LONIA AI / PALLAS

Data Protection Impact Assessment

GDPR Article 35 assessment of processing in Pallas by Lonia AI

A systematic description of the processing Pallas carries out, an assessment of its necessity and proportionality, an assessment of the risks to the rights and freedoms of data subjects with likelihood and severity stated, and the measures relied on to address those risks, each rendered from the Pallas control inventory with its status and evidence basis unaltered.

Version. 1.0. Published 2026-08-30 by Lonia AI.

Jurisdictions addressed. EU GDPR Article 35 primarily. The UK ICO screening criteria and the Quebec Law 25 privacy impact assessment duty are addressed where they diverge. Not a substitute for a controller's own assessment.

Every control claim in this document is rendered from the Pallas control inventory at src/data/control-inventory.ts, with its status, its evidence basis, and its disclosure text unaltered. Appendix A lists every control cited.

This assessment is published so that a controller carrying out its own Article 35 assessment over a Pallas deployment can start from a described system. It is Lonia AI's assessment of Lonia AI's processing, not a substitute for the controller's own.

This document is provided for procurement and vendor-onboarding review. It is not legal advice, and both parties should have any contractual artifact reviewed by counsel before execution. Questions: support@lonia.ai

Contents

1. Purpose, scope, and whose assessment this is

Article 35 places the obligation to carry out a data protection impact assessment on the controller. Pallas is a processor for customer organisation data and a controller for a narrow set of its own. This document is therefore two things at once, and it says which it is being in each section, because an assessment that blurs the two is useless to the party who has to file one.

- Where Lonia AI is the **processor**, this document is Article 28(3)(f) assistance made concrete. A controller carrying out its own assessment over a Pallas deployment can start from a described system rather than from a questionnaire. It does not discharge the controller's duty.
- Where Lonia AI is the **controller**, which is user profile data and marketing list membership, this is Lonia AI's own assessment.

The controller split is not only a policy statement. It is expressed in the schema: the user export and the organisation export are separate database functions rather than one filtered by the other.

Pallas distinguishes the two controller roles in its schema, not only in its policy text. Lonia AI is controller for user profile data; the customer organisation is controller for its own scan and compliance data. The user export and the organisation export are separate functions, not one filtered by the other.

> Control gdpr-controller-split-documented. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.
>
> Implementation. app: pallas_export_user_data 039:268 for the individual, pallas_export_org_data 009:13 rebuilt in 032:67 and 037:297 for the organisation.
>
> Verification. app: supabase/migrations/039_gdpr_user_rights.sql, controller split section.

Scope. All processing carried out by the Service, being the marketing site at pallas.lonia.ai, the authenticated application at app.pallas.lonia.ai, the Supabase database and Storage that back them, and the Cloudflare Workers that sit between. Out of scope: processing a customer carries out in its own systems, and processing by an OAuth identity provider before an authentication assertion reaches Pallas.

Version basis. This assessment reflects the 30 August 2026 control inventory pass. Every mitigation below is rendered from that inventory, so a control that was partial on that date says so here.

2. Is an assessment required

Article 35(1) requires an assessment where processing is likely to result in a high risk to the rights and freedoms of natural persons. Article 35(3) lists three cases where one is always required. Pallas meets none of the three squarely, and the assessment is carried out anyway. The reasoning is set out rather than the conclusion alone.

- **Article 35(3)(a), systematic and extensive evaluation based on automated processing, including profiling, producing legal or similarly significant effects.** Not met. Pallas evaluates web content against success criteria, not people. No decision about a person is produced by the Service, automated or otherwise. Pallas performs no profiling and holds no behavioural analytics.

- **Article 35(3)(b), processing on a large scale of special categories of data or data relating to criminal convictions.** Not met by design, with a qualification the next section carries: Pallas does not solicit special category data, but it scans customer-supplied URLs and accepts customer-uploaded documents, so content outside Pallas control transits the system.
- **Article 35(3)(c), systematic monitoring of a publicly accessible area on a large scale.** Not met. Pallas monitors web properties a customer designates as its own, not public areas, and not people in them.

Against the European Data Protection Board's nine criteria, the processing touches two: evaluation or scoring, in the weak sense that a page receives a conformance verdict, and data processed on a large scale, in the sense that the audit ledger accumulates. Two criteria is below the threshold at which the Board's guidance indicates an assessment is likely to be required.

The assessment is carried out regardless, for three reasons. The audit ledger is designed to be permanent and tamper-evident, which is a deliberate constraint on erasure and deserves to be assessed as one. Uploaded source files are the data class most likely to contain something unintended. And a procurement reviewer for a public sector or education buyer will ask for this document whether or not Article 35 compels it.

3. Systematic description of the processing

3.1 The four data categories assessed

This assessment is organised around four categories rather than around database tables, because they have four different risk profiles and four different lawful bases.

1. **Audit chain data.** Records of who did what, when, in which organisation. Tamper-evident by design, retained for a seven-year minimum floor, and deliberately resistant to silent removal. Controller: the customer organisation.
2. **Scan result data.** Assets, scans, findings, evidence, comments, reports, and transiently uploaded source files. Retained under per-organisation policy. Controller: the customer organisation.
3. **User profile data.** Name, avatar, and the email address held in the authentication record. Controller: Lonia AI.
4. **Organisation data.** Organisation name, membership, roles, invitations, subscription and billing metadata. Controller: the customer organisation for its own content; Lonia AI for the account relationship.

Alongside these, and outside the Service proper, marketing list membership is processed with Lonia AI as controller. It is assessed in Section 6.5 because its schema reached version control later than the rest of the data model and by a different route.

Pallas holds account identity, organisation records, scan findings, generated reports, transiently uploaded source files, audit events, and marketing list membership. It holds no payment card data, no special category data by design, and no health or student records.

> Control data-classes-enumerated. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.
>
> Disclosure. Pallas does not collect special category data as a matter of design, but it scans customer-supplied URLs and accepts customer-uploaded documents. Content a customer submits is outside Pallas control, so the accurate statement is that Pallas does not solicit or intentionally process special category data, not that none can ever transit

the system. Uploaded source files carry a 7 day default retention precisely because they are the class most likely to contain something unintended.

Implementation. app: The 29 tables enumerated in tenant-rls-every-table, plus the three Storage buckets in migration 020.

Verification. marketing: src/pages/trust.astro, What Pallas holds and What Pallas never collects.

3.2 Nature, scope, context, and purposes

Nature. Collection, structuring, storage, retrieval, analysis, disclosure by transmission to the subprocessors listed below, restriction, and erasure. No processing is carried out for a purpose other than providing the Service.

Scope. The Service holds account identity, organisation records, scan findings, generated reports, transiently uploaded source files, audit events, and marketing list membership. It holds no payment card data, which is held by Stripe. It holds no health records and no student records.

Context. The data subjects are business users acting in a professional capacity within an organisation that has chosen the Service, plus any individual whose personal data appears incidentally in scanned content. The relationship is a business one; there is no consumer imbalance of the kind that raises the risk profile of a mass-market service.

Purposes. Accessibility scanning of Controller-designated properties, remediation tracking, governance and audit logging, reporting, and account administration.

3.3 Data flows and recipients

An authentication assertion arrives from Google or Microsoft at sign-in. The application reaches Supabase over TLS. Scans of a customer-designated URL are fetched by a Cloudflare Worker at the edge, rendered on a separate sandbox origin, and analysed. Reports and notifications leave through Resend. Payment runs through Stripe. There is no other recipient.

The full subprocessor list is published, with each provider purpose, data categories, processing location, and published certifications.

Control subprocessors-published-list. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. The 2026-08-30 inventory record names four subprocessors in use: Supabase, Cloudflare, Stripe, and Resend. The published page carries six operative entries, adding Google and Microsoft as OAuth identity providers. Both are accurate about different things: Google and Microsoft receive an authentication assertion at sign-in and store no Pallas customer data. Procurement artifacts should carry all six operative entries, because a buyer assessing identity provider dependency needs to see the two that the four-entry description omits.

Implementation. marketing: src/pages/legal/subprocessors.astro.

Verification. marketing: src/pages/trust.astro, Sub-processors section.

A new subprocessor is published with 30 days advance notice before it processes any customer data.

Control subprocessors-change-notice. Status: Implemented. A commitment the operator makes contractually. No

code enforces it.

Disclosure. This is a contractual commitment by the operator, not a technical control. Nothing in either repository enforces the notice period, and no new subprocessor has yet been added under it.

Implementation. marketing: src/pages/legal/subprocessors.astro, Change Notification section.

Certifications listed against each subprocessor are those published by that provider.

Control subprocessors-certifications-are-theirs. Status: Implemented. A statement by a subprocessor. Pallas configures and relies on it; no Pallas code implements it and neither Pallas repository can demonstrate it.

Disclosure. Pallas holds no SOC 2, ISO 27001, or PCI DSS certification of its own. Every certification named on the subprocessor list belongs to the named provider and covers that provider infrastructure, not the Pallas application built on top of it. A procurement document must never present a subprocessor certification in a way that could be read as a Pallas certification. This is the single most consequential misreading available in this inventory, so the distinction is stated wherever the certifications appear.

Implementation. marketing: src/pages/legal/subprocessors.astro.

Pallas operates no data centre, server cage, office server room, or other facility in which institutional data is held. Physical and environmental security for every system that holds institutional data belongs to Supabase and Cloudflare.

Control physical-security-subprocessor-operated. Status: Implemented. A statement by a subprocessor. Pallas configures and relies on it; no Pallas code implements it and neither Pallas repository can demonstrate it.

Disclosure. Every physical-security control a questionnaire asks about, including staffed facilities, cages, enclosing barriers, redundant and tested power, cooling, fire suppression, and carrier diversity, belongs to Supabase or Cloudflare. Pallas can neither demonstrate nor test any of them, and does not inherit them as its own controls. Answers to physical-security questions state reliance and name the provider. A buyer who needs evidence should read the provider audit reports; a Pallas assertion is not evidence of a provider control.

Implementation. vendor: Supabase and Cloudflare published data centre physical and environmental security programmes.

Verification. marketing: src/pages/trust.astro, subprocessor list with per-provider processing locations.

Pallas operates no network perimeter, no host, and no network monitoring of its own. Edge filtering, DDoS mitigation, and TLS termination are Cloudflare functions, and the database network boundary is operated by Supabase.

Control network-perimeter-subprocessor-operated. Status: Implemented. A statement by a subprocessor. Pallas configures and relies on it; no Pallas code implements it and neither Pallas repository can demonstrate it.

Disclosure. Pallas runs no firewall of its own, no network intrusion detection or prevention system, and no host-based agent, because it operates no hosts. Stateful filtering, DDoS mitigation, and TLS termination happen at the Cloudflare edge and are answered as provider-operated. Intrusion detection, intrusion prevention tuned by Pallas, next-generation persistent threat monitoring, and 24x7 intrusion monitoring are answered no: no such capability is operated by Pallas or configured by Pallas at a provider. A buyer requiring a monitored perimeter under vendor control, or a staffed security operations centre, should treat this as an open item.

Implementation. vendor: Cloudflare edge network, web application firewall and DDoS mitigation; Supabase managed

database network boundary.

Verification. marketing: workers/, whose Workers execute on the Cloudflare edge rather than on any Pallas-operated host.

3.4 Retention periods

Retention is configured per organisation. Defaults are 365 days for scan data, 7 days for uploaded source files, 730 days for reports, and 2555 days for the audit log, with the audit figure being a floor rather than a default.

Retention is configured per organisation, with defaults of 365 days for scan data, 7 days for uploaded source files, 730 days for reports, and 2555 days (seven years) for the audit log.

Control retention-per-org-policies. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_retention_policies, migration 002:374.

Verification. app: uq_pallas_retention_org, migration 002:385.

Audit rows carry a seven-year (2555 day) retention floor enforced by a database CHECK constraint. An administrator cannot configure a shorter audit retention than the regulatory minimum.

Control audit-seven-year-floor. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: chk_audit_log_retention_min_2555 on pallas_retention_policies, migration 016:34; app: pallas_apply_retention_policies, migration 042.

Verification. app: supabase/migrations/016_audit_retention_floor_check.sql.

4. Necessity and proportionality

4.1 Lawful basis

For customer organisation data, the lawful basis is the controller's, and the Service processes only on documented instruction. For user profile data, where Lonia AI is controller, the basis is performance of a contract under Article 6(1)(b): a name and an avatar are what the interface needs to attribute an action to a person. For marketing list membership the basis is consent, and the moment consent was given is recorded alongside the enrolment time.

Marketing list membership records the source of the signup and the moment consent was given, alongside the enrolment time.

Control subscriber-consent-recorded. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Disclosure. Until 2026-09-06 the pallas_subscribers and pallas_subscriber_suppression tables had no data definition file in either repository. Migration 046 in pallas-app now captures both. The schema, its constraints, and the fact that Row-Level Security is enabled on both tables are under version control and can be shown to a buyer from source.

Two limits are worth stating. Migration 046 is a recovery transcription of tables that were created by hand on the live database on 2026-08-30, so it is a description of live rather than the file live was built from; the file itself says as much and carries a verification query in its final section. And the file is in pallas-app, so no release gate in pallas-marketing checks it. Recommended operator action: run the verification query at the foot of migration 046 against the live project and correct the migration if any column, constraint, or policy count disagrees with what live returns.

Implementation. marketing: src/lib/email-gate.ts:338 onward; app: supabase/migrations/046_subscribers_table_ddl_recovery.sql.

Verification. marketing: tests/email-gate.test.ts.

4.2 Data minimisation

The user profile record carries a name and an avatar. The email address lives in the authentication record that signs a user in, not in the profile record. No phone number and no postal address is stored in either. There are no behavioural analytics, no cross-site tracking cookies, no third-party analytics scripts, no advertising pixels, and no data enrichment services.

The minimisation that matters most is on the class most likely to contain something unintended: uploaded source files carry a 7 day default retention precisely because Pallas cannot know what a customer uploads.

Uploaded source files are purged from Storage daily at 05:00 UTC by a Supabase Edge Function invoked from the database.

Control retention-source-file-purge. Status: Partial. True of the migration file set. Live database state is not provable from source, and the operator pre-check that resolves it is named in the disclosure.

Disclosure. This job does no work unless two database settings are present: app.settings.functions_base_url and app.settings.cron_secret. When either is missing the job raises a warning and returns, so source files are NOT purged and nothing fails loudly. Migration 035:619 states this in the job body. The operator must confirm both settings are set on the live database before this control is claimed to a buyer. Separately, the 2026-08-30 inventory record describes this credential as being held in Supabase Vault under the name pallas_enforce_retention_bearer. No file in either repository references Vault or that name; the mechanism on disk is a Supabase Edge Function secret named CRON_SECRET plus a database setting named app.settings.cron_secret. The inventory description and the file set disagree, and the file set is what these repositories can show.

Implementation. app: pallas_daily_source_file_retention, '0 5 * * *', scheduled at migration 035:609; app: supabase/functions/enforce-retention/index.ts.

Verification. app: supabase/functions/enforce-retention/index.ts:239 to :252.

4.3 Proportionality of the audit ledger

The audit ledger is the one design decision in Pallas that deliberately constrains a data subject right, so it gets its own proportionality argument rather than a mention.

The ledger is append-only in a strong sense and retained for at least seven years. That constrains erasure. The justification is that the Service exists to produce a documented record an organisation can rely on in front of a regulator or a procurement officer, and a record a party can quietly shorten or edit is not a record. The proportionality measures are that attribution survives erasure through a snapshot rather than through a live identity link, so severing the link under Article 17 does not orphan the record; and that lawful removal is

recorded as a redaction entry rather than being prevented, so a reviewer can distinguish a lawful purge from a concealment.

Audit attribution survives user erasure. Actor name and email are snapshotted onto the audit row at write time, so severing the identity link under Article 17 does not orphan the audit trail.

> Control audit-actor-attribution-survives-erasure. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.
>
> Implementation. app: pallas_actor_snapshot 024:379, pallas_fill_audit_actor_snapshot 040:195; app: supabase/migrations/040_user_fk_set_null.sql.
>
> Verification. app: migration 041, whose update allowlist requires actor_name_snapshot and actor_email_snapshot unchanged.

Lawful removal of an audit row, whether under a retention policy or under GDPR Article 17, appends a redaction entry recording the sequence number, the hash before, the hash after, the reason, and the timestamp. The ledger outlives the row it describes, so a reviewer can distinguish a lawful purge from a concealment.

> Control audit-lawful-redaction-recorded. Status: Implemented. Observed in the production database by catalog query on 2026-09-16 and recorded in the pallas-app deployed-objects manifest. The migration that created the object is held out of the repository as a trade secret, so the observation, not a file, is the evidence; objects are cited by name only.
>
> Disclosure. The delete allowlist is a declaration, not an authorisation control. Anything holding service_role and a SQL connection can set the redaction reason. The control is not that setting it is hard; the control is that setting it does not buy silence, because the redaction entry lands in a table that role cannot subsequently edit or truncate. An audit row can be deleted. It cannot be deleted quietly. This distinction is stated here because a reviewer will press on it, and a document that presented the flag as an access control would be overclaiming.
>
> Implementation. app: pallas_audit_delete_reason_allowed, migration 041; the redaction hash is computed inside the same migration; app: update and delete triggers on pallas_audit_events, migration 041.
>
> Verification. app: migration 041, which carries the update and delete allowlists.

4.4 Data subject rights in practice

Any signed-in user can export the personal data Pallas holds about them, across every organisation they belong to. This serves GDPR Article 15 and Article 20, CCPA and CPRA right to know, PIPEDA Principle 9, and Australian Privacy Principle 12.

> Control gdpr-art15-art20-user-export. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.
>
> Implementation. app: pallas_export_user_data, migration 039:268, extended to ten collections in migration 042; app: pallas-app-api, POST /export-user-data, 3 per hour per user, fails closed.
>
> Verification. app: supabase/migrations/039_gdpr_user_rights.sql header, controller split section.

Any signed-in user can erase their own account. This serves GDPR Article 17, CCPA and CPRA right to delete, and PIPEDA withdrawal of consent.

> Control gdpr-art17-user-erasure. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only

on 2026-08-30, not gated from the marketing repository.

Disclosure. Erasure of a user does not erase the organisation content that user created. Findings, comments, and evidence belong to the organisation as a separate controller with its own retention obligation. The identity link is severed and the audit trail keeps the attribution snapshot. A data subject asking for erasure receives this explanation rather than a silent partial deletion. The request also returns HTTP 409 rather than proceeding if the caller is the sole org_admin of any organisation, because completing it would leave that organisation unadministrable.

Implementation. app: pallas_delete_user_account, migration 039:506; app: pallas-app-api, POST /delete-user-account, requires the X-Confirm-Deletion header, 3 per hour per user, fails closed.

Verification. app: supabase/migrations/040_user_fk_set_null.sql.

Pallas distinguishes the two controller roles in its schema, not only in its policy text. Lonia AI is controller for user profile data; the customer organisation is controller for its own scan and compliance data. The user export and the organisation export are separate functions, not one filtered by the other.

Control gdpr-controller-split-documented. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_export_user_data 039:268 for the individual, pallas_export_org_data 009:13 rebuilt in 032:67 and 037:297 for the organisation.

Verification. app: supabase/migrations/039_gdpr_user_rights.sql, controller split section.

An organisation administrator can export the full organisation data set for a data subject access request.

Control gdpr-org-dsar-export. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_export_org_data, migration 009:13, rebuilt 032:67 and 037:297; app: pallas-app-api, POST /export-org-data, org_admin only, 3 per hour per user and per org, fails closed.

Verification. app: supabase/migrations/032_export_completeness.sql.

An organisation administrator can delete the organisation. Deletion cancels the Stripe subscription, wipes Storage under the organisation prefix, and removes the database rows, archiving a defined subset of audit actions first.

Control gdpr-org-deletion. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_delete_org, migration 008:70, rebuilt 022:71, 024:702, 025:245, and again in migration 041; app: pallas-app-api, POST /delete-org, org_admin plus a typed confirmation phrase, deliberately not rate limited; app: pallas_retained_audit_events, migration 021:119.

Verification. app: pallas_record_post_delete_failure, migration 025:337.

The two individual rights functions are callable only by the service role. The acting user id is derived from a verified JWT and is never accepted from a request body.

Control gdpr-rights-service-role-only. Status: Implemented. Backed by a file in the pallas-app repository, verified

read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: supabase/migrations/039_gdpr_user_rights.sql, grant section; app: pallas-app-api.

Verification. app: supabase/migrations/039_gdpr_user_rights.sql header, service_role only section.

5. Consultation

5.1 Data protection officer

Lonia AI has not designated a data protection officer, and this assessment was not reviewed by one. Article 35(2) requires the controller to seek the advice of the DPO where one is designated; none is, so the sub-article does not engage.

The Article 37(1) analysis for why none is designated: Lonia AI is not a public authority or body; its core activities do not consist of processing operations requiring regular and systematic monitoring of data subjects on a large scale, because the Service monitors web properties rather than people; and its core activities do not consist of large-scale processing of special categories of data or of criminal conviction data, which it does not solicit at all. None of the three limbs is met.

This is stated rather than omitted because a procurement questionnaire asking for the DPO's contact details must be answered with the absence and the reasoning, not with a substitute name.

5.2 Data subjects and their representatives

Article 35(9) requires the controller to seek the views of data subjects or their representatives where appropriate. No structured consultation has been carried out. The Service does publish an accessibility feedback channel with a published response target, and accessibility barriers reported through it are treated as defects; that is a real channel but it is not a consultation and is not presented as one.

5.3 Supervisory authority

No prior consultation under Article 36 has been carried out, because the residual risk assessed in Section 6 is not high after mitigation. If a mitigation below is later downgraded such that a residual risk becomes high, Article 36 consultation is the required step and this assessment is the record that would trigger it.

6. Risk assessment

Risks are stated with a likelihood and a severity, each low, medium, or high, assessed before mitigation, followed by the residual position after the mitigations named. Where a mitigation is partial or its evidence basis is weaker than demonstrated, the residual position says so, because a residual risk calculated from an unverified control is not a residual risk.

6.1 Cross-tenant data exposure

Risk. A user of one organisation reads or writes another organisation's findings, evidence, reports, or audit records.

Likelihood before mitigation: high. Severity: high. A multi-tenant compliance platform with a defective isolation boundary exposes exactly the records a customer engaged it to protect.

Mitigations.

Row-Level Security is enabled on every Pallas application table. Twenty-nine tables carry it in the migration file set.

> Control tenant-rls-every-table. Status: Implemented. True of the migration file set. Live database state is not provable from source, and the operator pre-check that resolves it is named in the disclosure.
>
> Disclosure. Twenty-nine tables carry RLS in the migration file set (twenty-four before migrations 057 and 059 added the digest-run and rule-override tables on 2026-09-16, twenty-six before migration 067 added the two crawl-state tables on 2026-09-17, and twenty-eight before migration 072 added the API-token table). Two of them, pallas_subscribers and pallas_subscriber_suppression, were created directly on the live database on 2026-08-30 and captured into migration 046 only afterwards, so any count taken before that migration was written disagrees with this one. The 2026-08-30 inventory pass recorded twenty-two on the live project. That difference has not been reconciled against the live database, because code in these repositories does not execute SQL. Before this claim is stated to a buyer with a number attached, the operator should run a live count of pg_tables joined to pg_class.relrowsecurity for tables matching pallas_%, and either correct this row or correct the 2026-08-30 inventory record. Until then the safe published form of this claim omits the count and states only that RLS is enabled on every application table.
>
> Implementation. app: ENABLE ROW LEVEL SECURITY on 29 tables across migrations 001, 002, 005, 010, 017, 021, 024, 029, 034, 041, 046, 057, 059, 067, 072.
>
> Verification. app: Search supabase/migrations/*.sql for ENABLE ROW LEVEL SECURITY and compare the count against CREATE TABLE.

Every tenant-scoped policy resolves membership through two SECURITY DEFINER helpers with a pinned search_path, rather than by re-entering the membership table under its own RLS.

> Control tenant-org-scoping-helpers. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.
>
> Implementation. app: pallas_user_has_org_access, migration 002:8, hardened in 031:54; app: pallas_user_org_role, migration 002:17, hardened in 031:68.
>
> Verification. app: supabase/migrations/031_low_severity_hardening.sql.

The evidence, source-files, and reports Storage buckets are private and carry per-organisation RLS policies on storage.objects, with read, write, update, and delete scoped to organisation role.

> Control tenant-storage-per-org. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.
>
> Implementation. app: supabase/migrations/020_storage_bucket_rls.sql; app: pallas_storage_org_access, migration 020:50.
>
> Verification. app: supabase/migrations/020_storage_bucket_rls.sql header.

Server-side operator functions run under service credentials that bypass Row-Level Security by necessity. Their scope is limited, they are versioned in source, and their actions are covered by the audit trail.

Control tenant-service-role-bypass-disclosed. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Disclosure. Service-role credentials bypass Row-Level Security. This is a property of PostgreSQL and Supabase, not a Pallas design choice, and it means tenant isolation for service-role code paths rests on the correctness of those code paths rather than on the database refusing them. The mitigating controls are that the paths are few, versioned, and audit-logged, not that they are constrained by RLS.

Implementation. app: pallas-app-api; app: supabase/functions/enforce-retention/index.ts.

Verification. marketing: src/pages/trust.astro, Security posture, Tenant isolation.

Organisation membership carries one of four roles: org_admin, manager, contributor, viewer. Read is available to any member; write and update require contributor or above; delete requires org_admin.

Control access-org-roles. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_organization_users.role, migration 001:30; app: supabase/migrations/020_storage_bucket_rls.sql, role model section.

Verification. app: pallas_user_org_role, migration 002:17 hardened 031:68.

The per-plan findings cap is enforced in the database, inside the Row-Level Security INSERT policy, not in the browser.

Control plan-finding-cap-rls. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_findings INSERT policy, migration 034:256; app: pallas_check_finding_limit, migration 034:191.

Verification. app: supabase/migrations/034_enforce_finding_limit_and_retention_purge.sql.

Residual position. Low for browser-reachable paths, where the database refuses the query rather than the application declining to make it. Not zero for service-role paths: those bypass Row-Level Security by necessity, and isolation there rests on the correctness of a small number of versioned, audit-logged code paths rather than on the database refusing them. The disclosure on the service-role row states this in the terms a reviewer should assess it in.

6.2 Unauthorised access to an account

Risk. An attacker signs in as a legitimate user.

Likelihood before mitigation: medium. Severity: high.

Mitigations. Pallas stores no passwords, so the credential-stuffing, password-reuse, and password-reset attack surfaces do not exist. Sign-in is OAuth against Google Workspace or Microsoft Entra ID, which means multi-factor authentication, conditional access, and session revocation are the customer's own identity provider policies rather than a Pallas feature the customer must trust.

Pallas authenticates users through Google and Microsoft OAuth only. It stores no passwords, offers no password login, and has no password reset flow.

Control auth-oauth-only. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: src/contexts/AuthContext.tsx:172 and :181; app: src/pages/Onboarding.tsx:123.

Verification. app: Search src/ for signInWithPassword and signUp(.

The trigger that provisions a Pallas profile on first sign-in cannot block a sign-in if it fails.

Control auth-profile-trigger-fail-safe. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: handle_pallas_new_user, migration 001:105, corrected in 003 and 021:260.

Verification. app: supabase/migrations/021_reconcile_live_state.sql section 5.

Sessions are JWT-based through Supabase Auth, and refresh tokens rotate on use.

Control auth-session-jwt. Status: Implemented. A statement by a subprocessor. Pallas configures and relies on it; no Pallas code implements it and neither Pallas repository can demonstrate it.

Disclosure. Token rotation behaviour is a property of Supabase Auth. Pallas configures it and relies on it; no Pallas code implements it, and neither Pallas repository can demonstrate it independently.

Implementation. vendor: Supabase Auth refresh token rotation.

Verification. marketing: src/pages/trust.astro, Security posture, Session security.

No authentication state, entitlement, or quota is trusted from browser storage. Authorisation decisions are made server side.

Control auth-no-browser-storage-for-entitlements. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: Row-Level Security policies on every pallas_ table, enumerated in the tenant-isolation rows below; app: pallas-app-api.

Verification. marketing: Rule sweep for browser storage API calls across src/.

Authenticated endpoints carry per-user rate limits. Endpoints that move or expose data fail closed when the limiter is unavailable; read-only endpoints degrade to load.

Control access-rate-limiting-authenticated. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Disclosure. Cloudflare Workers KV is eventually consistent across colocations, so a KV-backed counter is a burst guard rather than a hard ceiling: concurrent requests can each read the same value before any write lands. Where a hard ceiling was required, the limit is transactional in PostgreSQL instead, as with the 50 invitations per organisation per 24 hours enforced by pallas_org_invites_today. Pallas does not claim KV-backed limits are exact.

Implementation. app: pallas-app-api, checkRateLimit() over the RATE_LIMIT_KV namespace.

Verification. app: workers/pallas-app-api/wrangler.toml, rate limiting comment block.

Every unauthenticated write endpoint verifies a Cloudflare Turnstile token and refuses the request

when the verification secret is unset, rather than running with the bot gate off.

> Control access-turnstile-fails-closed. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.
>
> Implementation. marketing: src/lib/turnstile.ts; marketing: pallas-scan-email and pallas-email-gate, both returning 500 when TURNSTILE_SECRET_KEY is unset.
>
> Verification. marketing: tests/turnstile.test.ts; marketing: scripts/test-turnstile.mjs.

Residual position. Low, with the qualification that session token rotation is a property of Supabase Auth that Pallas configures and relies on rather than implements, so the evidence for it is a vendor assertion.

6.3 Undetected tampering with the audit record

Risk. An event is altered, removed, or reordered after the fact, and nobody can tell.

Likelihood before mitigation: medium. Severity: high. For a compliance product, an audit trail that can be edited without trace is worse than none, because it is relied on.

Mitigations.

Every audit event appends an entry to a SHA-256 hash chain. Each entry is hashed together with the previous entry hash, so modification, deletion, or reordering of historical events breaks the chain and is detectable.

> Control audit-sha256-ledger. Status: Implemented. Observed in the production database by catalog query on 2026-09-16 and recorded in the pallas-app deployed-objects manifest. The migration that created the object is held out of the repository as a trade secret, so the observation, not a file, is the evidence; objects are cited by name only.
>
> Implementation. app: supabase/migrations/041_audit_tamper_evidence.sql; app: hashing helpers inside migration 041; row_hash and prev_row_hash columns on pallas_audit_events; app: insert trigger on pallas_audit_events, migration 041.
>
> Verification. app: pallas_verify_audit_chain, migration 041; app: pallas_verify_audit_row, migration 041.

The ledger table is append-only in the strong sense: Row-Level Security is enabled with zero policies, and INSERT, UPDATE, DELETE, and TRUNCATE are revoked from every role including service_role. Only SECURITY DEFINER trigger functions can write to it.

> Control audit-ledger-append-only. Status: Implemented. Observed in the production database by catalog query on 2026-09-16 and recorded in the pallas-app deployed-objects manifest. The migration that created the object is held out of the repository as a trade secret, so the observation, not a file, is the evidence; objects are cited by name only.
>
> Implementation. app: pallas_audit_chain, created by migration 041 with Row-Level Security enabled; app: SECURITY DEFINER trigger functions inside migration 041.
>
> Verification. app: pallas_audit_chain, migration 041; the table answers permission denied to the anon role through the API.

Lawful removal of an audit row, whether under a retention policy or under GDPR Article 17, appends a redaction entry recording the sequence number, the hash before, the hash after, the reason, and the timestamp. The ledger outlives the row it describes, so a reviewer can distinguish a lawful purge from a concealment.

Control audit-lawful-redaction-recorded. Status: Implemented. Observed in the production database by catalog query on 2026-09-16 and recorded in the pallas-app deployed-objects manifest. The migration that created the object is held out of the repository as a trade secret, so the observation, not a file, is the evidence; objects are cited by name only.

Disclosure. The delete allowlist is a declaration, not an authorisation control. Anything holding service_role and a SQL connection can set the redaction reason. The control is not that setting it is hard; the control is that setting it does not buy silence, because the redaction entry lands in a table that role cannot subsequently edit or truncate. An audit row can be deleted. It cannot be deleted quietly. This distinction is stated here because a reviewer will press on it, and a document that presented the flag as an access control would be overclaiming.

Implementation. app: pallas_audit_delete_reason_allowed, migration 041; the redaction hash is computed inside the same migration; app: update and delete triggers on pallas_audit_events, migration 041.

Verification. app: migration 041, which carries the update and delete allowlists.

A daily anchor snapshot of the chain tip is taken at 06:00 UTC and retained. Any anchor can be exported for independent verification on request.

Control audit-daily-anchors. Status: Partial. True of the migration file set. Live database state is not provable from source, and the operator pre-check that resolves it is named in the disclosure.

Disclosure. Anchor creation is automated. External publication of anchors is an operator action and not an automated feed, so Pallas does not claim an automated external anchoring service. Until external publication begins, an anchor proves the chain was in a given state at a given time only to a party who trusts the Pallas database clock. Additionally, the cron job schedules itself inside a DO block that catches its own exception: if pg_cron was not enabled at migration time, the job was never created and the migration still reported success. Whether the job exists on the live database is an operator check against cron.job, not a fact this repository can assert.

Implementation. app: pg_cron job at '0 6 * * *', scheduled by migration 041; app: pallas_create_audit_anchor, pallas_verify_audit_anchors, pallas_list_audit_anchors, pallas_mark_audit_anchor_exported, migration 041; app: pallas-app-api, POST /audit-chain/anchor, /anchors, /verify-anchors, /anchor-exported.

Verification. app: pallas_verify_audit_anchors, migration 041.

Attempts to modify or delete an audit event are refused with specific error codes: PA030 for update, PA031 for delete, PA032 for forged chain metadata on insert.

Control audit-error-codes. Status: Implemented. Observed in the production database by catalog query on 2026-09-16 and recorded in the pallas-app deployed-objects manifest. The migration that created the object is held out of the repository as a trade secret, so the observation, not a file, is the evidence; objects are cited by name only.

Implementation. app: migration 041, immutability triggers on pallas_audit_events.

Verification. marketing: src/pages/trust.astro, Security posture, Audit trail.

Audit rows carry a seven-year (2555 day) retention floor enforced by a database CHECK constraint. An administrator cannot configure a shorter audit retention than the regulatory minimum.

Control audit-seven-year-floor. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: chk_audit_log_retention_min_2555 on pallas_retention_policies, migration 016:34; app: pallas_apply_retention_policies, migration 042.

Verification. app: supabase/migrations/016_audit_retention_floor_check.sql.

Audit action names are validated against a registry table rather than accepted as free text, so an unrecognised action cannot be written.

Control audit-action-registry. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_audit_action_registry, created migration 024:143, RLS enabled 024:164; app: pallas_assert_audit_action, migration 024:431.

Verification. app: supabase/migrations/043_audit_action_registry_documentation.sql.

Audit attribution survives user erasure. Actor name and email are snapshotted onto the audit row at write time, so severing the identity link under Article 17 does not orphan the audit trail.

Control audit-actor-attribution-survives-erasure. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_actor_snapshot 024:379, pallas_fill_audit_actor_snapshot 040:195; app: supabase/migrations/040_user_fk_set_null.sql.

Verification. app: migration 041, whose update allowlist requires actor_name_snapshot and actor_email_snapshot unchanged.

Residual position. Low for detection, which is the property actually claimed. Two qualifications a reviewer should carry forward. The delete allowlist is a declaration rather than an authorisation control: anything holding service_role and a SQL connection can set a redaction reason, and the control is that doing so does not buy silence. And external publication of the daily anchors is an operator action rather than an automated feed, so until it begins an anchor proves the chain's state at a time only to a party who trusts the Pallas database clock.

6.4 Data retained beyond its purpose

Risk. Personal data, particularly in uploaded source files, persists after the purpose for which it was collected has ended.

Likelihood before mitigation: medium. Severity: medium.

Mitigations.

Retention is configured per organisation, with defaults of 365 days for scan data, 7 days for uploaded source files, 730 days for reports, and 2555 days (seven years) for the audit log.

Control retention-per-org-policies. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_retention_policies, migration 002:374.

Verification. app: uq_pallas_retention_org, migration 002:385.

A scheduled job applies retention policies daily at 03:00 UTC for every organisation with automatic deletion enabled.

Control retention-daily-sweep. Status: Implemented. True of the migration file set. Live database state is not provable from source, and the operator pre-check that resolves it is named in the disclosure.

Disclosure. The job is scheduled by a DO block that catches its own exception, so a database on which pg_cron was not enabled at migration time has no job and the migration still reported success. Whether the job exists and is running on the live database is an operator check against the cron.job and cron.job_run_details tables. This repository can show what was intended to be scheduled; it cannot show what is scheduled.

Implementation. app: pallas-daily-retention, '0 3 * * *', scheduled at migration 021:238; app: pallas_apply_retention_policies, migration 021:197, rebuilt 029:629, migration 041, and migration 042.

Verification. app: supabase/migrations/042_restore_retention_sweeps_and_gaps.sql.

Archived audit events past their retention window are purged nightly at 04:00 UTC, and each purge run is logged.

Control retention-audit-purge. Status: Implemented. True of the migration file set. Live database state is not provable from source, and the operator pre-check that resolves it is named in the disclosure.

Disclosure. Scheduled by the same self-catching DO block pattern as the 03:00 UTC sweep. Existence of the live job is an operator check against cron.job.

Implementation. app: pallas_nightly_retention_purge, '0 4 * * *', scheduled at migration 034:581; app: pallas_purge_expired_retained_audit_events, migration 034:460; app: pallas_retention_purge_log, migration 034:405, RLS enabled 034:416.

Verification. app: supabase/migrations/034_enforce_finding_limit_and_retention_purge.sql section 9.

Uploaded source files are purged from Storage daily at 05:00 UTC by a Supabase Edge Function invoked from the database.

Control retention-source-file-purge. Status: Partial. True of the migration file set. Live database state is not provable from source, and the operator pre-check that resolves it is named in the disclosure.

Disclosure. This job does no work unless two database settings are present: app.settings.functions_base_url and app.settings.cron_secret. When either is missing the job raises a warning and returns, so source files are NOT purged and nothing fails loudly. Migration 035:619 states this in the job body. The operator must confirm both settings are set on the live database before this control is claimed to a buyer. Separately, the 2026-08-30 inventory record describes this credential as being held in Supabase Vault under the name pallas_enforce_retention_bearer. No file in either repository references Vault or that name; the mechanism on disk is a Supabase Edge Function secret named CRON_SECRET plus a database setting named app.settings.cron_secret. The inventory description and the file set disagree, and the file set is what these repositories can show.

Implementation. app: pallas_daily_source_file_retention, '0 5 * * *', scheduled at migration 035:609; app: supabase/functions/enforce-retention/index.ts.

Verification. app: supabase/functions/enforce-retention/index.ts:239 to :252.

Retention settings cause actual deletion rather than being stored as a preference. The sweep issues DELETE statements against the tables it governs.

Control retention-deletion-actually-deletes. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Implementation. app: pallas_apply_retention_policies, migration 042.

Verification. app: migration 041, which enumerates the three code paths that lawfully remove audit rows.

Residual position. Medium rather than low, and this is the most important honest answer in this assessment. Three of the retention controls carry a live-unverifiable basis for one shared reason: the scheduled jobs are created by a database DO block that catches its own exception, so a database on which pg_cron was not enabled at migration time has no job and the migration still reported success. The source file purge additionally does no work unless two database settings are present, and when either is missing it raises a warning and returns rather than failing loudly.

A retention control that might not be running is not a mitigation a controller can rely on. The remediation is an operator check against the cron.job and cron.job_run_details tables and against the two database settings, and it is listed in Section 8 as an action rather than described here as done.

6.5 Marketing list schema captured only as a recovery transcription

Risk. The pallas_subscribers and pallas_subscriber_suppression tables were created directly on the live database on 30 August 2026 and had no data definition file in either repository until migration 046 was written on 6 September 2026. That migration is a transcription of live rather than the file live was built from, so the schema a reviewer reads and the schema the database holds are two separate things that have not yet been compared.

Likelihood before mitigation: low. Severity: medium. The data class is a list of email addresses and consent timestamps, which is low sensitivity, but a schema that has never been reconciled against the database it describes cannot be relied on to describe it.

Mitigations.

Marketing list membership records the source of the signup and the moment consent was given, alongside the enrolment time.

Control subscriber-consent-recorded. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Disclosure. Until 2026-09-06 the pallas_subscribers and pallas_subscriber_suppression tables had no data definition file in either repository. Migration 046 in pallas-app now captures both. The schema, its constraints, and the fact that Row-Level Security is enabled on both tables are under version control and can be shown to a buyer from source. Two limits are worth stating. Migration 046 is a recovery transcription of tables that were created by hand on the live database on 2026-08-30, so it is a description of live rather than the file live was built from; the file itself says as much and carries a verification query in its final section. And the file is in pallas-app, so no release gate in pallas-marketing checks it. Recommended operator action: run the verification query at the foot of migration 046 against the live project and correct the migration if any column, constraint, or policy count disagrees with what live returns.

Implementation. marketing: src/lib/email-gate.ts:338 onward; app: supabase/migrations/046_subscribers_table_ddl_recovery.sql.

Verification. marketing: tests/email-gate.test.ts.

The subscriber tables carry Row-Level Security with no policies, so the service role credential held server side is the only credential that can read or write them. No browser-reachable key can touch the list.

Control subscriber-rls-no-policies. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Disclosure. This claim rested on the live database state described in the Worker configuration comments until 2026-09-06. Migration 046 in pallas-app now carries the statement that enables Row-Level Security on both tables, and creates no policy on either, so the restrictive posture is readable from source rather than asserted. Two limits remain. The migration is a recovery transcription of tables created by hand on live on 2026-08-30 rather than the file live was built from, and it is in pallas-app, so no pallas-marketing release gate checks it. The operator procedure above is what closes both, and section V of migration 046 is the query that runs it.

Implementation. app: supabase/migrations/046_subscribers_table_ddl_recovery.sql; marketing: workers/pallas-email-gate/wrangler.toml and workers/pallas-scan-email/wrangler.toml secret documentation; marketing: src/lib/email-gate.ts:79.

Verification. marketing: Confirm RLS enabled and policy count zero on both tables in the Supabase dashboard.

The unsubscribe route is dispatched ahead of the CORS, method, and Turnstile checks, so a misconfigured bot gate for joining a list cannot take the mechanism for leaving it offline.

Control subscriber-unsubscribe-always-reachable. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. The route stays reachable but it does not claim a removal it could not perform. When the Supabase credentials are missing it reports the removal as unfinished rather than confirming it.

Implementation. marketing: src/lib/unsubscribe-route.ts; marketing: pallas-email-gate, GET and POST /unsubscribe.

Verification. marketing: tests/unsubscribe-route.test.ts and tests/unsubscribe.test.ts.

Unsubscribing writes to a suppression table keyed on a hash of the address, so a later signup of the same address is refused without retaining the plaintext address on the suppression list.

Control subscriber-suppression-list. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.

Implementation. marketing: src/lib/unsubscribe.ts:245 and :284; marketing: src/lib/email-gate.ts:327.

Verification. marketing: tests/unsubscribe.test.ts.

Every marketing message carries the operator physical postal address. The sending Worker refuses to send when that value is unset rather than sending a message missing it.

Control subscriber-postal-address-required. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. This addresses the identification duty in 15 U.S.C. 7704(a)(5), the Australian Spam Act 2003, and the UK Privacy and Electronic Communications Regulations. The control is that the send path fails rather than degrades.

Implementation. marketing: pallas-scan-email, POSTAL_ADDRESS secret, required with no default and no substitute string; marketing: src/lib/unsubscribe.ts, getPostalAddress().

Verification. marketing: tests/scan-email-compliance.test.ts.

Residual position. Medium, until the operator runs the verification query in the final section of migration 046

against the live project and records that every column, constraint, and policy count agrees. The controls themselves are sound, two of them are demonstrated by tests in the marketing repository, and a reviewer can now be shown the table definition those controls operate on; what is outstanding is the confirmation that the definition is accurate.

6.6 Exposure of personal data through the scan path

Risk. The scan proxy is induced to fetch an address it should not, or fetched third-party markup reaches a context where it can act.

Likelihood before mitigation: medium. Severity: medium.

Mitigations.

The public scan tool validates every submitted URL against a hostname and IP-literal denylist before fetching it, blocking loopback, link-local, unique-local, IPv4-mapped private addresses, cloud metadata endpoints, and the private, CGNAT, and multicast IPv4 ranges.

> Control appsec-ssrf-guard. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.
>
> Implementation. marketing: src/lib/ssrf-guard.ts; app: workers/pallas-cors-proxy/src/index.ts, isBlockedHost and isBlockedIpv4.
>
> Verification. marketing: tests/ssrf-guard.test.ts.

Fetched third-party responses are capped at 10 MB, enforced against the declared Content-Length and again against the bytes actually received.

> Control appsec-response-cap. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.
>
> Implementation. marketing: src/lib/response-cap.ts; app: workers/pallas-cors-proxy/src/index.ts:261, MAX_RESPONSE_BYTES.
>
> Verification. marketing: tests/response-cap.test.ts.

Scanned third-party markup renders on a dedicated origin with nothing in its storage, pinned shut with Content-Security-Policy, so retrieved markup cannot reach the application session, cookies, or IndexedDB.

> Control appsec-sandboxed-rendering. Status: Implemented. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.
>
> Implementation. app: workers/pallas-scanner-sandbox/src/index.ts.
>
> Verification. app: docs/OPERATOR_ACTIONS.md section 1.1, mitigations table.

Untrusted content is escaped before it reaches HTML or email. The only place untrusted input meets raw HTML on the marketing site is JSON-LD, fed exclusively by a serializer that escapes the closing angle bracket and the U+2028 and U+2029 line separators.

> Control appsec-output-escaping. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.

Implementation. marketing: src/lib/jsonld.ts, serializeJsonLd(); marketing: src/lib/html-escape.ts; marketing: src/lib/safe-url.ts.

Verification. marketing: tests/safe-url.test.ts and tests/scan-report.test.ts; marketing: SECURITY_ADVISORY_TRIAGE.md, section titled The one place untrusted input meets HTML.

Residual position. Low for the direct server-side request forgery vector, which the denylist closes. Not closed for DNS rebinding, which is a property of the Cloudflare Workers runtime rather than a gap in the guard, and which is disclosed publicly rather than omitted.

The scan proxy cannot fully defend against DNS rebinding. This is disclosed publicly rather than omitted.

Control appsec-dns-rebinding-limitation. A Cloudflare Worker resolves DNS inside fetch(). The runtime exposes no hook to read the address a hostname resolved to and no way to pin a connection to an already-validated address, so there is no point at which the Worker can compare the address it approved against the address it connected to. This is a property of the runtime rather than a gap in the guard. What limits the blast radius is that the Worker egresses from Cloudflare edge to the public internet: it is not inside a private network, no Pallas internal service is reachable from it, and Workers have no cloud instance metadata service. Mitigations in place are the denylist, a 10 MB response cap enforced against both the declared Content-Length and the bytes actually received, and rendering of fetched markup on a dedicated origin with empty storage pinned shut by Content-Security-Policy. A full fix requires a resolving egress proxy, which is new infrastructure rather than an edit to this Worker.

The blast radius is what keeps the residual position acceptable. The Worker egresses from Cloudflare edge to the public internet, it is not inside a private network, no Pallas internal service is reachable from it, and Workers have no cloud instance metadata service.

6.7 Loss of availability or of data

Risk. Data is lost, or the Service is unavailable for long enough to matter to a customer's own compliance deadline.

Likelihood before mitigation: low. Severity: medium.

Mitigations.

Application data is held in Supabase managed PostgreSQL with provider-operated automated backups and point-in-time recovery. The recoverable window is set by the provider tier.

Control backup-provider-operated. Status: Implemented. A statement by a subprocessor. Pallas configures and relies on it; no Pallas code implements it and neither Pallas repository can demonstrate it.

Disclosure. Pallas runs no backup schedule of its own. The backup posture is entirely the Supabase provider tier posture, and the operator has not customised it.

Implementation. vendor: Supabase automated backups and point-in-time recovery.

Verification. marketing: src/pages/trust.astro, Backup and disaster recovery questionnaire answer.

Pallas does not publish a recovery time objective or a recovery point objective.

Control backup-no-published-rto-rpo. Status: Not implemented. A commitment the operator makes contractually. No code enforces it.

Disclosure. No RTO or RPO is published, because a number that cannot be evidenced from a restore test is worth nothing on a questionnaire. No restore test has been performed. A negotiated RTO and RPO with service credits is an Enterprise onboarding item. Any questionnaire field asking for an RTO or RPO must be answered as not published rather than filled with an estimate.

Verification. marketing: src/pages/trust.astro:126.

Pallas has not performed a documented restore test.

Control backup-no-restore-test. Status: Not implemented. A commitment the operator makes contractually. No code enforces it.

Disclosure. No restore test has been carried out, so the recoverability of the backups is a provider assertion rather than a demonstrated property. This is a genuine gap and it is listed as one. A buyer for whom tested recoverability is a control requirement should treat this as an open item, and the operator should schedule a restore test into a non-production project as the remediation.

Pallas does not maintain a business continuity plan or a disaster recovery plan that is documented, assigned to a named owner, and exercised annually.

Control continuity-no-tested-bcp-drp. Status: Not implemented. A commitment the operator makes contractually. No code enforces it.

Disclosure. There is no annually tested BCP or DRP with a named owner. Continuity rests on the managed platform underneath: Supabase provider backups with point-in-time recovery, and Cloudflare for edge and Pages hosting. Because no restore test has been performed, recoverability is a provider assertion rather than a demonstrated property. A questionnaire item asking for a documented and tested plan is answered no, not answered yes from the existence of provider backups.

Verification. marketing: src/pages/trust.astro, Backup and disaster recovery questionnaire answer.

Residual position. Medium, and stated as a genuine gap rather than mitigated. Pallas runs no backup schedule of its own; the posture is entirely the Supabase provider tier posture. No restore test has been carried out, so recoverability is a provider assertion rather than a demonstrated property. No recovery time objective and no recovery point objective is published, because a number that cannot be evidenced from a restore test is worth nothing on a questionnaire.

6.8 Breach detection and notification delay

Risk. A breach occurs and the affected controller is not told in time to meet its own Article 33 obligation.

Likelihood before mitigation: medium. Severity: high.

Mitigations.

Pallas notifies affected customers without undue delay and in any event within 72 hours of becoming aware of a breach affecting their data. The clock starts on awareness, not on confirmation.

Control incident-72-hour-notification. Status: Implemented. A commitment the operator makes contractually. No code enforces it.

Disclosure. This is a contractual commitment. No technical control enforces it and no artifact in either repository can demonstrate it has been met, because no breach has occurred. A buyer should treat it as a term to be captured in the Data Processing Agreement rather than as a verified capability.

Implementation. marketing: src/pages/trust.astro, Breach commitments, Customer notification; marketing: src/pages/legal/dpa.astro.

Authority notification follows the jurisdiction: Ireland DPC, UK ICO, and EU supervisory authorities within 72 hours; Canada OPC under PIPEDA; Australia OAIC under the Notifiable Data Breaches scheme.

Control incident-authority-notification. Status: Implemented. A commitment the operator makes contractually. No code enforces it.

Disclosure. A contractual and regulatory commitment rather than a technical control. Pallas has not been required to make such a notification, so there is no track record to evidence.

Implementation. marketing: src/pages/trust.astro, Breach commitments, Authority notification.

Worker observability and logging are enabled on every deployed Worker so that failures surface rather than passing silently.

Control incident-detection-monitoring. Status: Partial. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. Observability is enabled and logs are collected. There is no security information and event management system, no alerting pipeline, no on-call rotation, and no documented incident response runbook with named roles. Detection today means an operator reading the Cloudflare dashboard. Any questionnaire asking about 24 by 7 monitoring, SIEM, or a tested incident response plan must be answered no. Pallas is operated by a single operator and a procurement document that implied a staffed security operations centre would be false.

Implementation. marketing: workers/pallas-scan-email/wrangler.toml and workers/pallas-email-gate/wrangler.toml, observability and observability.logs enabled; app: All six pallas-app Workers, observability enabled in wrangler.toml.

Verification. marketing: workers/pallas-email-gate/wrangler.toml observability comment.

A full technical postmortem is provided to affected customers within 30 days.

Control incident-post-incident-report. Status: Implemented. A commitment the operator makes contractually. No code enforces it.

Implementation. marketing: src/pages/trust.astro, Breach commitments, Post-incident report.

A subprocessor breach triggers the same 72 hour notification cascade, from the subprocessor to Pallas to the customer.

Control incident-subprocessor-cascade. Status: Partial. A commitment the operator makes contractually. No code enforces it.

Disclosure. The Pallas leg of this cascade is a commitment Pallas controls. The subprocessor leg depends on each provider notifying Pallas, which is governed by that provider terms and is outside Pallas control. The end-to-end 72 hours is therefore contingent on subprocessor performance, and a buyer requiring a guaranteed end-to-end window should raise it during Enterprise contracting rather than rely on this statement.

Implementation. marketing: src/pages/trust.astro, Breach commitments.

Pallas does not publish a cyber liability insurance policy or a certificate of cover.

Control insurance-no-published-cover. Status: Not implemented. A commitment the operator makes contractually. No

code enforces it.

Disclosure. No cyber liability certificate is published. A questionnaire field asking for evidence of cover is answered as not published rather than left blank or answered from intent, and where a contract requires evidence of cover the matter is addressed in the agreement during procurement.

Verification. marketing: src/pages/trust.astro, Cyber liability insurance, status Not yet formalized.

Residual position. Medium. The notification commitments are contractual and no technical control enforces them. Detection today means an operator reading the Cloudflare dashboard: there is no security information and event management system, no alerting pipeline, no on-call rotation, and no incident response runbook with named roles. Pallas is operated by a single operator, and a document that implied a staffed security operations centre would be false. The subprocessor leg of the notification cascade depends on each provider notifying Pallas, which is governed by that provider's terms and is outside Pallas control.

6.9 International transfer risk

Risk. Personal data transferred out of the EEA, the UK, or Switzerland is accessed by a public authority in a way incompatible with the exporting jurisdiction's standard.

Likelihood before mitigation: low. Severity: medium.

Mitigations. Standard Contractual Clauses Module Two incorporated by reference, the UK International Data Transfer Agreement or Addendum for UK transfers, and the Swiss amendments for Swiss transfers. Encryption in transit and at rest. A pre-populated Transfer Impact Assessment covering the subprocessor chain is issued on request.

Data is encrypted at rest with AES-256 through Supabase-managed infrastructure.

Control encryption-at-rest. Status: Implemented. A statement by a subprocessor. Pallas configures and relies on it; no Pallas code implements it and neither Pallas repository can demonstrate it.

Disclosure. Encryption at rest is provided and attested by Supabase. No Pallas code implements it and neither Pallas repository can demonstrate it. A buyer who requires independent evidence should request the Supabase SOC 2 Type II report rather than a Pallas assertion.

Implementation. vendor: Supabase managed PostgreSQL and Storage encryption at rest.

Verification. vendor: Supabase SOC 2 Type II report, available to the operator under NDA.

All connections are TLS 1.3, enforced by Cloudflare and Supabase.

Control encryption-in-transit. Status: Implemented. A statement by a subprocessor. Pallas configures and relies on it; no Pallas code implements it and neither Pallas repository can demonstrate it.

Disclosure. TLS termination and version are properties of Cloudflare and Supabase. What this repository demonstrates is the header set and the build gate that refuses to ship without it, not the negotiated cipher.

Implementation. vendor: Cloudflare edge TLS and Supabase connection TLS; marketing: public/_headers.

Verification. marketing: scripts/csp-hashes.mjs.

The default processing location is the United States. Supabase hosts the database and Storage in us-east-1; Cloudflare serves from its global edge network.

Control data-location-default-us. Status: Implemented. A statement by a subprocessor. Pallas configures and relies on it; no Pallas code implements it and neither Pallas repository can demonstrate it.

Implementation. vendor: Supabase project region us-east-1; marketing: src/pages/legal/subprocessors.astro.

Verification. marketing: src/pages/trust.astro, Data residency options.

Processing takes place in the United States only. No European Union hosting exists and none is offered.

Control data-location-eu-option. Status: Not implemented. A commitment the operator makes contractually. No code enforces it.

Disclosure. No customer data is processed in the European Union and no EU infrastructure subprocessor is engaged or planned. Any procurement document, questionnaire response, or sales conversation must present United States processing as the only option and must not present EU hosting as available, on request, or on a roadmap.

Implementation. marketing: src/pages/trust.astro, Data residency options; marketing: src/pages/legal/subprocessors.astro, six operative entries and no planned entry.

Residual position. Medium for a controller that requires processing inside the European Union, because no EU processing exists: all customer data is processed in the United States and no EU infrastructure subprocessor is engaged or planned. Any onboarding conversation must present United States processing as the only option.

7. Summary of residual risk

No residual risk in this assessment is high after mitigation, which is why no Article 36 prior consultation has been carried out. Four residual positions are medium and each has a named remediation:

1. **Retention enforcement (6.4).** Medium until the operator confirms the scheduled jobs exist on the live database and the two source-file purge settings are present.
2. **Marketing list schema (6.5).** Medium until the transcription in migration 046 is reconciled against the live database.
3. **Availability and recovery (6.7).** Medium until a documented restore test is carried out. A negotiated recovery objective remains an Enterprise contracting item either way.
4. **Breach detection (6.8) and international transfer (6.9).** Medium, and both are structural rather than fixable by a code change: single-operator detection and the absence of an EU deployment.

A controller for whom any of the four is unacceptable should raise it during onboarding rather than rely on this assessment's conclusion, because the conclusion is Lonia AI's risk appetite and the controller's may differ.

8. Actions arising

These are the actions this assessment generates. They are stated as open, not as done.

1. Run the operator pre-check against cron.job and cron.job_run_details for the four scheduled Pallas jobs, and against app.settings.functions_base_url and app.settings.cron_secret. Record the output. Until then, the retention controls carrying a live-unverifiable basis stay at that basis.

2. The live definition of pallas_subscribers and pallas_subscriber_suppression is captured in migration 046, so the marketing list schema is versioned like every other Pallas table. Run the verification query in the final section of that migration against the live project and record that the transcription matches. Until then the two subscriber controls carrying an app-repository basis stay at that basis.
3. Schedule a documented restore test into a non-production Supabase project, and record the result. This is the only route from a provider assertion to a demonstrated recovery property.
4. Reconcile the Row-Level Security table count between the migration file set, which now carries twenty-nine, and the live database, which recorded twenty-two at the 30 August 2026 pass.
5. Review this assessment when any of the above completes, when a subprocessor is added, when a control's status changes in the inventory, or in any event by 30 August 2027.

9. Contact

Questions about this assessment, requests for the operator pre-check output for a specific onboarding review, and general security questions go to support@lonia.ai. The Data Processing Agreement, the Standard Contractual Clauses, and the Transfer Impact Assessment are published at pallas.lonia.ai/trust.

Appendix A. Controls cited

This document cites 61 of the 84 rows in the Pallas control inventory. Each is listed below with the status and evidence basis it carried at the 2026-08-30 inventory pass. A row absent from this list is not claimed by this document.

- **auth-oauth-only** (Authentication). Implemented.
- **auth-profile-trigger-fail-safe** (Authentication). Implemented.
- **auth-session-jwt** (Authentication). Implemented.
- **auth-no-browser-storage-for-entitlements** (Authentication). Implemented.
- **tenant-rls-every-table** (Multi-tenancy isolation). Implemented.
- **tenant-org-scoping-helpers** (Multi-tenancy isolation). Implemented.
- **tenant-storage-per-org** (Multi-tenancy isolation). Implemented.
- **tenant-service-role-bypass-disclosed** (Multi-tenancy isolation). Implemented.
- **audit-sha256-ledger** (Tamper-evident audit chain). Implemented.
- **audit-ledger-append-only** (Tamper-evident audit chain). Implemented.
- **audit-lawful-redaction-recorded** (Tamper-evident audit chain). Implemented.
- **audit-daily-anchors** (Tamper-evident audit chain). Partial.
- **audit-error-codes** (Tamper-evident audit chain). Implemented.
- **audit-seven-year-floor** (Tamper-evident audit chain). Implemented.
- **audit-action-registry** (Tamper-evident audit chain). Implemented.
- **audit-actor-attribution-survives-erasure** (Tamper-evident audit chain). Implemented.
- **gdpr-art15-art20-user-export** (Data subject rights). Implemented.
- **gdpr-art17-user-erasure** (Data subject rights). Implemented.
- **gdpr-controller-split-documented** (Data subject rights). Implemented.

- **gdpr-org-dsar-export** (Data subject rights). Implemented.
- **gdpr-org-deletion** (Data subject rights). Implemented.
- **gdpr-rights-service-role-only** (Data subject rights). Implemented.
- **retention-per-org-policies** (Retention and deletion). Implemented.
- **retention-daily-sweep** (Retention and deletion). Implemented.
- **retention-audit-purge** (Retention and deletion). Implemented.
- **retention-source-file-purge** (Retention and deletion). Partial.
- **retention-deletion-actually-deletes** (Retention and deletion). Implemented.
- **plan-finding-cap-rls** (Plan and quota enforcement). Implemented.
- **encryption-at-rest** (Encryption). Implemented.
- **encryption-in-transit** (Encryption). Implemented.
- **access-org-roles** (Access control). Implemented.
- **access-rate-limiting-authenticated** (Access control). Implemented.
- **access-turnstile-fails-closed** (Access control). Implemented.
- **appsec-ssrf-guard** (Application security). Implemented.
- **appsec-dns-rebinding-limitation** (Application security). Partial.
- **appsec-response-cap** (Application security). Implemented.
- **appsec-sandboxed-rendering** (Application security). Implemented.
- **appsec-output-escaping** (Application security). Implemented.
- **subscriber-consent-recorded** (Marketing subscriber management). Implemented.
- **subscriber-rls-no-policies** (Marketing subscriber management). Implemented.
- **subscriber-unsubscribe-always-reachable** (Marketing subscriber management). Implemented.
- **subscriber-suppression-list** (Marketing subscriber management). Implemented.
- **subscriber-postal-address-required** (Marketing subscriber management). Implemented.
- **data-classes-enumerated** (Data classes and processing location). Implemented.
- **data-location-default-us** (Data classes and processing location). Implemented.
- **data-location-eu-option** (Data classes and processing location). Not implemented.
- **subprocessors-published-list** (Subprocessors). Implemented.
- **subprocessors-change-notice** (Subprocessors). Implemented.
- **subprocessors-certifications-are-theirs** (Subprocessors). Implemented.
- **physical-security-subprocessor-operated** (Subprocessors). Implemented.
- **network-perimeter-subprocessor-operated** (Subprocessors). Implemented.
- **incident-72-hour-notification** (Incident response). Implemented.
- **incident-authority-notification** (Incident response). Implemented.
- **incident-detection-monitoring** (Incident response). Partial.
- **incident-post-incident-report** (Incident response). Implemented.
- **incident-subprocessor-cascade** (Incident response). Partial.
- **insurance-no-published-cover** (Incident response). Not implemented.
- **backup-provider-operated** (Backup and recovery). Implemented.
- **backup-no-published-rto-rpo** (Backup and recovery). Not implemented.

- **backup-no-restore-test** (Backup and recovery). Not implemented.
- **continuity-no-tested-bcp-drp** (Backup and recovery). Not implemented.