ON PRIVILEGED AGENTS, TENANT ISOLATION AND THE UPDATE CHANNEL

We are the most privileged software you will install

A kernel-adjacent agent on every host, a copy of your telemetry, and a channel that pushes code into your estate without a maintenance window. Any honest security vendor has to be assessed as a threat, not only as a control.

IN BRIEF

Security platforms concentrate privilege by design: everywhere, deeply, with a persistent path in. That concentration is the reason they work and the reason a compromise of the vendor is a compromise of every customer at once. This paper describes the four exposures a customer inherits — the agent, the content channel, the data, and the vendor's own staff — and what each one requires of the vendor in return. It is the paper we would least like a competitor to write about us, which is the argument for writing it ourselves.

1 · Four inherited exposures

A security platform is not one risk. It is four, and they have different mitigations, different blast radii, and different people who should be asking about them.

The agent. Privileged code on every host, parsing untrusted input by definition. Its bugs are remote code execution on your entire estate, and its crashes are your outage. This is the exposure with the worst upper bound.

The content channel. The mechanism Paper 04 argues for — rapid rule distribution without a redeploy — is also a standing, authenticated path for pushing instructions into every enforcement point you own.

The data. A copy of your estate's behaviour, sitting in someone else's infrastructure — often the single richest description of your organisation that exists anywhere, including inside it.

The vendor's people. Support engineers with cross-tenant reach. The most probable of the four to be exercised, and the least often asked about in a security review.

Note the asymmetry between probability and severity. The agent has the worst consequence; the fourth is the one that happens. A review that spends its time on data residency and none on operator access has optimised for the wrong row.

FIGURE 1 · WHERE THE REQUIREMENT LANDS

`agent`	Minimal privileged surface · parsers in memory-safe code or sandboxed · **fail-open on its own fault**, never taking the host · a published crash record rather than a claim of never crashing.
`content channel`	Signed content, data not code · staged rollout the system enforces · agent rejects and reports rather than failing · previous known-good resident · rollback measured and drilled.
`data`	Per-tenant keys · isolation enforced at the storage layer, not by a query filter · encryption in transit and at rest as a floor, not a differentiator · residency stated per record class.
`vendor staff`	No standing access · time-boxed, reason-bound, customer-visible grants · every access an event *in the customer's own store* · step-up on every cross-tenant read.

The last row's phrasing matters: an access log the vendor holds is a promise. An access event in your store, queryable beside your telemetry, is a control.

2 · Tenant isolation is an architecture, not a setting

Multi-tenancy is usually described in terms of what customers cannot see of each other, which is the easy half. The interesting question is what the *platform* can do across tenants, because that is where the shared-fate risk lives.

Isolation implemented as a tenant identifier in every query is isolation that fails the first time someone writes a query without one. Isolation implemented as separate keys and separate storage boundaries fails differently: a mistake yields no data rather than the wrong tenant's data. Both designs pass a penetration test on a good day. Only one of them fails safely on a bad one.

There is a genuine tension here worth naming, because a vendor that hides it is hiding the interesting part. **Cross-tenant learning is valuable.** A technique seen in one customer should improve detection for all of them — that is much of the argument for a platform over a local tool. But the mechanism that lets a rule derived from tenant A's incident protect tenant B is a mechanism that touches both. The resolution is that

findings cross the boundary and *events* do not: what travels is a compiled rule, reviewed, carrying no customer-identifying content. It is a narrower benefit than the marketing version, and it is the only version that is defensible.

The question that separates the two designs. Ask what a query missing its tenant scope returns. "An error" is one answer; "nothing" is a better one; "we would catch that in review" means the isolation is a convention.

3 · The update channel deserves its own review

Paper 04 argued for compiling findings into enforced rules in hours. That capability and the industry's most public failures share a mechanism, and the honest framing is that **the safeguards are not a caveat on the feature — they are the feature**. A distribution system without staged exposure is a same-day outage generator with a security label.

Three properties are not negotiable. Content is data interpreted by the agent rather than code executed by it, so a pathological rule is rejected rather than run. Staged exposure is enforced by the distribution system rather than by operator intention, because the case where someone skips the stages is precisely the urgent one. And the previous known-good content stays resident, so rejection degrades to yesterday's protection rather than to none.

A customer-side control is worth asking for and rarely offered: the ability to hold non-urgent content for a stated interval, on a subset of the estate you nominate. It slows your own coverage slightly and it converts the vendor's rollout discipline from a promise into something you can verify.

4 · What we will not claim

Not that our agent will never crash. Privileged software parsing hostile input on heterogeneous hosts will fault. What we will say is that it is designed to fail without taking the host, and that the record is publishable.

Not that we cannot be compromised. We are a concentrated, high-value target and we will be attacked by people who are good at it. What is testable is whether a compromise of us is bounded per tenant, and whether you would find out from your own records.

Not that a certification answers this. An audit report describes controls at a point in time against a defined scope. It is necessary and it is not an answer to “what can your support engineer read tonight”.

Not that concentration is free. This series argues for one platform over six, and one platform is a single point of failure that six were not. The trade is real: fewer seams against a larger shared fate. It should be made deliberately, not sold as pure upside.

The unifying idea. Privilege is what makes the platform work. The only honest response is to make that privilege legible, bounded, time-limited and visible in the customer's own records — so trust rests on something checkable rather than on the vendor's reputation.

5 · Questions worth asking, whoever you are buying from

01 What happens to the host when your agent faults — and what is your published crash rate?

02 Is distributed content data or code, and what does the agent do with content it cannot parse?

03 Can I hold non-urgent content for an interval, on a subset of hosts I nominate?

04 What does a query missing its tenant scope return in your system?

05 Who on your staff can read our events tonight, under what grant, and can I see it in my own store?

06 When you learn from one customer's incident, what exactly crosses the tenant boundary?

07 If you are compromised, how would we find out, and on what timeline are you contractually bound?

08 What is the exit path — our events, our rules, in a form something else can read?

6 · Where this sits in the series

Ten papers argued for consolidation: one agent that observes and enforces, one event model, one graph, one meter. Each argument is also an argument for concentration, and this paper is the bill for it. A reader who accepted the previous ten and never asked what it means to hand one vendor that much privilege has

been reading uncritically — which is not what the series was for.

The four exposures do not disappear under any architecture. What varies is whether they are bounded, whether the bounds are enforced by mechanism rather than policy, and whether you can verify any of it from records you hold. That is a smaller claim than trust, and it is the only one worth making.

> *Ask your security vendor what a compromise of them would do to you. The quality of the answer is most of what you need to know.*

EventLake

ABOUT THIS PAPER

Paper 11 in a series on the architecture of active defense, published by **EventLake Research**. Papers 01–10 cover the closed loop, explainable alerts, legitimacy, rule compilation, bounded autonomy, evidence, continuous trust, coverage honesty, the integration plane, and metering. The series is deliberately vendor-neutral, and the questions in §5 are meant to be asked of us as readily as of anyone else.

EventLake operates with no standing operator access to customer events: grants are time-boxed and reason-bound, every access is an event in the customer's own store, and only compiled findings cross a tenant boundary. **eventlakes.com**