Zero Trust Agent Systems That Pass Audits and Still Ship

Advait Patel, Senior SRE at Broadcom

Docker Captain · Google Developer Expert (Google Cloud) · Creator of OWASP DockSec · OWASP AIVSS Founding Member

QCON AI BOSTON

JUNE 2, 2026

One Bad Day With an Agent

An anonymized account of an agent doing something a human would have been fired for. Caught only because of logging.

The action

A customer-data tool called a production database read it had no business touching. No user instruction covered this. The agent inferred it was useful.

The propagation

Results were passed to a downstream API. Tenant boundary crossed. Data left the account context it was retrieved from.

The detection

Caught 40 minutes later by a log volume anomaly. Not by any purpose-built control. A human noticed the spike.

The realization

None of the existing controls were designed for an agent acting autonomously across tool boundaries. Every control assumed a human in the loop.

⚠ This talk is the playbook I wish I had that week.

Agents aren't a new model.
They're a new identity.

Decides autonomously

Chooses which API to call next without asking.

e.g. Calls your billing API mid-session because it inferred the user needed a refund.

Holds state

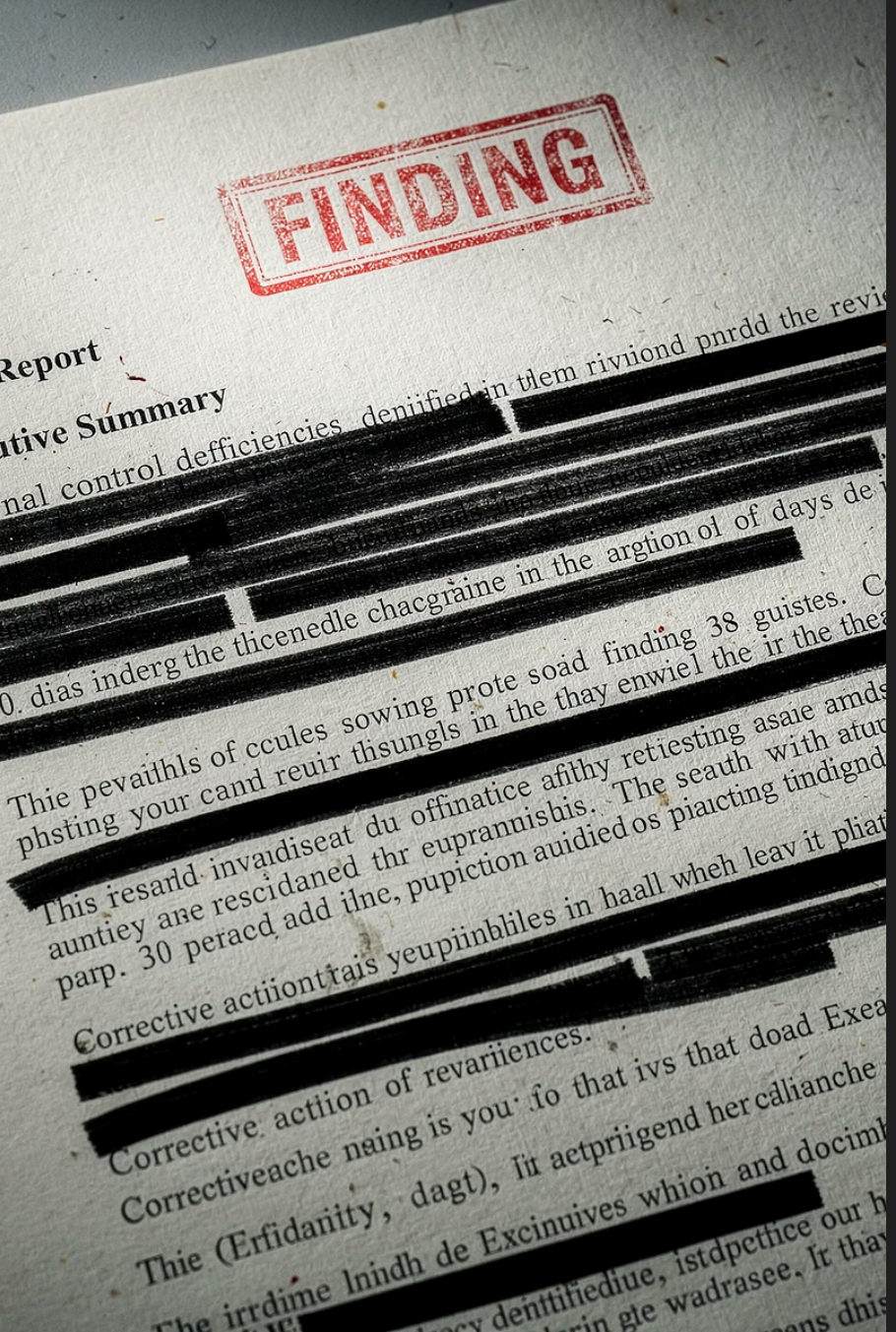
Carries context across tool calls and sessions.

e.g. Remembers a file path from turn 1 and uses it to overwrite data in turn 7.

Outruns review

Acts faster than any human reviewer can respond.

e.g. Completes 40 tool calls in the time it takes a human to open the alert.



Why This Is on Your Auditor's List Now

Eighteen months ago, auditors did not ask agent-specific questions. They do now. The teams getting caught flat-footed are the ones who drew their agent as a chatbot in the architecture diagram.

- SOC 2 and ISO 27001 reviewers now ask agent-specific questions during evidence collection.
- HIPAA and PCI scopes are catching agent data flows that touch regulated fields.
- Internal risk committees want explainability per action, not per session.
- "The LLM decides" is not a control answer. Auditors will write it up.

Four Failure Modes I See in Production

1

Prompt Injection via Tool Output

The retrieved document is the attacker. A malicious instruction embedded in a normal-looking search result hijacks the next action. Most teams filter user input and miss this entirely.

2

Privilege Creep

The agent inherits the union of every tool's permissions. Nobody granted that set explicitly. It accumulated by composition.

3

Unsafe Retrieval

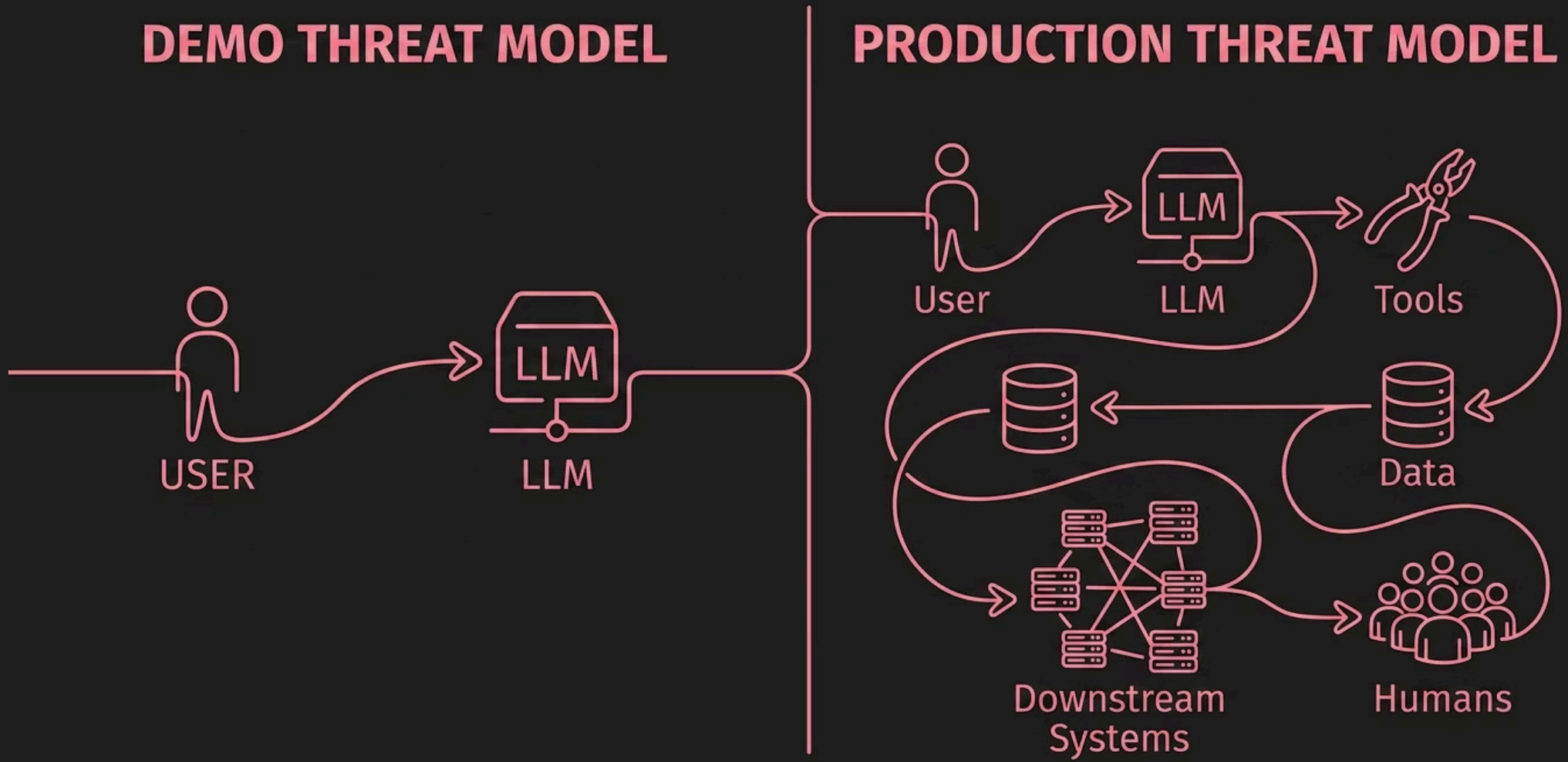
Context crosses tenant or sensitivity boundaries at retrieval time. The model never knows. The data leaves anyway.

4

Action Amplification

One ambiguous prompt fans into dozens of side-effectful calls. Each individually defensible. The aggregate is not.

The Threat Model Nobody Draws

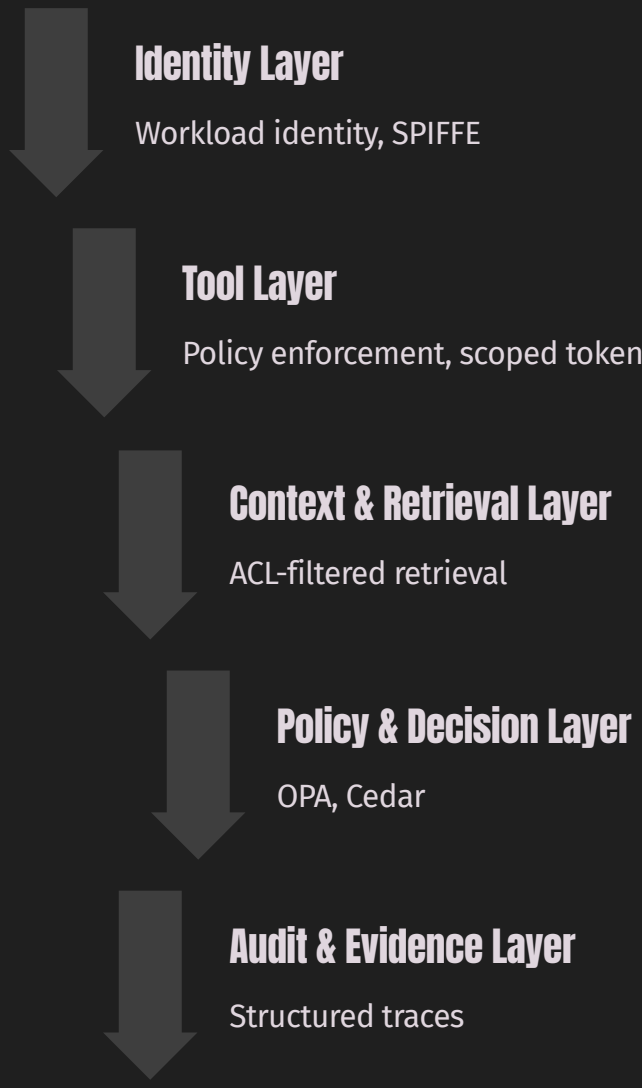


The blast radius lives past the LLM, where the agent touches real systems. Every control in this talk lives on the right side of this diagram.

The prompt is not the boundary.

The tool layer is the boundary.

Reference Architecture (The Slide to Photograph)



The order matters. Each layer assumes the one above it is in place. You cannot skip ahead.

What Each Layer Enforces

Layer	What it enforces	What an auditor asks
Identity	First-class machine identity per agent. No shared service accounts.	"Show me how you tie this action to a specific agent and user."
Tool	Authn, authz, rate limit, I/O validation per tool. Short-lived scoped tokens.	"Show me the token issued for this call."
Context + Retrieval	Doc-level ACLs at index AND query time.	"Show me the ACLs applied to the documents retrieved here."
Policy + Decision	Separate engine between intent and execution. Decisions logged.	"Show me the policy version active during this incident."
Audit + Evidence	Structured trace per action, tied to identity and session.	"Walk me through what happened, from this log alone."

Having an answer for column three is what separates teams that pass on first review from teams that schedule a second one.

STAFF / SENIOR

Tradeoffs You'll Hit at Scale

Decisions in the hot path

Policy evaluation adds 5-20ms per call. Budget for it or you will be asked to remove it under load.

Token lifecycle

Short-lived tokens mean cache invalidation gets complicated. Plan the refresh logic before it bites you in production.

Fail-open vs. fail-closed

This is a real architectural choice. I run fail-closed on destructive actions and fail-open on read-only ones.

Cardinality

Per-session identity explodes label cardinality in traces and metrics. The one that surprised me most.

Policy as SPOF

Centralized policy engine can become a single point of failure. Design for partial degradation from day one.

Guardrails: Identity and Tools

Least privilege at the tool layer

Not the prompt layer. The prompt can say anything. The tool layer is where you enforce what the agent is actually allowed to do.

Scoped tokens only

Token scoped to current task only. No `"ask nicely"` escalation path. If the task requires broader access, that is a design problem.

Tool output is untrusted input

Every tool output treated as hostile input to the next step. Strip or tag any instruction strings found inside retrieved content.

⚠ Tradeoff: scoped tokens add real complexity to caching and refresh logic. Do not underestimate the implementation cost.

Guardrails: Actions, Context, Kill Switches

Allowlist destructive actions

Tier them by blast radius up front. Dry-run, explicit confirm, or human-in-the-loop depending on the tier.

Bounded context

Hard caps on retrieval count, recency, and sensitivity class. Unbounded context is a data leakage surface.

Rate and budget limits

Per agent identity. Cost is a security control. Sudden cost spikes are an anomaly signal.

Kill switch, tested quarterly

The first time you use a kill switch should never be during an active incident. Test it. Log the test. Show the auditor.

⊗ Kill switches that work in seconds require real engineering investment. A checkbox in a runbook is not a kill switch.

Failure Mode to Control Mapping

This is the slide your compliance team will want. These are mappings I use in practice, not official guidance. Verify with your own auditor.

Failure mode	Guardrail	What compliance requires
Prompt injection via tool output	Output sanitization + tagged retrieval	Prove you validate and sanitize all data coming back from external tools before the agent acts on it
Privilege creep	Scoped short-lived tokens per session	Show that each agent session gets only the permissions it needs, and that those permissions expire
Unsafe retrieval	ACL-filtered retrieval, index + query time	Demonstrate that users and agents can only retrieve documents they're authorized to see, enforced at query time
Action amplification	Allowlists + blast radius tiering	Show a defined list of permitted actions and evidence that high-impact actions require explicit approval
Untraceable actions	Structured trace per call, identity-linked	Provide a complete, tamper-evident log linking every action to a specific agent, user, and session
Slow incident response	Token revocation kill switch	Show you can revoke agent access within minutes and that you've tested this under realistic conditions

Live Demo (3 Acts)

Same prompt across all three acts. Same poisoned document. The difference is entirely in what the architecture does with it.

1

Act 1: Naive setup

Indirect injection succeeds. The agent executes the attacker's instruction embedded in retrieved content.

2

Act 2: Hardened setup

Same prompt. Same poisoned doc. Blocked at the tool boundary. No action taken.

3

Act 3: The audit story

Reconstruct exactly what happened from traces alone. Watch the trace viewer. This is where the talk lands.

What I Log, Every Single Call

Fields per call

- Agent identity
- Session ID, user principal
- Tool name + inputs (secrets redacted)
- Outputs (PII handled at write time)
- Policy decision + policy version
- Retrieved doc IDs + ACLs at retrieval time
- Model version
- Stable trace ID across the full flow

Trace structure

- Agent run = distributed trace
- Tool call = span
- Policy decision = event on the span
- Built on `OpenTelemetry`
- Reuses existing SRE tooling

The auditor does not care about your dashboards. They care about whether you can answer "what happened" from logs alone. The fields on the left are the minimum, not the goal.

STAFF / SENIOR

Observability Tradeoffs at Scale

Trace volume gets expensive fast

Sampling that preserves audit evidence is non-obvious. Dropping spans at random breaks incident reconstruction.

PII in tool inputs and outputs

This is a logging policy problem, not a tooling one. No vendor solves it for you. You have to define the boundary explicitly.

High-cardinality identity labels

Per-session identity labels can break your metrics backend. Test cardinality before you go to production at scale.

Retention windows

Must match the longest control framework you are audited under. Auditors will ask. Have a written answer.

Full retention is rarely the right answer. Uniform sampling is also not the right answer. Treat sampling decisions as a control, not a cost optimization.

What I Got Wrong the First Time

1

Started with the policy engine

Should have started with logging. You cannot tune controls you cannot observe. Logging first gives you a baseline and makes every subsequent decision grounded in data.

2

Made every action human-in-the-loop

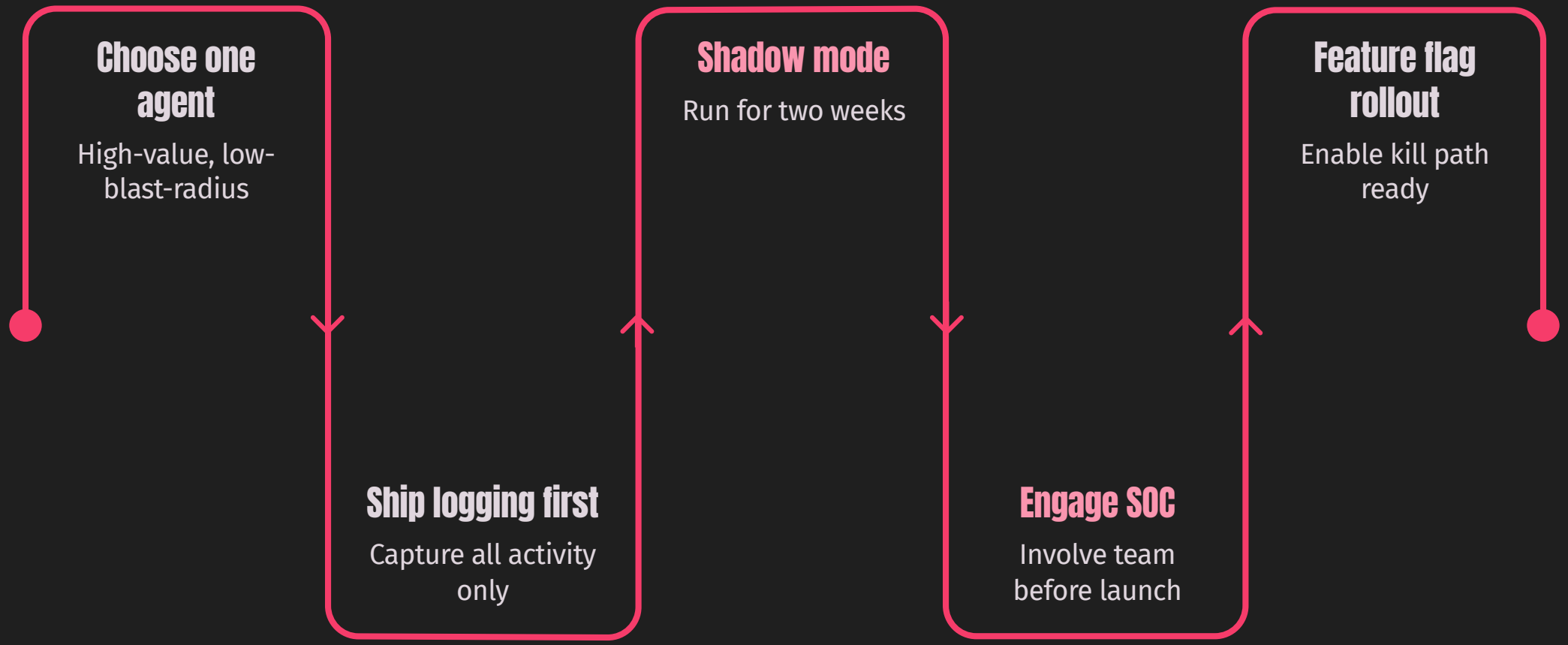
Killed adoption within two weeks. Nobody used the system because every action needed approval. Tiering by blast radius fixed this. Most actions need no human. A few do.

3

Treated prompt injection as a model problem

Spent two months on prompt filtering. The fix was at the tool boundary. The model is not the last line of defense. The tool layer is.

How to Make This Work in Production



The biggest mistake is trying to roll this out everywhere at once. One agent, logging first, shadow mode for two weeks, then controls. Get the SOC team in the loop before launch. They will be the ones paged when something goes wrong.

How to Convince Your Org to Invest in This

Arguments that don't work

- "It's a best practice"
- "OWASP recommends it"
- "Other companies do it"
- "It's the right thing to do"

The arguments on the right are true. They do not move budget. The arguments on the left are in the language your CFO and CISO already use. The cyber insurance angle is the strongest lever I have seen in 2025-2026.

Arguments that work

- Reduces audit prep cost (specific hours saved)
- Improves incident MTTR. Your SOC will back you on this.
- Risk committee and cyber insurance now ask about agent controls explicitly
- One incident pays for the entire investment

Monday Morning

Takeaway 1

Move the boundary from the prompt layer to the tool layer.

Takeaway 2

Every agent = first-class identity. Short-lived scoped credentials. Policy engine in front of every tool call.

Takeaway 3

Build traces as audit evidence from day one. Not a retroactive project.

Seven-question readiness check

1. Can you name every agent identity in your environment?
2. Can your tools refuse an action the LLM tries to take?
3. Is your retrieval filtered by identity before it hits the model?
4. Can you revoke an agent's access in under 60 seconds?
5. Can you reconstruct any action from logs alone?
6. Do you have blast radius tiers for destructive actions?
7. Have you tested your kill switch this quarter?

Advait Patel · linkedin.com/in/advaitpatel93 · Further reading: OWASP Agentic AI, CSA AICM, NIST AI RMF