

AN EXECUTIVE BRIEFING

Application Security in the Age of AI™

You cannot prove what an AI application will do before it runs.
So you must control what it is allowed to do while it runs.

Stephen R. Jordan

Founder, SRJ Consulting & Services LLC

A thick orange vertical bar is positioned to the left of the text.

Eleven controls reported green.
They covered one stage out of six.

Every scan passes. The application still acts on its own.

The old application had three stages: input, code, and output. Every instrument the discipline owns assumes the behavior lives in the code.

An AI application has six: input, interpretation, decision, authority, action, and consequence. Three of the six did not exist before, and the toolchain sees one of them.

Four questions the toolchain cannot answer:

- 01 Which internal applications call a model, and who owns each one?
- 02 What can this application reach, and what can it change?
- 03 What bounded it at the moment it acted?
- 04 What did it actually do, and can you reconstruct it?

Four parties will ask what your applications can reach. On their schedule.

The board

Asks what the enterprise is carrying, and expects a number rather than an assurance.

Internal audit

The most useful ally the function has, and the one that asks for evidence first.

The external auditor and the insurer

Ask which controls were operating, on which applications, on which date.

The regulator

Under the EU AI Act and the erasure regimes, asks where the data went and whether it can be removed.

None of these are going away. The only variable is whether the answer is assembled or improvised.

The Runtime Determinism Gap™

Every application security tool in production was built on a quiet assumption: that an application produces the same output for the same input. AI features invalidated it.

Verifying that an arbitrary agent satisfies a safety property defined by a static permission set reduces to the halting problem. Pre-runtime assurance is not hard. In the general case it is impossible.

What the gap changes:

- 01** Scans certify the code, not the behavior.
- 02** Runtime defenses match patterns the application no longer follows.
- 03** Bug bounties pay for reproducible exploits in a probabilistic runtime.
- 04** Green dashboards coexist with unintended action.

Five instruments. One question each.

The operating model is five working instruments, each answering a question the enterprise must be able to answer out loud, with twelve further instruments supporting them.

Every one is a working document with an owner, a cadence, and the decision it feeds. None is a policy.

Register

What do we run?

Tier map

What can it reach and change?

Authority plane

What bounds it while it runs?

Behavior baseline

What did it actually do?

Security debt

What are we carrying, and what never closes?

Bounded outside the model. Never inside the prompt.

Prompt-level defenses fail under adaptive attack. The bound has to sit somewhere the application did not build and cannot bypass.

The runtime authority plane is a platform service every application inherits, sitting outside the execution step because that is provably the only place it can hold.

What the plane enforces:

- 01** Authority is granted per boundary crossing, not per feature.
- 02** Tokens are downscoped at exchange, so reach is bounded before the call.
- 03** Every discharge is recorded against the policy that allowed it.
- 04** Excessive agency becomes countable in the authority ledger.

Three properties a running estate must hold.

Known

Every application that calls a model is in the register, with an owner and a tier, and coverage is measured rather than assumed.

Bounded at runtime

No application can cause more than its tier permits, enforced outside the model by a plane it did not build and cannot bypass.

Observed

What each application actually did is reconstructable from telemetry the enterprise owns.

Stated as properties a running estate must hold, rather than as principles a team should value.

Untrusted content may inform. It must never authorize.

This is the design rule the Volume turns on, and the evidence behind it is unambiguous.

Trust attaches to a source's write path, never to its location. A document inside the perimeter that anyone can write to is not trusted content.

Where the rule is enforced:

- 01** Retrieval authorization, at the index rather than the answer.
- 02** Tool invocation, bound to the caller's authority rather than the model's intent.
- 03** Deletion, mapped across all five systems a document reaches.
- 04** Multimodal input, classed and bounded like any other untrusted channel.

Three signs the estate has already outgrown the program.

- 01 You cannot produce, in an afternoon, a list of every internal application that calls a model, with an owner against each one.
- 02 A developer can grant an application a credential without the security function learning about it.
- 03 When an application does something unintended, the investigation begins by asking engineers what they remember.

The estate is larger than the register.

The register is almost always a fraction of what is running. The estate spans five populations, and the enterprise finds them four different ways.

The discovery sweep sets the denominator. Until it runs, coverage is a percentage of a number nobody knows.

The five populations:

- 01** Applications the function already knows about.
- 02** Features added to applications that predate AI.
- 03** Vendor products with AI switched on since purchase.
- 04** Departmental builds on low-code and agent platforms.
- 05** Developer tooling with production reach.

Coverage is the number. Everything else is an anecdote.

The program is measured as covered applications over discovered applications, by tier. A board can fund against that. It cannot fund against a feeling.

The baseline scores the program across seven areas, dated and reviewed, and it is the artifact every later instrument reads.

Denominator

The discovery sweep, run on a cadence rather than once.

Numerator

Applications in the register with an owner, a tier, and a bound.

Direction

The escape rate, which says whether findings close faster than they arrive.

The first ninety days, sequenced by dependency.

Each week produces the artifact the next week reads. The sequence has already been discovered, which is the whole value of running it in order.

Weeks 1-4

Baseline scored. Discovery sweep run. The register exists and the denominator is real.

Weeks 5-8

Applications tiered by blast radius. The authority ledger filled. The first bound live.

Weeks 9-12

Telemetry specified, the behavior baseline observing, the first Control Evidence Pack assembled.

The AI Application Portfolio Register

One row per application that calls a model, with an owner, a tier, and a measured coverage position against the discovery sweep.

- 01 Five populations, four discovery methods
- 02 Owner and tier on every row
- 03 Coverage measured, never assumed
- 04 Feeds every other instrument

The Blast Radius Tier Map and Seven Stop Conditions

Applications sorted by what they can reach and change, not by how important they feel. In front of it, seven conditions that refuse what cannot be bounded.

- 01 Reach and change, not perceived importance
- 02 Tier decides the bound and the review
- 03 Stop conditions refuse before design
- 04 Scarce judgment goes where the radius is

The Runtime Authority Plane

A platform service every application inherits, bounding what any of them can cause, outside the model and outside the prompt.

- 01 Authority per boundary crossing
- 02 Downscoped token exchange
- 03 Policy and discharge recorded
- 04 The application cannot bypass it

The Runtime Behavior Baseline

What applications actually do, observed and diffed, so a change in behavior is detectable rather than anecdotal. This is the work Volume VII handed forward by name.

- 01 Telemetry the enterprise owns
- 02 Detection content for behavior, not signatures
- 03 Reconstruct what happened, after the fact
- 04 A consumption envelope with six ceilings

The AI Security Debt™ Model

Findings closed at the rate they arrive, and an honest register of the ones that never close.

- 01 Remediation service levels by tier
- 02 Exception register with expiry
- 03 Escape rate as the direction of travel
- 04 What will not close is stated, not buried

Coverage by tier, and the escape rate beside it.

Two numbers turn an application security backlog into a budget conversation. Coverage says how much of the estate is bounded. The escape rate says whether the program is gaining or losing.

Both are computed from instruments the program already operates, so neither costs a separate reporting exercise.

What leadership sees each cycle:

- 01** Coverage of the discovered estate, by tier.
- 02** Escape rate: findings arriving against findings closed.
- 03** Exceptions live, with expiry dates against each.
- 04** The debt that will not close, named rather than implied.

One assembly answers four audiences.

The Control Evidence Pack answers internal audit, the external auditor, the insurer, and the board from a single assembly, rather than four projects a quarter.

Inside the pack:

- 01** The baseline score, dated and reviewed.
- 02** Register coverage with owners and tiers.
- 03** Authority policy and discharge records.
- 04** Telemetry evidence and the exception register.

How the operating model is stood up.

The book is the operating manual for teams who will run this themselves. The engagement is for organizations that want the baseline scored, the sweep run, the bounds drawn, and the first ninety days sequenced inside a defined timeline.

Scored

The defensible baseline, on your estate, with your CISO.

Discovered

The sweep run, so coverage has a real denominator.

Bounded

The authority plane specified against your platform.

Operated

Instruments handed over with owners and a cadence.

What you operate afterward.

- 01** The Defensible AI Application Security Baseline, scored and dated.
- 02** The AI Application Portfolio Register, with coverage by tier.
- 03** The Blast Radius Tier Map and the Seven Stop Conditions, in use at intake.
- 04** The Authority Ledger and the runtime authority plane policy.
- 05** The Telemetry Specification and detection content pack.
- 06** The Control Evidence Pack, assembled once and reused.

What this program is not.

- 01 Not a tool purchase. No software is sold, resold, or implemented.
- 02 Not a scanner replacement. The existing stack keeps its job and gains a denominator.
- 03 Not a policy document. Every instrument is a working artifact with an owner.
- 04 Not a pre-runtime guarantee. That is the one thing the Gap says nobody can sell you.

Four readers, one operating model.

The CISO

Gets the board question and the baseline score.

The VP of Engineering or Platform

Gets the number, and the plane their teams inherit.

The Director of AppSec

Gets the instrument, with an owner, a cadence, and the decision it feeds.

The AppSec engineer

Gets one worked artifact at build depth in every chapter.

ABOUT THE AUTHOR



Stephen R. Jordan

Founder and Principal Advisor of SRJ Consulting & Services LLC, an operator-led AI advisory practice in Frisco, Texas, built on three decades in enterprise operations, security, and risk.

At Citi he owned endpoint security across roughly 500,000 endpoints in 165 countries. At Intel and McAfee he led product security incident response. The practice sells no software and holds no vendor partnerships.

The Library

Book 08 of 9 in The Operating Discipline for AI Library™, and Book 4 of 5 in AI Risk Governance & Security™.

Publishes

September 16, 2026.

Free today

Every figure from the manuscript, and all twenty-two instruments of the Consulting Toolkit.

Say what your applications can reach. Then bound it.

A 30-minute consultation to scope the question your leadership team needs answered. No deck, no pitch.

415-413-7772

info@srjconsultingservices.com

srjconsultingservices.com/services/risk-governance-security/application-security/