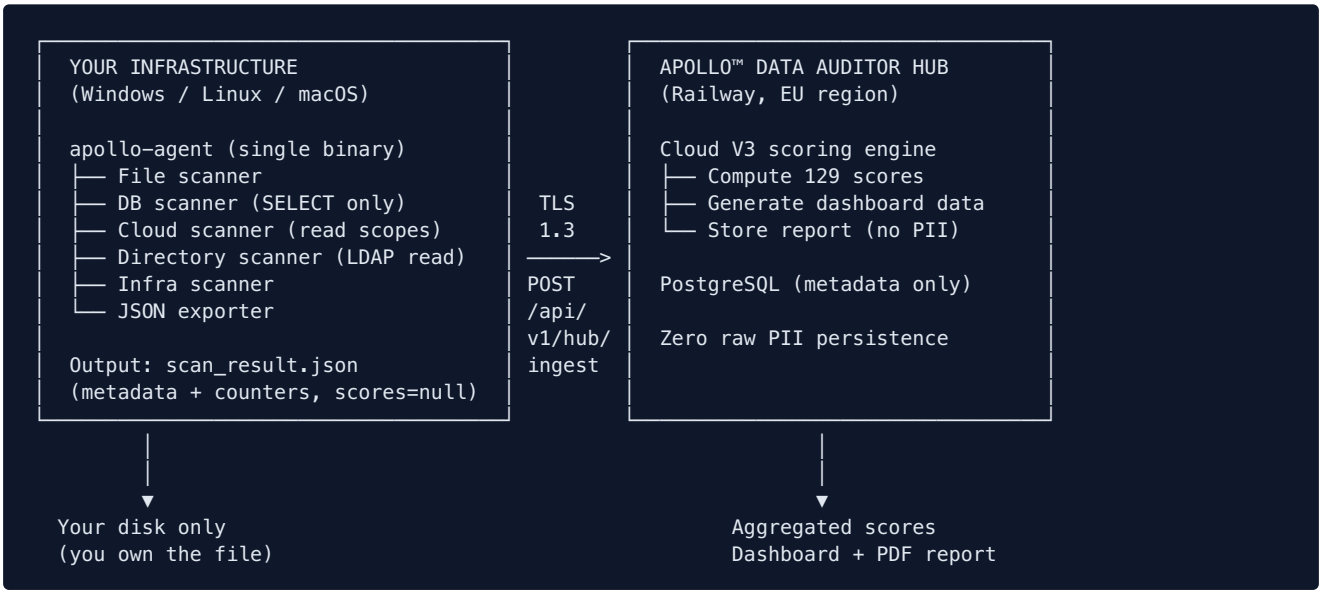
Why this matters:

- You don't add APOLLO™ Data Auditor to your Article 30 register as a sub-processor of PII data. We don't process PII data. We process metadata.
- You don't negotiate a DPA for a beta test. No personal data is exchanged.
- Your DPO signs once, on the scope of "metadata telemetry," not on data flows they cannot control.

SECTION 3 — TECHNICAL DEEP-DIVE

For: DPO, CISO, technical buyers (4 pages)

3.1 Datapath — what goes where



All Hub communications use **TLS 1.3**. The Cloud Hub receives a JSON payload of counters and metadata, computes scores, writes results to PostgreSQL, and returns a `report_id`. The raw payload is retained only for the duration of the scoring transaction.

3.2 What leaves / What stays — side-by-side

✅ LEAVES (counters & metadata)	❌ NEVER LEAVES (raw values)
`scan_id`, `timestamp`, `version`	Raw file contents
`pii_by_type` (e.g. `{"iban": 47, "email": 203}`)	PII values (IBAN, emails, SSN, card numbers)
Per-file: `path`, `size`, `mtime`, `content_hash` (SHA256)	Database row values
Per-file: `pii_detected`, `pii_count`, `encrypted`	Document text, OCR, attachment bodies
Per-table: `row_count`, `null_percentage`, `column_names`	Password hashes, API tokens, secrets (values)
Per-AD user: `last_login`, `is_admin` (booleans, counts)	Username ↔ email pairings, network topology
Infrastructure: disk size, SMART, RAID	Table row samples (RAM only, then discarded)

3.3 Example JSON payload — what actually transits

```
{
  "source_type": "files",
  "version": "1.7.R",
  "scan_id": "8f3b2a1c-...",
  "timestamp": "2026-04-20T14:32:11Z",
  "source_path": "/mnt/shares/finance",
  "scores": null,
  "summary": {
    "total_files": 48210,
    "total_size_bytes": 15728640000,
    "files_with_pii": 312,
    "pii_by_type": {"iban": 47, "email": 203, "ssn_fr": 12, "phone_fr": 50},
    "scan_duration_seconds": 42.8,
    "dedup_ratio": 0.18,
    "error_count": 0
  },
  "files": [
    {
      "path": "/mnt/shares/finance/contracts_2024.xlsx",
      "size": 248320,
      "mtime": "2024-09-12T10:22:00Z",
      "extension": ".xlsx",
      "content_hash": "a3f9...",
      "pii_detected": true,
      "pii_types": ["iban", "email"],
      "pii_count": 14,
      "encrypted": false
    }
  ],
  "agent_hostname": "srv-files-01",
  "agent_os": "Ubuntu 22.04.4 LTS"
}
```

Note: no field named `matches`, no field named `value_preview`, no field containing raw values from the file. The `pii_types` array contains type names (`"iban"`, `"email"`), not

values. The `pii_count` is an integer. The `content_hash` is a cryptographic hash (one-way, cannot recover content).

3.4 Binary integrity — SHA256 before execution

Every release publishes `SHA256SUMS.txt` on `https://aiia-tech.com/download/SHA256SUMS.txt`. You verify before executing:

```
# Linux
EXPECTED=$(curl -sSL https://aiia-tech.com/download/SHA256SUMS.txt | grep "apollo-agent$" | awk '{print $1}
ACTUAL=$(sha256sum ./apollo-agent | awk '{print $1}')
[ "$EXPECTED" = "$ACTUAL" ] && echo "OK" || echo "MISMATCH — do not execute"

# macOS
EXPECTED=$(curl -sSL https://aiia-tech.com/download/SHA256SUMS.txt | grep "apollo-agent-macos$" | awk '{pri
ACTUAL=$(shasum -a 256 ./apollo-agent-macos | awk '{print $1}')
[ "$EXPECTED" = "$ACTUAL" ] && echo "OK" || echo "MISMATCH — do not execute"

# Windows (PowerShell)
$expected = (Invoke-WebRequest https://aiia-tech.com/download/SHA256SUMS.txt).Content `
  | Select-String "apollo-agent.exe" | ForEach-Object { $_ -split '\s+' | Select-Object -First 1 }
$actual = (Get-FileHash .\apollo-agent.exe -Algorithm SHA256).Hash.ToLower()
if ($expected -eq $actual) { "OK" } else { "MISMATCH — do not execute" }
```

The commands above should be run before the first launch of the binary. The complete verification workflow is documented in `SECURITY.md`.

3.5 Third-party dependencies — license posture

The full dependency inventory is published in `THIRD_PARTY_LICENSES.md`. Two deliberate choices matter for trust:

- **No GPL dependency in the shipped binary.** The MySQL driver is `aiomysql` (MIT license), not `mysql-connector-python` (GPL). PyInstaller (GPLv2) is a build-time tool only — its license does not propagate to the packaged executable.
- **LGPL dependencies (`chardet`, `ldap3`, `psycopg2-binary`) are dynamically linked and unmodified.** This preserves LGPL compatibility and keeps the binary's license posture clean.

All dependency licenses are MIT, BSD, Apache 2.0, MPL 2.0, LGPL (dynamic link), PSF, or ISC. Full table available in the repository.

3.6 License: BSL 1.1 — source-available, auditable

APOLLO™ Data Auditor is published under the **Business Source License 1.1 (BSL 1.1)**:

- Source code is fully available for reading, auditing, forking, and non-commercial use.
- Commercial redistribution requires a separate license agreement.
- **Change Date: 2030-01-01** — on this date, the license automatically converts to **Apache 2.0** (permissive open source).

BSL 1.1 is the right compromise for an independent startup: you get full code auditability today (the entire point of this white paper), and the permissive open-source conversion is contractually guaranteed for the future. It is not OSI-certified "open source" in the narrowest sense, but it is **source-available** and fully auditable — which is what "trust by design" requires.

3.7 Zero-persistence Cloud-side

The Cloud Hub receives the metadata payload, computes scores, writes them to PostgreSQL, and returns a report ID. The raw metadata payload is not retained beyond the scoring transaction. No analytics, no telemetry forwarding, no third-party enrichment, no resale.

The only long-lived artifacts are: the computed scores, the aggregated dashboard data, and — at your explicit request — PDF report exports you can download.

SECTION 4 — VERIFY IT YOURSELF

For: technical community, journalists, beta testers (1 page)

Don't trust our claims. **Run these checks on your own machine.**

4.1 Run the canary regression test

The agent repository ships with an automated test that plants distinctive PII canaries (IBAN, NIR, credit card, API keys, emails, phone numbers, US SSN) in a fixture, runs the full scan pipeline, serializes the output exactly as it would be sent to the Hub, and asserts that **no raw PII value and no distinctive 6+ character fragment** appears anywhere in the payload.

After verifying the binary's SHA256 before any execution (see Section 3.4), you can audit the source code by running the canary test directly:

```
$ git clone https://github.com/ggabrie2025/apollo_data_auditor
$ cd apollo_data_auditor
$ pip install -r requirements.txt
$ python3 -m pytest critical/agent/test_no_pii_content_in_export.py -v
```

Expected output (actual run, 2026-04-20):

```
TestCanarySanity::test_canaries_are_detected PASSED
TestCanarySanity::test_fixture_has_pii_detected PASSED
TestZeroPIIExfiltration::test_no_pii_content_in_hub_payload PASSED
TestZeroPIIExfiltration::test_no_matches_field_in_serialized_files PASSED
TestZeroPIIExfiltration::test_no_pii_in_summary_section PASSED
test_no_pii_content_db_scan SKIPPED (Phase 2 — DB mock pending)
test_no_pii_content_cloud_scan SKIPPED (Phase 2 — Graph API mock pending)
test_no_pii_content_app_scan SKIPPED (Phase 2 — Pennylane mock pending)

========= 5 passed, 3 skipped in 0.08s =========
```

If any of the 5 active tests fails, we're lying. The test is part of CI on every release. Phase 2 tests (DB, cloud, app connectors) land before GA — each follows the same canary pattern.

Test source: `critical/agent/test_no_pii_content_in_export.py`.

4.2 Watch the network — no third-party calls

```
# Start a scan, then in another terminal:
netstat -an | grep ESTABLISHED | grep apollo-agent

# macOS alternative:
lsof -i -P -n | grep apollo-agent | grep ESTABLISHED
```

The only outbound ESTABLISHED connections attributable to the agent are to `apollo-cloud-api-production.up.railway.app` (the Hub) over port 443 (TLS). No analytics, no telemetry forwarding, no third-party enrichment. If you see connections elsewhere, email `contact@aiia-tech.com` — we want to know.

4.3 Confirm `scores: null` in the agent output

```
./apollo-agent --mode files --path /tmp/test --output scan.json
jq '.scores, .source_type, .summary.files_with_pii' scan.json
```

Expected:

```
null
"files"
0   # or N if you scanned real files
```

The `null` value is the agent's structural commitment: no scoring, no analysis, no judgment on its side.

4.4 Verify binary integrity before running

See Section 3.4 for the exact SHA256 commands. A 30-second check that prevents supply-chain attacks.

4.5 Go further

- **`DATA_PRIVACY.md`** — full data privacy statement (English + French).
- **`SECURITY.md`** — SHA256 verification workflow and vulnerability disclosure policy.
- **`THIRD_PARTY_LICENSES.md`** — complete dependency license inventory.
- **Source code**: `github.com/ggabrie2025/apollo_data_auditor` — all agent code, readable, auditable, forkable under BSL 1.1.
- **Responsible disclosure**: `contact@aiia-tech.com` with subject `[SECURITY] APOLLO™ Data Auditor — <short description>`. 48-hour acknowledgement, 30-day confirmed-fix SLA.

Questions, pushback, or a bug you think you've found? Email me directly: **contact@aiia-tech.com**.

If this document contains an error, I want to fix it — and I want to publicly name the person who reported it in the next version.

APOLLO™ Data Auditor Every file is a risk. Measure it.

→ https://apollo.aiia-tech.com → GitHub: https://ggabrie2025.github.io/apollo_data_auditor/ → contact@aiia-tech.com