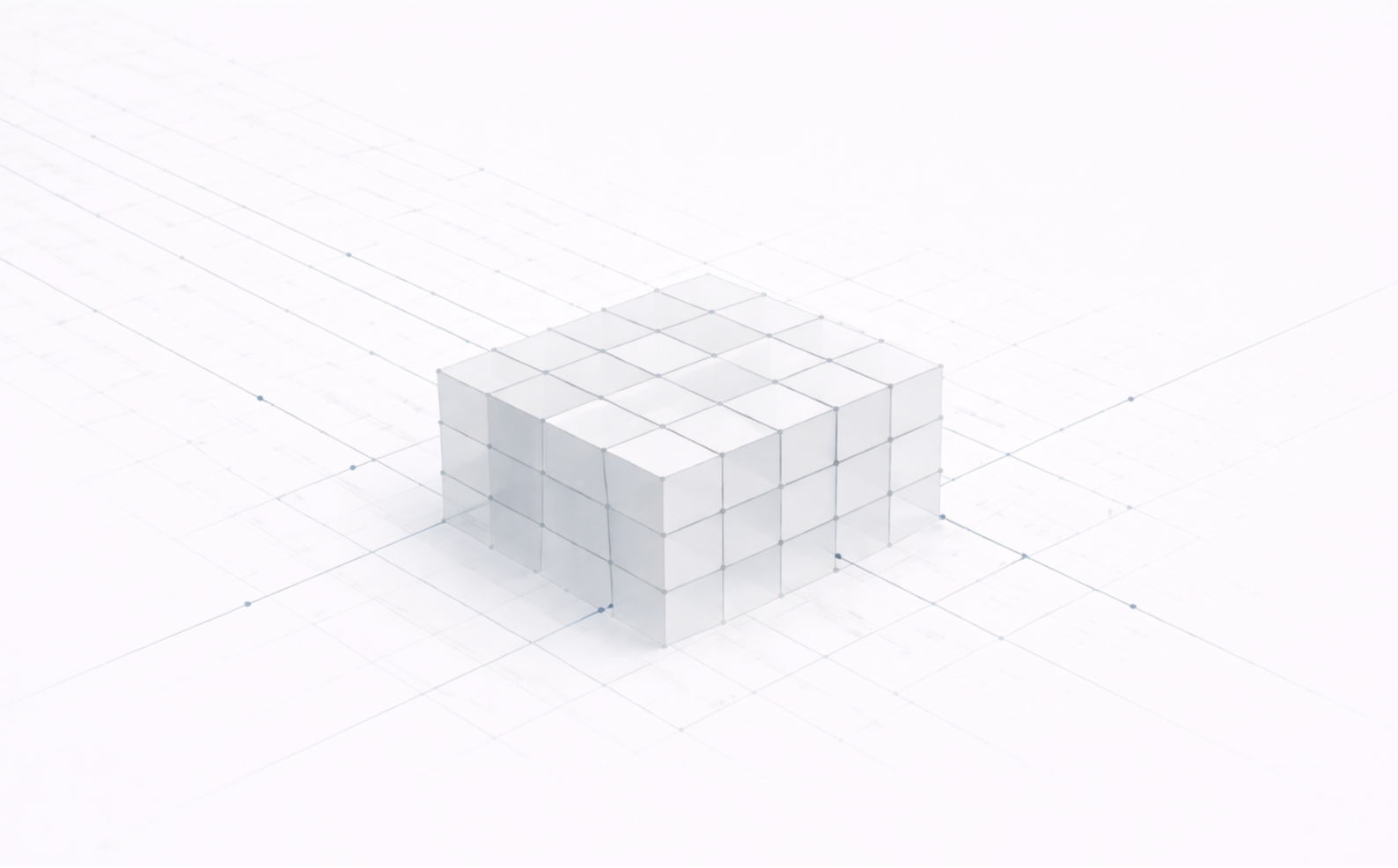


APOLLO Data Auditor

LIVRE BLANC · APOLLO™ DATA AUDITOR

Mesurer sans collecter

L'architecture trust-by-design d'APOLLO™ Data Auditor — les preuves derrière « Yours stays yours. Local. Always. »

SECTION 1 — RÉSUMÉ EXÉCUTIF

Pour : CEO, CFO, acheteurs non techniques (1 page)

Je suis un solo founder. J'ai publié le code source de l'agent parce que demander à un DPO de faire confiance à une boîte noire pour auditer ses données est une contradiction. Lisez le code. Lancez les tests. Vérifiez chaque affirmation de ce document. **C'est le seul modèle auquel je crois pour un auditeur de données.**

APOLLO™ Data Auditor scanne vos fichiers, vos bases de données et votre cloud — et vous dit où se trouvent vos données personnelles, quelle est votre exposition financière, et ce qu'il faut corriger en priorité. Il fait tout cela sans accéder au contenu de vos données.

La différence est fondamentale. La plupart des outils d'audit de données exigent que vous envoyiez vos données sur le cloud d'un éditeur pour analyse. Vous recevez un rapport. Vous perdez le contrôle. Vous ajoutez un sous-traitant à votre registre Article 30, un DPA à signer, un vecteur de fuite supplémentaire à gérer.

APOLLO™ Data Auditor inverse cette logique. L'agent tourne sur votre infrastructure. Il scanne localement. Il compte. Il envoie des compteurs — pas du contenu — à notre Hub Cloud pour le scoring. Vos fichiers ne quittent jamais votre disque. Les lignes de vos bases de données ne traversent jamais votre pare-feu. Les valeurs de vos PII ne sont jamais sérialisées, jamais transmises, jamais stockées chez nous.

Ce document explique les trois principes de conception que nous nous sommes imposés pour que cette promesse soit littéralement vraie, les choix d'architecture qui les enforcent, et comment vous pouvez vérifier chaque affirmation par vous-même — avec des commandes que vous pouvez exécuter aujourd'hui.

Les trois principes :

1. **Collecteur, pas processeur** — l'agent calcule zéro score. Il exporte des métadonnées brutes. Tout le scoring est server-side.
2. **Lecture seule par architecture** — uniquement des SELECT, scopes cloud en lecture, zéro chemin d'écriture.

3. **Compteurs de métadonnées uniquement** — « 156 IBAN détectés », jamais les valeurs des IBAN.

Ce que ce design vous apporte :

- Pas de nouveau sous-traitant à ajouter à votre registre RGPD Article 30 (métadonnées uniquement).
- Pas de nouveau vecteur de fuite (aucune donnée brute ne traverse le périmètre réseau).
- Pas de boîte noire (code source publié sous BSL 1.1, auditable par tous).
- Une suite de tests de régression qui prouve, à chaque release, qu'aucune PII ne fuite dans le payload vers le Hub.

La Section 4 vous donne les commandes exactes pour le prouver sur votre machine, en moins de cinq minutes.

SECTION 2 — LES 3 PRINCIPES TRUST-BY-DESIGN

Pour : tous les lecteurs (2 pages)

Principe 1 — Collecteur, pas processeur

L'agent APOLLO™ Data Auditor est un **pur collecteur**. Il produit des faits bruts sur votre environnement de données. Il ne calcule pas de scores, ne pondère pas de risques, ne génère pas de recommandations. Chaque champ de sortie qui porterait un jugement analytique est explicitement positionné à `null`.

Visible dans le JSON produit par l'agent :

```
{
  "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}
  }
}
```

Le champ `scores: null` est intentionnel. **Les 129 scores sur les quatre modules d'APOLLO™ Data Auditor (Risk Exposure, Compliance, Data Protection, Intelligence)**

sont tous calculés côté serveur par le Hub Cloud, uniquement à partir des métadonnées envoyées par l'agent.

Pourquoi c'est important :

- L'agent est petit, ciblé, auditable. 44 patterns regex PII, 11 connecteurs, un exporter JSON. C'est toute la surface à auditer.
- La logique de scoring peut évoluer côté serveur sans imposer une nouvelle version de l'agent à chaque client — votre agent installé reste stable et prédictible.
- Les débats sur la méthodologie de scoring portent sur le Hub Cloud, pas sur ce que l'agent a observé. Les métadonnées brutes restent la vérité terrain, toujours reproductible via un nouveau scan.

Principe 2 — Lecture seule par architecture

L'agent est **physiquement incapable de modifier vos données**. Pas par politique — par les chemins de code qui n'existent pas.

Bases de données (PostgreSQL, MySQL, MongoDB, SQL Server) : les connecteurs émettent uniquement des requêtes `SELECT` et des lectures sur les vues système (`information_schema`, `pg_stat_*`, `INFORMATION_SCHEMA.TABLES`). Aucun `INSERT`, aucun `UPDATE`, aucun `DELETE`, aucun DDL. L'inspection de schéma passe exclusivement par les vues système en lecture.

Stockage cloud (OneDrive, SharePoint) : l'agent demande des scopes OAuth en lecture seule (`Files.Read`, `Sites.Read.All`). Les tokens restent en mémoire de session. Jamais écrits sur disque par le binaire.

Annuaires (Active Directory, LDAP) : search bind LDAP uniquement — pas de modify, pas de add, pas de delete de DN. Le scoring hygiène annuaire (comptes dormants, blast radius admin, politique mot de passe) fonctionne intégralement en lecture.

Fichiers (local, NFS, SMB) : l'agent ouvre les fichiers en lecture, lit les octets, applique les patterns regex, ferme le fichier. Il ne modifie pas atime (montez en `noatime` si vous voulez une garantie au bit près), ne renomme pas, ne déplace pas, ne supprime rien. Aucun caching de contenu hors de votre environnement.

Pourquoi c'est important :

- Même si un attaquant compromettait le binaire pour le remplacer par une version malveillante, la version légitime que vous avez installée ne peut pas détruire vos données. Le contrôle d'intégrité SHA256 décrit en Section 3 détecte les binaires compromis avant exécution.
- Vous pouvez lancer APOLLO™ Data Auditor sur des systèmes de production sans risque de dommages collatéraux. Pas de script de cleanup, pas de troncature accidentelle, pas de dérive de schéma.
- La validation DPO est nettement plus rapide. « Lecture seule » est une affirmation bien comprise que les AIPD valident sans friction.

Principe 3 — Compteurs de métadonnées uniquement

C'est le cœur de la promesse de confiance. L'agent extrait des faits sur vos données. Il ne transmet jamais vos données.

Ce qui quitte votre infrastructure (par scan) :

- Compteurs : « 156 IBAN détectés dans la source MySQL, 3 tables concernées »
- Booléens : `pii_detected: true` , `encrypted: false` , `has_audit_columns: true`
- Métadonnées structurelles : noms de colonnes, types, row counts, tailles de table, couverture clés primaires
- Métadonnées fichiers : path, size, mtime, magic bytes, content hash (SHA256), entropie, zone
- Comptes de classification PII agrégés par type (jamais les valeurs)
- Métadonnées infrastructure : disque, SMART, présence backup
- Un identifiant de scan et des timestamps

Ce qui ne quitte jamais votre infrastructure — garanti structurellement par le code :

- Contenu brut des fichiers
- Lignes de base de données
- Valeurs de PII (chiffres IBAN, adresses email, chiffres SSN, numéros de téléphone)
- Corps de pièces jointes, texte de documents, sorties OCR

- Mots de passe, tokens API, secrets (même détectés comme PII — seul le compteur transite)

Comment c'est enforcé dans le code : la dataclass `FileMetadata` de l'agent ne contient aucun champ pour des échantillons de valeurs. Les objets `PIIMatch` qui portent en interne un preview 4 caractères pendant le scan regex en mémoire ne sont jamais attachés à `FileMetadata` — la fonction `scan_files_for_pii()` ne copie que `pii_detected`, `pii_types`, `pii_count` et `estimated_data_subjects` depuis le résultat du scan, puis tout le reste est abandonné au retour de la fonction.

La fonction `_serialize_file()` dans l'exporter écrit 26 champs, dont aucun ne contient de valeurs PII brutes. **Il n'existe aucun chemin de code entre `PIIMatch.value_preview` et le payload envoyé au Hub.** Cette invariance est gardée par un test de régression qui plante des canaris PII dans une fixture et affirme qu'ils n'apparaissent jamais dans la sortie sérialisée (Section 4).

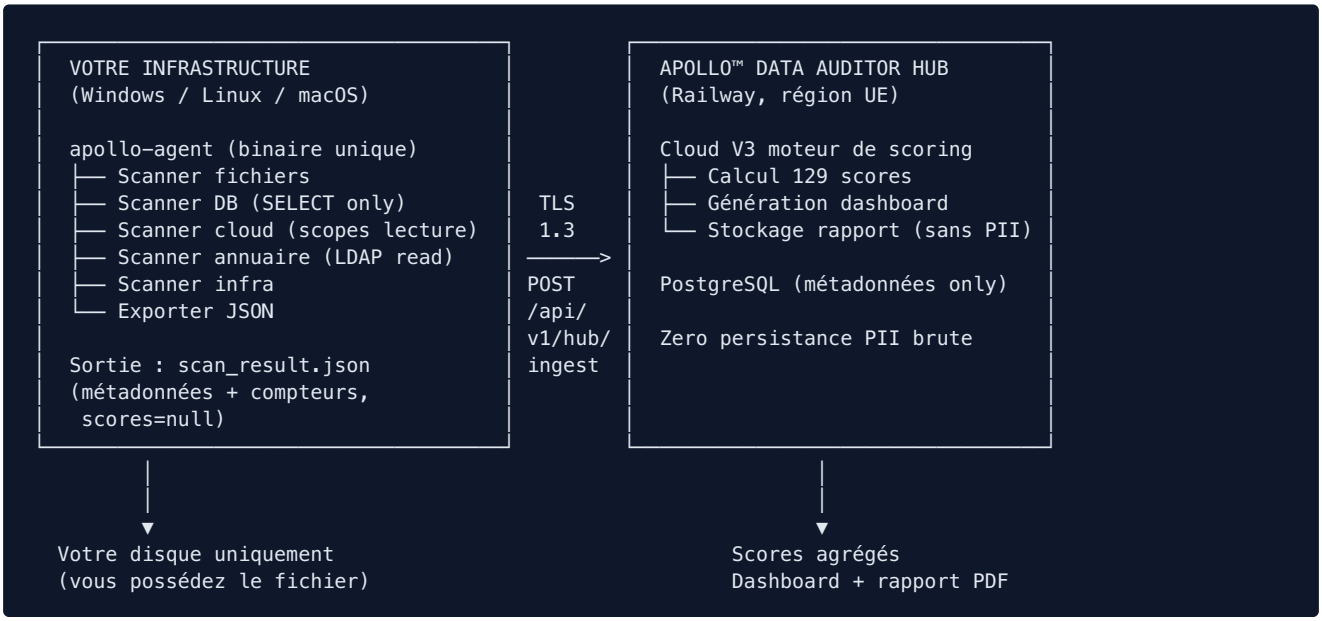
Pourquoi c'est important :

- Vous n'ajoutez pas APOLLO™ Data Auditor à votre registre Article 30 comme sous-traitant de données PII. Nous ne traitons pas de PII. Nous traitons des métadonnées.
- Vous ne négociez pas de DPA pour un test beta. Aucune donnée personnelle n'est échangée.
- Votre DPO signe une fois, sur le périmètre « télémétrie métadonnées », pas sur des flux de données qu'il ne peut contrôler.

SECTION 3 — DEEP-DIVE TECHNIQUE

Pour : DPO, CISO, acheteurs techniques (4 pages)

3.1 Datapath — qui va où



Toutes les communications vers le Hub utilisent **TLS 1.3**. Le Hub reçoit un payload JSON de compteurs et métadonnées, calcule les scores, écrit les résultats en PostgreSQL, retourne un `report_id`. Le payload brut n'est conservé que pour la durée de la transaction de scoring.

3.2 What leaves / What stays — lecture en un coup d'œil

✅ QUITTE (compteurs & métadonnées)	❌ NE QUITTE JAMAIS (valeurs brutes)
`scan_id`, `timestamp`, `version`	Contenu brut des fichiers
`pii_by_type` (ex. `{"iban": 47, "email": 203}`)	Valeurs de PII (IBAN, emails, NIR, cartes)
Par fichier : `path`, `size`, `mtime`, `content_hash` (SHA256)	Lignes de bases de données
Par fichier : `pii_detected`, `pii_count`, `encrypted`	Texte de documents, OCR, pièces jointes
Par table : `row_count`, `null_percentage`, `column_names`	Hashes de mots de passe, tokens, secrets (valeurs)
Par user AD : `last_login`, `is_admin` (booléens, compteurs)	Associations nom ↔ email, topologie réseau
Infrastructure : taille disque, SMART, RAID	Échantillons de lignes (RAM uniquement, puis jetés)

3.3 Exemple de payload JSON — ce qui transite réellement


```
{
  "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"
}
```


Observation : aucun champ `matches`, aucun champ `value_preview`, aucun champ contenant des valeurs brutes issues du fichier. Le tableau `pii_types` contient des noms de types (`"iban"`, `"email"`), pas des valeurs. `pii_count` est un entier. `content_hash` est un hash cryptographique (unidirectionnel, non reversible).

3.4 Intégrité binaire — SHA256 avant exécution

Chaque release publie `SHA256SUMS.txt` sur `https://aiia-tech.com/download/SHA256SUMS.txt`. Vous vérifiez avant d'exécuter :

```
# 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 — ne pas exécuter"

# 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 — ne pas exécuter"

# 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 — ne pas exécuter" }
```

Les commandes ci-dessus sont à exécuter avant tout premier lancement du binaire. La procédure complète de vérification et le workflow recommandé sont documentés dans `SECURITY.md`.

3.5 Dépendances tierces — posture de licences

L'inventaire complet est publié dans `THIRD_PARTY_LICENSES.md`. Deux choix délibérés sont importants pour la confiance :

- **Aucune dépendance GPL dans le binaire livré.** Le driver MySQL utilisé est `aiomysql` (licence MIT), pas `mysql-connector-python` (GPL). PyInstaller (GPLv2) est un outil de build uniquement — sa licence ne se propage pas au binaire packagé.
- **Les dépendances LGPL (`chardet` , `ldap3` , `psycopg2-binary`) sont liées dynamiquement et non modifiées.** Cela préserve la compatibilité LGPL et garde une posture de licence propre pour le binaire.

Toutes les licences des dépendances sont MIT, BSD, Apache 2.0, MPL 2.0, LGPL (liaison dynamique), PSF ou ISC. Tableau complet disponible dans le dépôt.

3.6 Licence : BSL 1.1 — source-available, auditable

APOLLO™ Data Auditor est publié sous **Business Source License 1.1 (BSL 1.1)** :

- Le code source est entièrement disponible pour lecture, audit, fork et usage non-commercial.
- La redistribution commerciale nécessite un accord de licence séparé.
- **Change Date : 2030-01-01** — à cette date, la licence se convertit automatiquement en **Apache 2.0** (open source permissive).

BSL 1.1 est le bon compromis pour une startup indépendante : vous obtenez l'auditabilité complète du code aujourd'hui (c'est précisément l'objet de ce document), et la conversion vers une licence open-source permissive est contractuellement garantie à terme. Ce n'est pas une licence OSI « open source » au sens strict, mais c'est **source-available** et pleinement auditable — ce qui est exactement ce que « trust by design » exige.

3.7 Zero-persistance côté Cloud

Le Hub reçoit le payload de métadonnées, calcule les scores, les écrit en PostgreSQL, retourne un identifiant de rapport. Le payload brut de métadonnées n'est pas conservé au-delà de la transaction de scoring. Pas d'analytics, pas de forwarding télémétrie, pas d'enrichissement tiers, pas de revente.

Les seuls artefacts durables sont : les scores calculés, les données agrégées du dashboard, et — à votre demande explicite — les exports PDF de rapport que vous téléchargez.

SECTION 4 — VÉRIFIEZ VOUS-MÊME

Pour : communauté technique, journalistes, beta testeurs (1 page)

Ne nous croyez pas sur parole. **Lancez ces vérifications sur votre propre machine.**

4.1 Lancer le test de régression canary

Le dépôt de l'agent embarque un test automatisé qui plante des canaris PII distinctifs (IBAN, NIR, carte bancaire, clés API, emails, téléphones, SSN US) dans une fixture, déroule le pipeline de scan complet, sérialise la sortie exactement comme elle serait envoyée au Hub, et affirme qu'**aucune valeur PII brute ni fragment distinctif de 6+ caractères** n'apparaît dans le payload.

Après avoir vérifié le SHA256 du binaire avant toute exécution (voir Section 3.4), vous pouvez auditer le code source en lançant directement le test canary :

```
$ 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
```

Sortie attendue (run réel, 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 — mock DB en attente)
test_no_pii_content_cloud_scan SKIPPED (Phase 2 — mock Graph API en attente)
test_no_pii_content_app_scan SKIPPED (Phase 2 — mock Pennylane en attente)

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

Si l'un des 5 tests actifs échoue, nous mentons. Le test tourne en CI à chaque release. Les tests Phase 2 (DB, cloud, connecteur app) arrivent avant la GA — chacun suit le même pattern canary.

Source : `critical/agent/test_no_pii_content_in_export.py`.

4.2 Surveiller le réseau — aucun appel tiers

```
# Lancez un scan, puis dans un autre terminal :
netstat -an | grep ESTABLISHED | grep apollo-agent

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

Les seules connexions sortantes ESTABLISHED attribuables à l'agent pointent vers `apollo-cloud-api-production.up.railway.app` (le Hub) sur le port 443 (TLS). Pas d'analytics, pas de télémétrie, pas d'enrichissement tiers. Si vous observez autre chose, écrivez à `contact@aiia-tech.com` — nous voulons savoir.

4.3 Confirmer `scores: null` dans la sortie de l'agent

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

Attendu :

```
null
"files"
0   # ou N si vous avez scanné des fichiers réels
```

La valeur `null` est l'engagement structurel de l'agent : aucun scoring, aucune analyse, aucun jugement côté agent.

4.4 Vérifier l'intégrité binaire avant exécution

Voir Section 3.4 pour les commandes SHA256 exactes. Une vérification de 30 secondes qui prévient les attaques supply-chain.

4.5 Pour aller plus loin

- `DATA_PRIVACY.md` — engagement confidentialité complet (FR + EN).
- `SECURITY.md` — workflow de vérification SHA256 et politique de divulgation.
- `THIRD_PARTY_LICENSES.md` — inventaire complet des licences de dépendances.
- **Code source** : `github.com/ggabrie2025/apollo_data_auditor` — tout le code agent, lisible, auditable, forkable sous BSL 1.1.
- **Divulgation responsable** : `contact@aiia-tech.com` avec pour objet `[SECURITY] APOLLO™ Data Auditor — <description courte>`. Accusé de réception sous 48h, correctif confirmé sous 30 jours.

Questions, objections, ou un bug que vous pensez avoir trouvé ? Écrivez-moi directement : **contact@aiia-tech.com**.

Si ce document contient une erreur, je veux la corriger — et je veux nommer publiquement la personne qui l'a signalée dans la version suivante.

APOLLO™ Data Auditor Tout fichier est un risque. Mesurez-le.

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