TRUST & ASSURANCE

How this platform is built, tested and operated

Written for the security review team, not the buyer. Everything on these pages is either independently verifiable or stated as a commitment we can be held to.

What this document is not. It is not a certification, and it does not assert that you are compliant with anything. An audit report describes controls at a point in time against a defined scope; this describes how the system works and what you can check yourself. Both are useful and they are not substitutes.

1 · How it is built

The privileged surface is deliberately small. The agent's parsers — the components handling untrusted input by definition — are memory-safe or sandboxed, and the code running with full privilege is kept as thin as we can make it. This is where a bug becomes remote code execution across an estate, so it gets the disproportionate share of review.

Distributed content is data, not code. Detection and enforcement content is interpreted by the agent rather than executed by it. A malformed or pathological rule causes rejection and a report, not a fault — and the previous known-good content stays resident, so rejection degrades to yesterday's protection rather than none.

Everything shipped is signed, and the build is reproducible. Agent binaries, content bundles and marketplace modules carry signatures the agent verifies before use. Dependencies are pinned, inventoried and reviewed on a schedule rather than on incident.

Third-party modules run under a declared capability set. Enforced by a sandbox rather than by policy: no ambient credential access, an explicit resource budget, and a listing that can be withdrawn. A badly written module degrades itself, not the pipeline.

2 · How it is tested

New content is validated against retained events before it is enabled anywhere — which answers the question no one can answer by inspection: how often would this have fired last month, and on what. It then reaches a bounded slice of the fleet in observation mode, and only afterwards enforces, in stages the distribution system enforces rather than an operator chooses.

Beyond that: continuous internal adversary simulation against a twin of a representative estate, external testing by parties we do not select for a friendly result, and a published crash and defect record for the agent. We do not claim the agent never faults — privileged software parsing hostile input on heterogeneous hosts will fault. We claim it is designed to fail without taking the host, and that the record is available rather than asserted.

> **Rollback is a measured number.** Time-to-withdraw content already on the fleet is tested on the same schedule as a backup restore. A rollback path exercised for the first time during the incident it exists for is a hypothesis, not a control — and we would rather tell you the figure than describe the capability.

3 · How it is operated

No standing access to your events. Support and engineering access is time-boxed, bound to a stated reason, and requires step-up on every cross-tenant read. Operator sign-in has no password door and no self-service recovery — one hardware credential on an attested device.

Every access is an event in your store. Not in a log we hold and describe to you — a record beside your own telemetry, queryable on the same clock. This is the single control we would most want a reviewer to test.

Isolation is keys and storage boundaries. A query missing its tenant scope returns nothing rather than another tenant's data. Only compiled findings cross a tenant boundary — reviewed rules carrying no customer-identifying content. Events never do.

Retention and erasure are per record class. Each window carries the reason it exists. An erasure request destroys that subject's key across replicas, backups and archives at once, and the erasure is itself recorded without re-recording the erased content.

4 · What you can verify without taking our word

01 Trigger a support access grant, then query it from your own event store. Confirm the reason, the scope and the expiry are all present.

02 Open an alert from a year ago and check whether its evidence is stored or re-queried on open. The behaviour is observable.

03 Feed the agent deliberately malformed content in a lab and watch what happens to the host. Then confirm the previous content is still enforcing.

04 Ask an automated action for its blast radius, then change the estate underneath it and ask again. The number should move.

05 Submit an erasure request for a test subject, then attempt to read that subject's historical records. Including from a restored backup.

06 Regenerate a published coverage figure while watching. If it takes a person and a spreadsheet, it was an opinion with a decimal point.

07 Hold non-urgent content on a subset of hosts you nominate, and confirm the hold is honoured.

08 Export your events and your rules, and confirm something other than us can read them.

5 · What we owe you when something goes wrong

We are a concentrated, high-value target and we will be attacked by people who are good at it. The commitments that matter are therefore about disclosure rather than invulnerability: notification on a contractual timeline rather than a best-effort one, scoped to what was actually reachable in your tenant; the evidence in your own store so you can verify our account against your records; and a named path to a human, not a status page.

Consolidation is part of this. Running one platform instead of six removes seams and creates a shared fate that six did not have. We think the trade is right and it should be made deliberately — the honest version of the argument is in Paper 11, including the parts that do not favour us.

6 · Open items

A control that is planned rather than in place belongs here, not in the sections above. These are the items outstanding as this version was published.

The privacy notice is not yet published

Our demo form's consent sentence refers to a notice a reader should be able to open, and that page is not live. The consent language is accurate about what we collect and do; the document backing it is outstanding. No publication date is committed, and we have left the link visible rather than removing it so the gap stays legible.

No coverage figure is published yet

The coverage sheet is a generated artefact — computed from the runtime with a build guard that fails when a published figure drifts from its source — and that generator is not finished. Until it is, no coverage percentage appears anywhere in our collateral, and the illustrative figures in Paper 08 §1 are labelled illustrative on the page.

The agent defect record has no published cadence

§2 commits to the record being available to a review team rather than asserted. It is not yet on a stated publication interval, which means today you have to ask for it. Committing to an interval is the intended end state and is not in place.

If a later version of this page shows no open items, ask for the current list before assuming there are none.

EventLake

GETTING THE DETAIL

This brief is published rather than gated, and so is everything it references. Architecture detail, test reports and the agent defect record are available to a review team on request — we collect a work email once, for that conversation, and tell you what we do with it before you submit.

Companion reading: Paper 05 *Delegation is a design problem* · Paper 06 *Evidence should be a query* · Paper 08 *A vendor who publishes no gaps* · Paper 11 *We are the most privileged software you will install*. eventlakes.com