LONIA AI / PALLAS

Pallas security whitepaper

Controls, evidence, and the limits of both

An account of how Pallas by Lonia AI is secured, written so that a reviewer can tell the difference between a control this repository can demonstrate, a control that lives in a sibling repository, a control that depends on live database state, and a statement a subprocessor makes that Pallas relies on and cannot verify.

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

Jurisdictions addressed. Not jurisdiction-specific. Statutes are named at the control they explain rather than in a jurisdiction section. The Data Processing Agreement and the Data Protection Impact Assessment carry the jurisdictional analysis.

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 whitepaper describes the Pallas security posture as of the 2026-08-30 control inventory pass. It is published in full rather than issued on request, because a security posture a buyer has to ask for is one they cannot compare.

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

How to read this document

Most security whitepapers are a list of capabilities. This one is a list of capabilities with their evidence attached, because the two are not the same thing and a reviewer is entitled to know which they are being handed.

Every control claim below is rendered directly from the Pallas control inventory, a file in the Lonia AI marketing repository at src/data/control-inventory.ts. That file is the single source of truth for every control Pallas states to a buyer. Nothing in this document is typed by hand into prose: the claim sentence, the status, the evidence basis, the implementation citations, and any disclosure text are read out of the inventory at the moment this PDF is generated, and a build gate refuses to release the site if this document falls behind the inventory it was generated from.

That arrangement exists to close one specific failure. A procurement document that claims a control which does not exist is legally worse than having no document at all, and the way such a claim gets in is not dishonesty but paraphrase: a status word gets dropped, a caveat becomes a footnote, a subprocessor's certification comes to read as the vendor's own. Removing the opportunity to paraphrase removes the failure.

The four evidence bases, and why the distinction is load-bearing

Each control below carries one of five evidence bases. Four of them mean something weaker than a reviewer might assume from the claim alone.

- **Demonstrated.** Backed by a file in the pallas-marketing repository and checked by a release gate. This is the strongest basis in this document and it applies to a minority of rows.
- **App repository.** Backed by a file in pallas-app, the sibling repository that holds the authenticated application, its Supabase migration set, and six Cloudflare Workers. Verified read-only on 30 August 2026. A reviewer who can only see the marketing repository is taking Pallas at its word.
- **Live unverifiable.** True of the migration file set. Code in these repositories never executes SQL against the production database, so some claims are about the state of a running system that no file can prove. Each such row names the operator pre-check that resolves it. This is not a hedge: migration 045 in pallas-app records the file set and the live database having actually disagreed once, so the possibility is a documented historical fact.
- **Vendor assertion.** A statement by Supabase, Cloudflare, Stripe, or Resend. Pallas configures and relies on it. No Pallas code implements it.
- **Operator commitment.** A commitment made contractually. No code enforces it.

What Pallas does not hold

Pallas holds no SOC 2 report, no ISO 27001 certificate, and no PCI DSS attestation of its own. Certifications named anywhere in Pallas materials, including on the published subprocessor list, belong to the named provider and cover that provider's infrastructure, not the Pallas application built on top of it. This is the single most consequential misreading available in this document, so it is stated here, at the front, before any control that a certification might seem to support.

Product and architecture in brief

Pallas is an accessibility compliance platform. It scans web content against the WCAG 2.2 success criteria, tracks findings to remediation, keeps a tamper-evident record of who changed what, and produces the reports an organisation needs when a regulator or a procurement officer asks. It spans two repositories and one authenticated application.

- **pallas-marketing.** The static site at pallas.lonia.ai, its build gates, and two Cloudflare Workers: pallas-scan-email, which sends a scan report by email, and pallas-email-gate, which handles newsletter and resource-download signup and the unsubscribe route.
- **pallas-app.** The authenticated application at app.pallas.lonia.ai, the Supabase migration set, one Supabase Edge Function, and six Cloudflare Workers including the API, the checkout handler, the CORS proxy that fetches pages to be scanned, and the sandbox origin that renders them.

Data is held in Supabase managed PostgreSQL and Supabase Storage in the United States. Cloudflare serves the sites and runs the Workers. Stripe handles payment. Resend delivers transactional and report email. Google and Microsoft are OAuth identity providers and store no Pallas customer data.

There is no server the operator administers, no virtual machine, no container host, and no private network. That shapes the threat model throughout: several classes of attack that dominate a conventional deployment do not apply, and one class, covered under Known limitations, is harder to close than it would be in a conventional deployment.

Authentication

Pallas stores no passwords. There is no password login, no password reset flow, and therefore no password reset attack surface, no credential stuffing surface, and no password database to breach. Sign-in is OAuth against Google Workspace or Microsoft Entra ID. There are no magic links, which are a common phishing vector, and no SMS codes, which are a SIM-swap vector.

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/.

Multi-tenancy isolation

Every Pallas application table carries PostgreSQL Row-Level Security. Tenant scoping is resolved by two SECURITY DEFINER helper functions with a pinned search_path rather than by re-entering the membership table under its own policy, which is the recursion that makes a naive RLS design fail open under load. Storage buckets are private and carry their own per-organisation policies on storage.objects.

The honest part of this section is the last row. Service-role credentials bypass Row-Level Security. That is a property of PostgreSQL and Supabase rather than a Pallas design choice, and it means isolation for service-role code paths rests on the correctness of those paths rather than on the database refusing them.

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.

Tamper-evident audit chain

This is the control Pallas is built around, and it is the one most worth a reviewer's attention, because "audit log" is a phrase that covers everything from an append-only table to a text file nobody reads.

Every audit event appends an entry to a SHA-256 hash chain, each entry hashed together with the previous entry's hash. The ledger table has Row-Level Security enabled with zero policies, and INSERT, UPDATE, DELETE, and TRUNCATE are revoked from every role including service_role, so only SECURITY DEFINER trigger functions can write to it. Lawful removal of an audit row, 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 time.

The design goal is not that an audit row cannot be deleted. It is that an audit row cannot be deleted quietly. The disclosure on the redaction row states that distinction in the terms a reviewer will press on it.

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.

Encryption

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.

Production secrets are held in the platform secret store and never in either repository. Worker configuration files carry variable names only.

Control encryption-no-secrets-in-repo. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. Two Cloudflare KV namespace identifiers are committed, at workers/pallas-scan-email/wrangler.toml and in the pallas-app Workers. A KV namespace id is an identifier rather than a credential: reaching the namespace still requires authenticated access to the Cloudflare account. This is recorded as a deliberate decision, with the note that the ids should move to deploy-time values if either repository is shared with a buyer, an auditor, or a contractor.

Implementation. marketing: workers/pallas-scan-email/.dev.vars.example and workers/pallas-email-gate/.dev.vars.example; marketing: .gitignore.

Verification. marketing: Rule sweep for key, token, price identifier, and webhook secret strings.

Finding comments, evidence descriptions and code snippets, and evidence files are encrypted at rest by the database and storage provider under a key the provider holds, and in transit by TLS. They are not client-side encrypted: every member of the organization can read them, they are included in the organization's data export, and the Processor can read them as any SaaS operator can.

Control encryption-member-prose-provider-held-key. 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: docs/DATABASE_ENFORCEMENT.md:86, What is and is not encrypted; app: src/audit/auditMetadataKeys.ts, AUDIT_METADATA_KEYS.

Verification. app: tests/audit/auditMetadataKeys.test.ts.

Access control and rate limiting

Organisation membership carries one of four roles. An organisation cannot be left without an administrator. Losing the privilege that let a user invite others revokes their outstanding invitations.

Rate limiting is worth reading closely, because the disclosures name where the limits are exact and where they are not. Cloudflare Workers KV is eventually consistent across colocations, so a KV-backed counter is a burst guard rather than a hard ceiling. Where a hard ceiling was required, the limit is transactional in PostgreSQL instead. Pallas does not claim KV-backed limits are exact.

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.

An organisation cannot be left without an administrator. Removing or demoting the last org_admin is refused at the database layer.

Control access-last-admin-guard. 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_guard_last_admin, migration 010:106, rebuilt 025:162.

Verification. app: supabase/migrations/015_drop_duplicate_last_admin_trigger.sql.

When a user loses the privilege that let them invite others, their outstanding invitations are revoked.

Control access-privilege-loss-revokes-invites. 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_revoke_invites_on_privilege_loss, migration 029:266, rebuilt 040:260.

Verification. app: supabase/migrations/029_medium_security_hardening.sql.

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.

Unauthenticated marketing endpoints are rate limited. The newsletter and resource gate endpoint carries a per-IP burst ceiling of 12 per 60 seconds and a per-email burst ceiling of 5 per 60 seconds, keyed on the normalised address; the anonymous checkout endpoint carries 5 per 60 seconds.

Control access-rate-limiting-anonymous. Status: Partial. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. The Cloudflare native rate limiting binding supports a 10 second or 60 second period only. The email gate therefore carries per-IP and per-email burst ceilings and nothing else: there is no daily global signup ceiling on that endpoint and none is claimed. What bounds abuse there is the Turnstile gate plus the unique constraint on (email, source) in pallas_subscribers, which makes a repeat submission of the same pair produce no new row. A daily ceiling would require a Durable Object binding the operator has not provisioned.

Implementation. marketing: workers/pallas-email-gate/wrangler.toml, native SIGNUP_RATE_LIMIT binding, 12 per 60 seconds; marketing: workers/pallas-email-gate/wrangler.toml, native SIGNUP_EMAIL_RATE_LIMIT binding, 5 per 60 seconds, keyed on the trimmed and lowercased email; the Worker refuses with a 500 if either binding is absent; app: workers/pallas-checkout/wrangler.toml, native PURCHASE_LIMITER binding, 5 per 60 seconds, Worker returns 503 if the binding is absent.

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

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.

Pallas does not operate a documented background-screening programme, a security-awareness programme, or a recurring mandatory security or privacy training programme for personnel.

Control personnel-no-formal-screening-program. Status: Not implemented. A commitment the operator makes contractually. No code enforces it.

Disclosure. Pallas is operated by a small team and publishes no background-screening programme, no recurring security-awareness curriculum, and no annual training record. Personnel questions on a vendor questionnaire are answered as not formalised rather than answered yes from intent. What limits operator access instead is technical and is inventoried separately: OAuth-only sign-in with no password to share or leak, least-privilege service roles, and a tamper-evident audit trail over administrative actions that the operator cannot silently alter.

Verification. marketing: src/pages/trust.astro, Personnel screening and security training, status Not yet formalized.

Application security

The public scan tool fetches a URL the visitor supplies. That is the one place where Pallas makes an outbound request to an address a stranger chose, so it carries the guard, the response cap, the sandboxed rendering origin, and the one limitation this document cannot close.

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.

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

Control appsec-dns-rebinding-limitation. Status: Partial. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Disclosure. 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.

Implementation. app: workers/pallas-cors-proxy/src/index.ts:56; marketing: src/pages/trust.astro:388, Known limitations section.

Verification. app: docs/OPERATOR_ACTIONS.md section 1.1.

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.

The build command itself runs every gate: a type check, the typography and published-document freshness checks, the six written-claim verification gates, the unit test suite over the security and validation code, the build, an automated accessibility test against every built page, a Turnstile configuration check against the built artifact, and a sweep of the built output for forbidden claims. The hosting platform runs that command on every push, so a non-zero exit at any step fails the build and nothing deploys; the failing guards remove the dist directory so a failed build leaves nothing to deploy.

> Control appsec-release-gate-fails-closed. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.
>
> Implementation. marketing: package.json, the build script.
>
> Verification. marketing: scripts/test-a11y.mjs, scripts/test-turnstile.mjs, scripts/check-dashes.mjs, scripts/verify-doc-artifact-freshness.mjs.

Plan enforcement

A plan limit that lives only in the browser is a suggestion. Every limit Pallas sells is enforced in the database: asset, scan, and findings caps and seat caps inside Row-Level Security policies, feature access inside the policy that admits a report and the function that exports the audit log, and, since migration 049 on 2026-09-16, bulk actions and non-default retention windows. Those last two were browser-only until that date, while the pricing page sold them as tier features; the row below says so, because a reviewer who finds the gap in the migration history should find it here first.

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.

Per-plan asset and scan caps are enforced in the database, inside the Row-Level Security INSERT policies.

Control plan-asset-and-scan-caps-rls. 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. Migration 045 folds these two caps into the RLS INSERT policies, and its own header states plainly that whether the LIVE database carried limit-bearing policies before it was applied was not confirmed when the file was written. Migration 013 asserted these policies already called the limit functions; a search of the file set found no policy body containing either function name, so 013 was evidently written against a live database that had drifted from the files. Until the operator runs the section 0 pre-check and confirms migration 045 is applied on live, the honest statement to a buyer is that asset and scan caps are enforced in the database by the current migration set, with live application pending operator confirmation. The 2026-08-30 inventory record describes these caps as verified in RLS; that record and migration 045 do not agree, and this row follows migration 045 because it is the more recent and more specific document.

Implementation. app: pallas_assets INSERT policy migration 045:168, pallas_scans INSERT policy migration 045:186; app: pallas_check_asset_limit 017:188, pallas_check_scan_limit 017:246.

Verification. app: supabase/migrations/045_asset_and_scan_cap_policy_alignment.sql section 0 pre-check.

Seat caps are enforced server side, both in the Row-Level Security policy that admits a membership row and in the function that accepts a pending invitation.

Control plan-seat-caps-server-enforced. 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. The 2026-08-30 inventory record lists seat caps as client-side only. That is contradicted by the file set: the cap appears in an RLS INSERT policy at migration 034:354 and in the invitation acceptance path at 010:86 and 029:218. This row follows the file set. The inventory record understated the control rather than overstating it, but a control inventory that repeated the understatement would still be inaccurate.

Implementation. app: pallas_organization_users INSERT policy, migration 034:354; app: pallas_org_at_member_limit 010:47 rebuilt 026:146, called from pallas_accept_pending_invites at 010:86 and 029:218; app: pallas_org_invites_today, migration 026:212.

Verification. app: supabase/migrations/026_invitation_seat_accounting.sql.

Per-plan feature access is enforced server side, in the Row-Level Security policy governing report creation and in the audit export function.

Control plan-feature-flags-server-enforced. 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. The 2026-08-30 inventory record lists feature flags as client-side only. That is contradicted by the file set at migration 035:296 and 035:380. This row follows the file set. As with seat caps, the inventory record understated rather than overstated the control.

Implementation. app: pallas_reports INSERT policy, migration 035:296; app: pallas_check_feature_access 035:207, also called from pallas_export_audit_events at 035:380 for the 'audit_export' feature.

Verification. app: src/config/plans.ts:12.

Bulk actions (Growth and above) and non-default retention windows (Enterprise and above) are enforced in the database. Bulk updates run through a server-side function that refuses below the plan, and the retention-policy write policies refuse a non-default window below the plan.

Control plan-bulk-actions-and-custom-retention-server-enforced. 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 migration 049 (2026-09-16) both of these were enforced in the browser only: the bulk-actions page had no plan check and issued single-row updates the database could not tell from one-at-a-time edits, and the retention settings page let any organisation administrator on any plan write any window. The site sold both as tier features throughout. The database is now the enforcement point and the browser only hides the controls below the plan; a questionnaire asking whether plan limits are server-enforced can be answered yes for every limit the inventory lists.

Implementation. app: pallas_bulk_update_findings, migration 049:151; app: pallas_check_feature_access, migration 035:207, extended 048 and 049:88; app: pallas_retention_policies INSERT and UPDATE policies, migration 049:273 and 049:288.

Verification. app: tests/db/planGates.rls.test.ts.

Plan price strings displayed in the application are literals maintained by hand, separate from the Stripe price identifiers that determine what a buyer is charged.

Control plan-price-display-drift. Status: Partial. Backed by a file in the pallas-app repository, verified read-only on 2026-08-30, not gated from the marketing repository.

Disclosure. No automated check in either repository can catch drift between a displayed price and the Stripe price it corresponds to, because asserting they agree requires a live Stripe call that nothing here makes. Changing a price in the Stripe dashboard therefore requires editing src/config/plans.ts in the same batch. This is a documented operational constraint, not an enforced control, and it is listed here so that no generated artifact claims price integrity as a technical control.

Implementation. app: src/config/plans.ts, finding A-12 comment block.

Verification. app: Grep src/config/plans.ts for the currency symbol and reconcile all seven literals.

Data subject rights

Pallas distinguishes 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 database functions, not one filtered by the other.

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.

Retention and deletion

Retention settings cause actual deletion rather than being stored as a preference. Three of the five rows below carry a live-unverifiable basis for the same underlying reason: the scheduled jobs are created 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. The operator check that resolves it is named in each disclosure.

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.

Data classes and processing location

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.

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.

Subprocessors

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.

Vulnerability management

Three of the five rows in this category are statements about what Pallas has not done. They are in the inventory, and therefore in this document, for the same reason the implemented rows are: a buyer's risk assessment is wrong if it assumes a control that is absent.

Dependency advisories are triaged by whether a request path actually reaches the affected code, and the triage is recorded in version control with a review date and the reasoning rather than the conclusion alone.

Control vuln-advisory-triage-artifact. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. Three advisories are currently open in npm audit, reported as one low and two high. All ten individual advisories were assessed as unreachable in this build configuration, on the basis that the site is a fully static build with no server runtime, no client hydration, and no build-time image processing. The advisories remain open rather than fixed: the proposed remediation is a two major version Astro upgrade that npm audit labels breaking, and that upgrade is scheduled on its own timeline rather than taken as an emergency for unreachable findings. A buyer running npm audit against this repository will see the counts, and this file is the answer to what they mean.

Implementation. marketing: SECURITY_ADVISORY_TRIAGE.md.

Verification. marketing: SECURITY_ADVISORY_TRIAGE.md, section titled What would invalidate this triage.

Pallas has not commissioned an independent penetration test.

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

Disclosure. No third party penetration test has been performed and no report exists. Security review to date has been internal, through repeated structured audits recorded in this repository. Any questionnaire field asking for the date of the last penetration test or for a report must be answered as none performed. This is a real gap for institutional buyers and it should be presented as one rather than deflected to the internal audit history.

Verification. marketing: src/pages/trust.astro, questionnaire answers section.

Pallas holds no SOC 2, ISO 27001, or other independent security certification.

Control vuln-no-independent-certification. Status: Not implemented. A commitment the operator makes contractually. No code enforces it.

Disclosure. Pallas holds no certification of its own. Certifications named anywhere in Pallas materials belong to

subprocessors. Where a buyer requires a converted assurance rather than an assertion, the mechanism is a contractual term in the Institutional Terms of Service, the signable agreement, not a certificate Pallas can produce. There is no separate master services agreement.

Verification. marketing: src/pages/trust.astro, Compliance frameworks, Planned.

Every change reaches production through version control, and the build fails closed rather than degrading.

Control vuln-version-control-and-review. Status: Implemented. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. Pallas is built and operated by a single operator, so there is no second-party code review before merge and no segregation of duties between the person who writes a change and the person who deploys it. The compensating control is the automated build chain, which the hosting platform runs on every push, is identical for every change, and cannot be selectively skipped. Any questionnaire asking about mandatory peer review or segregation of duties must be answered no.

Implementation. marketing: package.json build script; marketing: scripts/csp-hashes.mjs and scripts/test-turnstile.mjs.

Verification. marketing: src/pages/trust.astro:121.

Deployment to Cloudflare Pages and to every Worker is a deliberate manual operator step, not an automatic consequence of a merge.

Control vuln-deploys-are-manual. 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. Manual deployment is a deliberate choice rather than a missing pipeline: it means a malformed build cannot ship silently. It also means deployment timing depends on operator availability, which a buyer assessing change velocity or emergency patch turnaround should know.

Implementation. app: docs/OPERATOR_ACTIONS.md section 2, standing operator steps.

Verification. app: docs/OPERATOR_ACTIONS.md, items 1 through 4.

Incident response and monitoring

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.

Backup and recovery

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.

Marketing subscriber management

This category covers the marketing list rather than the product, and it is included because the same reviewer usually assesses both. Two of its rows carry an app-repository basis for a reason worth naming plainly: the subscriber tables were created by hand on the live database and their definition reached version control afterwards, as migration 046 in pallas-app, which is a transcription of live rather than the file live was built

from. The disclosures say so and name the verification query that closes the gap.

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.

Scanning engine capability and its limits

What the scanner does is a security question as well as a product question, because a buyer who overestimates the coverage will under-invest in the manual review that closes the gap.

The website scanner analyses a single page of static HTML.

Control scan-single-page-static-only. Status: Partial. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. The scanner fetches one page, strips scripts, and analyses the resulting static markup. It does not crawl, it does not execute JavaScript, and single-page applications that render their content client side will not be represented in the results. A buyer evaluating Pallas against a client-rendered application should understand that the public scan tool will report on the served shell rather than the rendered page.

Implementation. marketing: src/lib/scan-report.ts; app: pallas-cors-proxy and pallas-scanner-sandbox.

Verification. marketing: tests/scan-report.test.ts.

Pallas reports against the WCAG 2.2 success criteria catalogue. Automated detection covers approximately 17 criteria; the remainder require manual review.

Control scan-wcag-coverage. Status: Partial. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. Listing the full WCAG 2.2 catalogue is not the same as detecting all of it. Roughly 17 success criteria are reliably detectable by automated tooling; the rest depend on human judgement about context, purpose, or

equivalence and cannot be settled by a rule engine. Any procurement document that presents the catalogue must state the automated subset in the same sentence, not in a footnote.

Implementation. marketing: src/data/stats.ts; marketing: package.json, axe-core dependency.

Verification. marketing: src/pages/accessibility.astro and src/pages/features.astro.

Section 508 and EN 301 549 results are produced by applying the corresponding axe-core tag presets, not by an independent rule engine for each standard.

Control scan-508-and-en301549-are-tag-presets. Status: Partial. Demonstrated in the pallas-marketing repository and checked by a release gate.

Disclosure. Section 508 and EN 301 549 coverage in Pallas is the axe-core tag preset for each standard. Pallas has not implemented, and does not claim, an independent rule engine per standard. A buyer procuring against EN 301 549 specifically should read this as WCAG-derived automated coverage mapped to that standard, and should not treat it as a full EN 301 549 conformance assessment, which includes non-web requirements Pallas does not evaluate at all.

Implementation. marketing: package.json, axe-core dependency, and the tag selection in the scan path.

Verification. marketing: tests/scan-report.test.ts.

Pallas publishes a VPAT 2.5 Accessibility Conformance Report for its own product.

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

Disclosure. PDF content is verifiable in this repository and gated on every release. PDF tagging is not verifiable here and is confirmed by the operator running PAC 2024 against the regenerated file. The build gate can prove the words are right; it cannot prove the tag tree is.

Implementation. marketing: public/documents/pallas-vpat-2.5-acr.pdf.

Verification. marketing: scripts/verify-doc-artifact-freshness.mjs, check 10; marketing: Validate the regenerated PDF with PAC 2024, target zero PDF/UA and zero WCAG failures.

Pallas performs no artificial intelligence or machine learning processing on any data. Accessibility findings are produced by deterministic rule evaluation in axe-core, and no model, inference service, or AI provider is present in either repository.

Control ai-no-model-processing. 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. The absence of an AI dependency is demonstrable in both repositories by inspection of the dependency sets, and that is the strongest form this claim takes. It is a statement about what is installed, not a runtime proof that no network call ever reaches a third-party model, which no dependency list can establish. A buyer whose control requires attested absence of model processing should ask for it contractually; the answers to the AI worksheet rest on this row and on nothing stronger than it.

Implementation. marketing: package.json dependency set: astro and axe-core at runtime, with no model runtime, inference client, or AI provider SDK; app: package.json dependency set: @supabase/supabase-js, axe-core, pdf-lib, jszip, react and react-router-dom, with no model runtime, inference client, or AI provider SDK.

> Verification. marketing: scripts/test-a11y.mjs.

Known limitations

This section exists because a residual risk that has been accepted should be readable by the buyer evaluating Pallas, not only by the operator who accepted it. Each item below renders its disclosure text verbatim from the control inventory, so the wording here cannot drift from the wording published at pallas.lonia.ai/trust.

DNS rebinding against the scan proxy

This is finding A-15, accepted by the operator on 30 August 2026 at informational severity on the recommendation of the pre-launch re-audit, which named no action before launch. A Cloudflare Worker resolves DNS inside its own fetch call and the runtime exposes no hook to read the resolved address and no way to pin the connection to an already-validated one, so there is no point at which the Worker can compare the address it approved against the address it connected to.

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 partial mitigation considered and not taken was pre-resolving the hostname over DNS-over-HTTPS and rejecting private answers. It is a small change and it does not close the finding, because it is time-of-check to time-of-use: an attacker controlling the zone flips the answer between the pre-resolution and the Worker's own. It would raise the attack cost from trivial to requiring a race, and in exchange it would add a DNS round trip to every scan and a new failure mode when the DNS-over-HTTPS endpoint is slow. The finding will be revisited if the stack gains an egress proxy or a private network, if the proxy is ever given access to a private network or to credentials, if Cloudflare's runtime gains address pinning or a resolved-address hook, or if the proxy's role widens beyond fetching public third-party HTML for scanning.

Single-page static-HTML scanning scope

The website scanner analyses a single page of static HTML.

> Control scan-single-page-static-only. The scanner fetches one page, strips scripts, and analyses the resulting static markup. It does not crawl, it does not execute JavaScript, and single-page applications that render their content client side will not be represented in the results. A buyer evaluating Pallas against a client-rendered application should understand that the public scan tool will report on the served shell rather than the rendered page.

Automated detection covers roughly 17 success criteria

Pallas reports against the WCAG 2.2 success criteria catalogue. Automated detection covers approximately 17 criteria; the remainder require manual review.

> Control scan-wcag-coverage. Listing the full WCAG 2.2 catalogue is not the same as detecting all of it. Roughly 17 success criteria are reliably detectable by automated tooling; the rest depend on human judgement about context, purpose, or equivalence and cannot be settled by a rule engine. Any procurement document that presents the catalogue must state the automated subset in the same sentence, not in a footnote.

Section 508 and EN 301 549 results are tag presets

Section 508 and EN 301 549 results are produced by applying the corresponding axe-core tag presets, not by an independent rule engine for each standard.

> Control scan-508-and-en301549-are-tag-presets. Section 508 and EN 301 549 coverage in Pallas is the axe-core tag preset for each standard. Pallas has not implemented, and does not claim, an independent rule engine per standard. A buyer procuring against EN 301 549 specifically should read this as WCAG-derived automated coverage mapped to that standard, and should not treat it as a full EN 301 549 conformance assessment, which includes non-web requirements Pallas does not evaluate at all.

No independent penetration test

Pallas has not commissioned an independent penetration test, and no report exists. Security review to date has been internal: repeated structured audits of the code, recorded in the repositories, with their findings and fixes; dependency advisory triage, recorded per advisory with its reachability reasoning in an internal record (last triaged 17 September 2026, issued on request); and the release gates described above. A questionnaire field asking for the date of the last penetration test, or for the report, is answered "none performed". This is a real gap for an institutional buyer and it is presented as one, not deflected to the audit history. No date is promised here; when a test is commissioned, this section changes and the report is issued on request.

Pallas has not commissioned an independent penetration test.

> Control vuln-no-penetration-test. No third party penetration test has been performed and no report exists. Security review to date has been internal, through repeated structured audits recorded in this repository. Any questionnaire field asking for the date of the last penetration test or for a report must be answered as none performed. This is a real gap for institutional buyers and it should be presented as one rather than deflected to the internal audit history.

No SOC 2, ISO 27001, or other independent certification

Pallas holds no SOC 2, ISO 27001, or other independent security certification.

> Control vuln-no-independent-certification. Pallas holds no certification of its own. Certifications named anywhere in Pallas materials belong to subprocessors. Where a buyer requires a converted assurance rather than an assertion, the mechanism is a contractual term in the Institutional Terms of Service, the signable agreement, not a certificate Pallas can produce. There is no separate master services agreement.

No published recovery time or recovery point objective

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

Control backup-no-published-rto-rpo. 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.

No documented restore test

Pallas has not performed a documented restore test.

Control backup-no-restore-test. 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.

No security operations centre, and one operator

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

Control incident-detection-monitoring. 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.

Every change reaches production through version control, and the build fails closed rather than degrading.

Control vuln-version-control-and-review. Pallas is built and operated by a single operator, so there is no second-party code review before merge and no segregation of duties between the person who writes a change and the person who deploys it. The compensating control is the automated build chain, which the hosting platform runs on every push, is identical for every change, and cannot be selectively skipped. Any questionnaire asking about mandatory peer review or segregation of duties must be answered no.

Processing location

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

Control data-location-eu-option. 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.

What a buyer should do with this document

A reviewer who needs more than an assertion has four routes, in increasing order of effort.

1. Read the disclosures rather than the claims. Every row whose status is partial or not implemented, and every row whose basis is live unverifiable, carries the exact wording that qualifies it. Those are the rows a risk assessment turns on.
2. Request the subprocessor reports. Where a control rests on a vendor assertion, the evidence a buyer actually wants is that vendor's SOC 2 Type II report, which the operator holds under NDA and can route a request for. A Pallas assertion is not a substitute and is not offered as one.
3. Ask for the operator pre-checks. Every live-unverifiable row names a specific query or dashboard check that resolves it. The operator will run them and return the output for a specific onboarding review.
4. Contract for what the controls do not cover. A negotiated recovery time objective with service credits and an end-to-end breach notification window that does not depend on subprocessor performance are not deployed capabilities, and each says so in its own row above.

Questions about anything in this document 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; the executed version of the Data Processing Agreement is issued on request once terms are agreed.

Appendix A. Controls cited

This document cites 82 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.
- **plan-asset-and-scan-caps-rls** (Plan and quota enforcement). Partial.
- **plan-seat-caps-server-enforced** (Plan and quota enforcement). Implemented.
- **plan-feature-flags-server-enforced** (Plan and quota enforcement). Implemented.
- **plan-bulk-actions-and-custom-retention-server-enforced** (Plan and quota enforcement). Implemented.
- **plan-price-display-drift** (Plan and quota enforcement). Partial.
- **encryption-at-rest** (Encryption). Implemented.
- **encryption-in-transit** (Encryption). Implemented.
- **encryption-no-secrets-in-repo** (Encryption). Implemented.
- **encryption-member-prose-provider-held-key** (Encryption). Implemented.
- **access-org-roles** (Access control). Implemented.
- **access-last-admin-guard** (Access control). Implemented.
- **access-privilege-loss-revokes-invites** (Access control). Implemented.
- **access-rate-limiting-authenticated** (Access control). Implemented.
- **access-rate-limiting-anonymous** (Access control). Partial.
- **access-turnstile-fails-closed** (Access control). Implemented.
- **personnel-no-formal-screening-program** (Access control). Not 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.
- **appsec-release-gate-fails-closed** (Application security). Implemented.
- **scan-single-page-static-only** (Scanning engine capability). Partial.
- **scan-wcag-coverage** (Scanning engine capability). Partial.
- **scan-508-and-en301549-are-tag-presets** (Scanning engine capability). Partial.
- **scan-vpat-published** (Scanning engine capability). Implemented.
- **ai-no-model-processing** (Scanning engine capability). 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.
- **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.
- **vuln-advisory-triage-artifact** (Vulnerability management). Implemented.
- **vuln-no-penetration-test** (Vulnerability management). Not implemented.
- **vuln-no-independent-certification** (Vulnerability management). Not implemented.
- **vuln-version-control-and-review** (Vulnerability management). Implemented.
- **vuln-deploys-are-manual** (Vulnerability management). Implemented.