SECURITY & TRUST WHITEPAPER

How Behavry secures the system of record for autonomous behavior.

A technical reference for security reviewers, CISO teams, GRC and compliance functions, and technical evaluators conducting due diligence on Behavry.

Version 1.0

Published July 2026

Reviewed quarterly and after any material control change

Contact security@behavry.ai

CONTENTS

What's in this document.

> **How to read this document:** every section distinguishes what is shipped today from what is on our roadmap. Where a control is partial, we say so and name what remains. We do not display a "certified" badge for any framework unless an accredited third party has issued that certification.

01 Executive Summary

Behavry is the independent system of record for autonomous agent behavior. It sits inline across the platforms your agents already use, reconstructs a signed account of each agent's actions, and makes that account independently verifiable — including by parties who do not trust Behavry.

The premise is simple: **the entity that acts cannot credibly attest to its own behavior.** An agent's own logs, and the logs of the vendor whose product the agent runs on, are not independent evidence of what happened — they are the actor's own account. Behavry is architecturally separate from the agents and vendors it observes, so it can produce a record neither of them controls.

THE OPEN-VERIFIER PROMISE

You can verify any Behavry audit record offline, against a public key you pin yourself, with no Behavry service in the loop. Every event is SHA-256 hash-chained and Ed25519-signed; export bundles carry a Merkle root; a standalone verifier script checks all of it without calling our API or reading our database. Section 3 explains exactly how.

This document is organized the way we expect a security review to actually proceed: architecture, then the specific integrity model behind our central claim, then data handling, tenant isolation, our own application-security track record, compliance mappings, deployment options, reliability, vulnerability response, sub-processors, and how to get more. Throughout, we distinguish **shipped and verifiable** controls from **roadmap** items — a distinction that matters more to a careful reviewer than a longer list of claims.

02 Product & Architecture Overview

Behavry's backend is deliberately unglamorous: a single FastAPI application with two supporting services — Open Policy Agent (OPA) for policy evaluation and a TimescaleDB/PostgreSQL audit store — rather than a sprawling microservices mesh. Internal modules communicate via direct function calls, not through an internal network of services to secure. This keeps the trusted computing base small and auditable, at some cost to the "distributed systems" story a larger architecture diagram might imply.

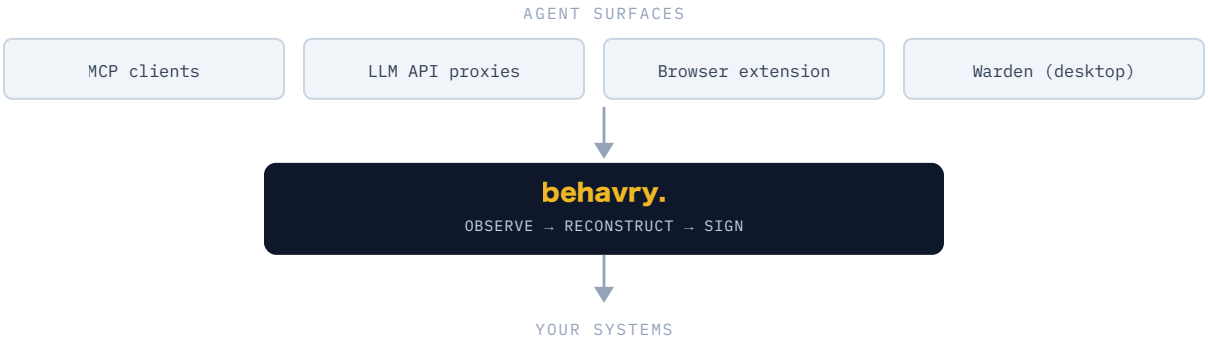

Figure 1 — conceptual request path: Agent surfaces (MCP clients, LLM API proxies, browser extension, Warden desktop agent) → Behavry (observe, reconstruct, sign) → your systems (files, CRM, code, production). Not an infrastructure diagram; no internal network topology is shown or implied.

Four ways agents connect:

- **MCP Proxy** — the ingress path for MCP-speaking agents, over JSON-RPC 2.0 / Streamable HTTP. Every call undergoes JWT authentication, a DLP scan, an OPA policy check, and enforcement, in that order, before reaching its target.
- **LLM API proxies** — OpenAI- and Anthropic-compatible endpoints that gate direct model access behind the same identity and policy layer used for tool calls.
- **Browser extension** — a Manifest V3 extension installed by customer IT, scoped to a declared list of AI web domains and authenticating via a short-lived extension token (90-day expiry). It cannot read or modify content on any site outside that list.
- **Warden** — a Go-based desktop proxy agent for endpoints not otherwise covered. It is maintained in its own public repository so customers can audit the proxy code running on their machines. It enrolls per device via a signed token and a per-device certificate.

Multi-tenant from the ground up. Every tenant-scoped table includes a tenant identifier, enforced first at the application layer and backed by PostgreSQL row-level security as a defense-in-depth measure (Section 5).

> **Fail-closed by design.** If the policy engine (OPA) becomes unreachable, every tool call is denied — not allowed. This is verified by dedicated automated tests for both timeout and connection-error conditions, not asserted only in documentation.

03 The Decision Trace Integrity Model

Every action Behavry observes becomes an audit event. Three independent mechanisms protect the resulting record:

- **Hash chain.** Each event's hash incorporates the hash of the immediately preceding event for that agent: `event_hash = SHA-256(canonical_event_data || previous_hash)`. Editing any past

event invalidates every hash computed after it — the tampering is structurally detectable, not merely policy-forbidden.

- **Per-event signatures.** Each event is signed with Ed25519 at write time. The private key never leaves the signing service and is never logged. Production deployments must explicitly provision this key — Behavry will not silently auto-generate or rotate it, because rotating the key would break every customer's ability to verify prior history.
- **Merkle-rooted exports.** Bulk exports — audit bundles, compliance-framework exports, and evidentiary provenance packages — include a manifest containing a Merkle root computed over the export's events, plus a manifest-level signature.

THE OPEN VERIFIER

`tools/verify_event.py` is a standalone script that checks a manifest signature, recomputes the Merkle root, walks the hash chain, and verifies every per-event signature — entirely offline, against a public key the customer pins themselves. It requires no Behavry API call, no access to our database, and no live key-management service. Running it is the fastest way to confirm we are telling the truth about the record's integrity, because it does not require trusting us to check our own work.

Signing-key management. The per-event Ed25519 signer described above is one layer. A separate, higher-level signer secures aggregate provenance records and is built around a pluggable key-management interface: **AWS KMS is fully wired and functional today.** Azure Key Vault and GCP KMS share the same interface and are architecturally complete, but currently fall back safely to the internal Ed25519 signer rather than calling a live cloud KMS — we will update this document when that changes, rather than describe all three as equivalent today.

> **Scope of the integrity guarantee.** The tamper-evidence described in this section applies to the audit-event stream Behavry produces from what it directly observes — actions at the tool-call, filesystem, and database layers today. Reconstructing a single, unified cryptographic proof across every third-party vendor surface an agent might touch is the direction of the architecture, not a shipped, end-to-end guarantee across every vendor combination as of this writing.

04 Data Handling & Privacy

Behavry is designed as a policy-enforcement checkpoint, not a data warehouse. By design, it stores only the minimum information needed to enforce policy and produce compliance evidence — not the content of agent interactions.

COMPONENT	STORED	NEVER STORED
MCP Proxy	Tool name, action, resource path, SHA-256 hash of input	Full tool input/output, file contents, database results
Audit trail	Agent ID, action, policy result, DLP pattern + location, hash chain	Message content, prompts, completions, SQL data
DLP scanner	Pattern matched, severity, redacted sample (e.g. `sk3***xyz`)	The full secret value
Anthropic proxy	Model, token counts, `has_system_prompt` (bool), finish reason	System instructions, prompts, completions
OpenAI proxy	Model, token counts, `has_tools` (bool), finish reason	Messages, function arguments, tool results

Input hashing. The `input_hash` field stores SHA-256(input) — not the input itself. The original cannot be recovered from the hash; it exists to prove an action occurred and to detect changes without revealing what the action contained.

Encryption. TLS is used at every network hop in production. Stored payloads use AES-256-GCM envelope encryption, keyed either by a customer-supplied local key or, in AWS deployments, by AWS KMS — Azure and GCP equivalents are not yet implemented for data-at-rest envelope encryption.

Retention and erasure. Retention windows are customer-configurable. Because the audit log is an append-only, hash-chained ledger, an erasure request (for example, under GDPR Article 17) is honored by anonymizing the natural-person identifiers and payload fields in place — rather than physically deleting the row — so the hash chain's tamper-evidence remains intact through the erasure. Rows are stamped as purged; opaque agent and session identifiers are retained.

The practical implication. A compromised Behavry datastore exposes governance metadata — who did what and whether it was allowed — but not the underlying prompts, files, or query results, which are never captured in the first place.

05 Tenant Isolation & Access Control

Two independent layers enforce tenant boundaries. The **application layer** routes every tenant-scoped query through a tenant-aware access helper, introduced specifically to prevent the "unscoped lookup" pattern that produced real findings during security testing (Section 6) from recurring. The **database layer** enforces PostgreSQL row-level security — with `FORCE ROW LEVEL SECURITY` enabled so even the table owner is subject to policy — on every tenant-scoped table. Coverage was built incrementally and then closed systematically: a later migration enumerated every table with a tenant identifier column and compared it against live database policies to confirm nothing was missed.

> We treat row-level security as a **backstop**, not the primary boundary. The primary boundary is the application code; row-level security exists so that a mistake in the application code fails safe — denying cross-tenant access — rather than silently leaking data across tenants.

Identity & authentication.

- **Agents** authenticate via the OAuth 2.1 client-credentials flow. A registered agent receives a `client_id` and `client_secret` — the secret is shown only once, at registration, and stored as a bcrypt hash thereafter. Every API call includes a short-lived, RS256-signed JWT that is re-validated against a live session record on each request, so revoking a session takes effect immediately rather than waiting for token expiry.
- **Human administrators** authenticate via password login, Clerk-managed OIDC, a generic OIDC connection to an enterprise identity provider (Microsoft Entra ID, Okta, Ping), or SAML 2.0 SSO, with validation performed by a dedicated XML-signature verification library. SSO today is service-provider-initiated; identity-provider-initiated login is on our roadmap.
- Admin and agent tokens are signed with **separate keys**, and JWT verification is hard-restricted to RS256 — there is no algorithm-confusion fallback path.

Session controls. Administrators can revoke individual sessions, and repeated failed admin login attempts trigger a temporary lockout. Multi-factor authentication enforcement and SCIM-based auto-provisioning are represented in our data model today but are not yet exposed in the product — we are not claiming either as shipped.

06 Application Security Program

Behavry's own codebase and public-facing infrastructure are subject to recurring third-party security testing, in addition to internal review. To date:

- An initial static-analysis security audit identified a set of critical and high-severity findings, including an authentication-bypass issue and several cross-tenant data-access issues. All findings were remediated before the next testing cycle.
- A subsequent white-box penetration test, aligned with the OWASP Web Security Testing Guide and covering the production application and public web properties, identified nine findings ranging from critical to medium severity — including additional cross-tenant access issues in newer features, a SAML signature-validation weakness, and two server-side request forgery findings. Each finding was remediated and independently retested, with the specific fix commit and regression test recorded for each finding.
- Remediation work also produced defense-in-depth improvements that outlast the specific findings that prompted them: a dedicated tenant-scoped data-access helper and expanded row-level security coverage described in Section 5.

> We treat "no known open findings from our most recent engagement" as the honest bar — not "we have never had a vulnerability." The audit history above, including the findings we didn't enjoy uncovering, is part of why we're comfortable publishing it.

Platform hardening. Production traffic enforces a restrictive Content-Security-Policy, HTTP Strict-Transport-Security, explicit non-wildcard CORS origins, and standard protections against clickjacking and MIME-sniffing — all covered by an automated test suite that fails the build if any of these regress.

07 Compliance Posture & Framework Mappings

We display a "certified" badge only when it is literally true. Below is the current state of our compliance program as of this writing.

SOC 2	CONTROL MAPPING COMPLETE	CC6–CC8 (Security) mapped, with supporting CC9 (Availability) controls. An independent Type II audit is on our roadmap and has not yet been completed. We are not SOC 2 certified today.
ISO/IEC 27001:2022	MAPPED	Controls mapped to A.5.24, A.8.2, A.8.15, A.8.16, and A.8.24. Certification has not been pursued.
Framework crosswalks	LIVE	FSI (OCC SR 11-7 · NYDFS 23 NYCRR Part 500 · SEC Rules 206(4)-7, 204-2 · Regulation S-P) · OWASP Agentic Security Initiative · HITRUST AI · CIS AI/ML · NIST AI Risk Management Framework · PCI DSS v4.0 · EU AI Act · GDPR · HIPAA.

Where coverage is partial, we say so. For example, under PCI DSS v4.0: formal vulnerability-management programs, PCI-qualified penetration testing, and ASV scanning are organizational functions performed by the customer's own security program — Behavry contributes continuous behavioral testing and posture evidence, not a replacement for that program. The same disclosure pattern applies to human-oversight contestability workflows under the NIST AI RMF, which are organizational governance functions we provide the technical audit trail for, not a function we perform ourselves.

Evidence is exportable in PDF, CSV, and JSON formats, keyed to specific controls and API endpoints, enabling an auditor to trace a claim back to its source. `Compliance mappings were last reviewed in April 2026.`

08 Deployment & Data-Residency Options

Behavry supports several deployment topologies so that data-residency requirements can be met without changing how the product is governed:

MODE	WHAT IT IS
Managed SaaS	Behavry operates the control plane. Fastest path to deployment.
Self-hosted — Docker Compose	The full stack (application, OPA, database) inside your own infrastructure, with a documented quick start and environment-variable reference.
Self-hosted — Kubernetes (Helm)	A maintained Helm chart with a zero-downtime rolling-update strategy and support for an externally managed database.
Self-hosted — AWS (Terraform)	A published Terraform module that stands up the full stack, including a load-balancer endpoint, from infrastructure-as-code.
Self-hosted — Azure	Azure Container Apps with Azure Database for PostgreSQL Flexible Server (TimescaleDB extension enabled), delivered via CLI tooling and a pre-filled onboarding configuration.
Hybrid	Behavry hosts the control plane; your data plane runs inside your own VPC. If connectivity to the control plane is lost, existing agents keep working uninterrupted for a 7-day grace period while new enrollments pause — a transient network issue does not become an outage.
Warden (desktop)	A lightweight Go proxy for endpoints, enrolled per device, maintained in its own public repository so customers can review the code running on their machines.

> In every self-hosted and hybrid mode, you are required to block direct agent-to-provider network paths at the infrastructure layer (security group, DNS, or firewall). Without that control in place, an agent could bypass the proxy entirely. This is a configuration you own — it is not something Behavry enforces remotely.

Supply-chain integrity. Container images are signed with cosign using keyless, OIDC-based signing and accompanied by a software bill of materials generated at build time. Signatures are verifiable against a public transparency log.

09 Reliability & Continuity

Behavry's own hosted control plane maintains an active/passive disaster-recovery posture across two AWS regions, with cross-region database backups and storage replication, a documented two-person-rule failover runbook, and a recurring drill that measures actual recovery time against internal targets — a recovery point objective of one hour or less and a recovery time objective of fifteen minutes or less. Our production database runs on a managed Postgres/TimescaleDB service, which provides vendor-managed backup, high availability, and point-in-time recovery as part of that managed service.

> **For self-hosted and BYOC deployments**, backup and restore are the customer's responsibility, as with any self-hosted software. We document standard tooling — `pg_dump` for point-in-time snapshots and pointers to continuous-backup tools such as Barman or pgBackRest for production-grade continuous protection — but we do not operate or test your backup pipeline. If your deployment has specific recovery-point or recovery-time objectives, plan and test them independently of the infrastructure practices described above, which apply only to Behavry's hosted environment.

Live system status, incidents, and maintenance windows are available at `status.behavry.ai`.

10 Vulnerability Management & Security Response

Behavry does not yet operate a large in-house security-operations function, and we would rather state this plainly than imply otherwise. What we do have:

- A recurring, independent third-party testing cadence (Section 6), with every finding tracked to a specific fix and re-verified.
- A public responsible-disclosure channel — `security@behavry.ai` and `behavry.ai/.well-known/security.txt` — with a committed two-business-day acknowledgment window and a safe-harbor commitment for good-faith research.
- Product-level alerting and webhook escalation (Slack, PagerDuty, or a custom SIEM endpoint) enable customers to route Behavry's findings — anomalies, policy violations, DLP triggers — into their existing incident-response process.

We consider a formal vulnerability-management program, in the sense a PCI Qualified Security Assessor or a large enterprise security team would recognize, to be a maturity milestone ahead of us rather than behind us.

11 Sub-processors & Shared Responsibility

VENDOR	WHAT IT TOUCHES
AWS	Cloud infrastructure & hosting
Google Cloud	Infrastructure & workspace services
Microsoft	Enterprise SSO integration (Entra ID / OIDC), email, and cloud infrastructure & hosting
DigitalOcean	Infrastructure & hosting
Clerk	Admin identity & authentication (OIDC)
Anthropic	Claude model API — metadata-only proxy, no prompt/completion content
OpenAI	GPT model API — metadata-only proxy, no prompt/completion content
Timescale (TimescaleDB)	Audit log database

Shared responsibility, in brief. Behavry is responsible for the security of the platform we operate — the SaaS control plane, our own infrastructure, and the code we ship. If you run a self-hosted or BYOC deployment, you are responsible for the infrastructure you run it on: patching the host operating system, maintaining your own backups, and enforcing the network boundary described in Section 8. A full sub-processor list with data-flow detail is available under NDA.

12 How to Request More

This document is deliberately conservative in what it claims. If your review needs more than what is written here, we would rather hand it to you directly than have you infer it.

QUESTIONNAIRES

SIG-Lite and CAIQ responses are available on request.

DEEPER DETAIL

Source-available review, live-environment walkthroughs, and penetration-test reports are available under NDA.

DOCUMENTATION

API reference and integration guides at `docs.behavry.ai`.

Contact `security@behavry.ai`. This document is reviewed quarterly and after any material change to the controls it describes.