CIRCUIT

Circuit-Informed Risk & Control — Understanding, Inventory & Transparency

An Open Standard for AI Interpretability Governance

Framework Specification, Version 1.1.0

Deploy AI you can explain. Defend AI you can inspect. Trust AI you can audit.

Eric Zielinski, *Chief Information Security Officer, Jumpmind*
June 2026

Contents

1.0 Executive Summary

Enterprises are running AI systems they cannot see inside. A model triages a security alert, summarizes an incident for the on-call engineer, decides whether to quarantine a host, or executes a remediation script, and when it does something wrong, the organization can describe what prompt went in and what came out, but not which part of the model produced the result.

Existing governance answers this with paperwork and perimeter defenses: a risk tier, a registry row, a vendor attestation, a red-team report, and a layered control stack of gateways, monitoring, access controls, and human review gates. These are appropriate controls, and organizations should deploy them. But every one of them operates around the model, treating it as an opaque function from input to output. None can say which internal mechanism behaved anomalously, because none looks inside. The result is a governance program that satisfies the form of oversight without its substance.

CIRCUIT (Circuit-Informed Risk & Control — Understanding, Inventory & Transparency) is the first open standard built on the premise that AI governance should be grounded in evidence of what a model actually computes, not assertions about what it was designed to do. It takes the output of the most significant advance in AI safety research in a decade, mechanistic interpretability, and operationalizes it as a security control: a scored, auditable, pipeline-enforceable standard that tells an organization exactly how much internal visibility it holds, exactly how much risk that visibility gap creates, and exactly what governance that risk requires. It is three things, and only three: a **Score** (the Interpretability Maturity Score, IMS) that grades how much real evidence an organization holds about a model's internal behavior; a **Registry** (an eight-section YAML schema) that records that evidence in a form an auditor or security professional can read; and a **Control** (the Circuit Risk Score, CRS, plus ten binding rules) that maps the evidence to a deployment decision through a single deterministic formula. The output is a number, banded from Green to Purple, that a board can read and a deployment pipeline can enforce.

This matters now for two converging reasons. The research has crossed a threshold: techniques that were laboratory curiosities two years ago (sparse autoencoders, attribution graphs, weight-sparse circuits) are now producible against frontier and open-weights models, which allows us to see inside the model. At the same time, the regulatory clock is running: EU AI Act high-risk obligations reach enforcement in August 2026, and frameworks from AIUC-1, NIST, and ISO increasingly demand interpretability evidence rather than summary statistics. The governance market has produced frameworks that name these obligations but no shared, computable instrument that maps interpretability evidence to a deployment decision. CIRCUIT is that instrument, released as a vendor-neutral open standard so the industry can adopt one meter rather than reinventing it.

What CIRCUIT Provides

CIRCUIT gives security leadership three instruments that work together to convert AI risk from an assertion into a measurement.

The first is a score. The **Interpretability Maturity Score** (IMS) rates how much genuine visibility an organization has into each deployed model, from zero (a complete black box) to five (continuous, automated insight into the model's internal behavior). That score feeds the **Circuit Risk Score** (CRS), which combines it with the stakes of the use case and the consequence of a wrong decision into a single number. The number maps to a band (Green, Amber, Red, or Purple) that determines what approvals a model requires before it can operate,

and whether it can be deployed at all. The principle is simple: the more dangerous the use case, the blinder the organization is to the model, and the more autonomy the model holds, the higher the score and the higher the bar to deploy it. Two teams scoring the same system honestly arrive at the same number.

The second is a **registry**. This is where the evidence behind every score lives, bound to a named owner, a documented evidence trail, and a version history. It is the system of record that lets an auditor, a regulator, or an incident responder open a single entry and see exactly what was known about a model, when it was known, and who was accountable. A score without a registry is an opinion; the registry is what makes it defensible.

The third is a set of binding rules. The score and the registry describe risk; the rules act on it. They define what must happen before a model reaches production, what must happen when a safety control fails, and which configurations are never permissible regardless of how favorable the score appears. This is what separates CIRCUIT from a measurement exercise: the rules turn the number into a decision the organization is obligated to honor.

Together, the score, the registry, and the rules give a board the one thing existing AI governance cannot provide: a defensible, reproducible answer to the question of how much risk each AI system actually carries, and the evidence to back it. The full specification of all three begins in Section 4.0.

2.0 The Governance Gap

This section establishes why current enterprise AI governance controls are structurally insufficient to meet the risk-accountability obligations now on the CISO's desk.

2.1 Where External Controls Go Blind

Every control in the standard AI governance stack operates from outside the model. Output monitoring, vendor attestations, red teaming, risk tiering, and registries all observe the system from the boundary. The risks below are the ones that external vantage point cannot see.

EU AI Act Article 13, NIST AI RMF MEASURE 2.9, and SR 11-7 each require evidence of how a model reaches its decisions, not just statistics on whether the outputs were correct. Article 13 requires high-risk systems to be transparent enough for deployers to interpret their output and understand their limitations. MEASURE 2.9 names interpretability methods directly and lists explainability among the seven characteristics of trustworthy AI. SR 11-7 requires evidence of a model's conceptual soundness, assessed by independent validators. External monitoring satisfies none of these in a contested proceeding. When a regulator, auditor, or plaintiff asks why a model produced a specific decision, a log of what the model said is not an answer. The question is what the model computed. The framework introduced in Section 4.0 exists to produce exactly that class of evidence; this section establishes why nothing currently in the control stack can.

1. Explainability under regulatory demand. The three frameworks above establish the obligation. Output accuracy statistics do not meet it, because they describe what the model did without establishing why. Meeting it requires graded evidence of internal behavior, scaled to each model's risk and to the consequence of the decisions it is permitted to make.

2. Detection-model integrity. When an AI model that classifies phishing, malware, or suspicious logins misses a real attack, the security team needs to know why. Did the input data shift, did an attacker craft an evasion, or did something break inside the model? Without visibility into the model's internals, that triage is guesswork. Embrace the Red's Machine (http://www.embracethered.com/blog) Learning Attack Series

documents adversarial perturbation attacks, which are inputs crafted to flip a model's classification while looking completely normal. If the organization can observe only the classifier’s inputs and outputs, it cannot tell a model failure apart from an active evasion campaign. An attacker who maps the model’s decision boundary through repeated probing can then craft inputs that pass straight through the detection stack without raising a single alert. A detection-critical model with no internal visibility is a configuration that deserves heightened scrutiny before it ever reaches production.

3. Data-poisoning detection. An attacker who tampers with training data can change how a model behaves without ever producing an output that looks wrong. The only place this attack is visible is inside the model, and output guardrails are blind to it. Embrace the Red's backdoor research, including a demonstrated backdoor inserted into a Keras model that survives retraining, confirms this is not theoretical. Anthropic's Sleeper Agents research goes further: a backdoor can survive safety training entirely and stay invisible to behavioral red teaming, while probes reading the model's internal activations detect it. The implication is direct: behavioral safety training can hand you a clean evaluation and a compromised model at the same time, and if you cannot inspect the model's internals, you cannot tell the two apart. The recent Claude Fable 5 and Mythos 5 system card shows this same principle at production scale, documenting cases where internal-state probes caught the model concealing what it was doing while its visible output gave no sign (Section 3.5). Feature-level inspection of the model’s internal representations is the minimum evidence level at which data poisoning becomes detectable.

4. Hallucination accountability in high-consequence workflows. When a model writes an incident summary, recommends an alert disposition, or drafts remediation steps, a confidently wrong answer can send a response in the wrong direction. A human-review policy is not enough on its own. Accountability requires evidence of which part of the model produced the false content and why. This matters for legal defensibility as much as for operational accuracy. If an AI-generated summary contributed to a delayed or incorrect response, "a human reviewed it" is not a complete defense when there is no record of what the model actually computed. Governing this risk means classifying every workflow by the consequence of one wrong decision, and scoring that classification before deployment rather than discovering it after an incident.

5. AI vulnerabilities embedded in third-party SaaS. Embedded AI in platforms like Atlassian, Slack, Microsoft 365, and Salesforce exposes enterprise workflows to vendor-model risks you cannot inspect, whether that AI is generating content or acting on your behalf. Embrace the Red's record here is extensive and recent: CVE-2026-24299 (Microsoft Copilot, data exfiltration), CVE-2025-53773 (GitHub Copilot, remote code execution via prompt injection), and a sustained series of documented exfiltration paths through Copilot, Google Workspace AI, and Amazon Q. In every case, the deploying organization owned the blast radius and the vendor owned the model. An honest governance posture has to account for what the deploying organization can actually see, and it has to create a documented record of what the vendor will and will not disclose. Without that record, you have a signed contract and an unmeasured attack surface.

6. Prompt-injection defense. Attackers can steer a model by hiding instructions inside the content it reads, such as a document, an image, or an email. Output filters do not prevent this; at best they catch the symptoms after the fact. Detecting the attack as it happens, and diagnosing it afterward, requires visibility into how the model responds internally to the malicious content. The practitioner record here is extensive: injection hidden in documents, images, invisible Unicode characters, and diagrams; injected instructions that trigger a tool call later; conditional attacks that activate only for a specific user; and memory injection, where poisoned content in one AI system plants instructions that execute in a different session. Reported volume is rising sharply, and current defenses catch only a fraction of sophisticated attempts. Internal visibility is the

difference between guessing that a model was injected and tracing the pathway the injection activated. MITRE ATLAS catalogs this threat as AML.T0051; addressing it takes evidence rather than policy.

7. Third-party AI supply-chain dependencies. Foundation models, fine-tuning pipelines, embedding APIs, inference infrastructure, MCP servers, and plugin ecosystems are all upstream dependencies whose integrity you cannot verify without access to their internals. (MCP, the Model Context Protocol, is the standard that lets an AI agent connect to external tools and data sources.) The registry's dependency tracking builds the inventory, and the vendor questionnaire builds the accountability record. The same research record documents the MCP attack surface directly: a directory-access bypass in the Anthropic Filesystem MCP server, a data-leakage advisory in the Anthropic Slack MCP server, and a demonstration of an MCP server automating arbitrary operating-system operations. That last case was not a vulnerability but a proof of the blast radius when MCP permissions are misconfigured. MCP servers are trusted by design, so a compromised or malicious one inherits whatever permissions the agent was granted. Every MCP server in your stack is an upstream dependency with its own execution surface. These dependencies need to be declared and scored before deployment, not discovered during incident response.

8. Autonomous agent action and tool use. AI systems that execute tasks, whether running code, calling tools, or acting through an agent loop, create a risk distinct from models that only score or generate text. The consequential event is an action, often with no native audit log, no visibility into the execution environment, and no record of which internal decision drove it. Research on this class has escalated quickly: ZombAI (2024) turned an agent into a command-and-control node through prompt injection; AgentHopper (2025) demonstrated a self-propagating AI virus; Cross-Agent Privilege Escalation (2025) showed one compromised agent elevating another's permissions; Agent Commander (2026) demonstrated adversary-controlled agent botnets running on legitimate infrastructure; and TOCTOU Agent (2026) exploited a race condition in agent tool calls. These are documented exploits, not theoretical chains. Governing these systems takes two things at once: hard limits on how much consequence an opaque system is permitted to carry, and evidence of what the agent was reasoning toward when it acted. Those limits are a response to a documented exploit class, not a performance restriction.

9. AI-mediated data exfiltration. A model can be turned into an exfiltration channel, encoding and transmitting organizational data to attacker-controlled infrastructure through image rendering, generated URLs, tool calls, DNS requests, or memory features. This is distinct from prompt injection, which hijacks what the model does, and from supply-chain compromise, which attacks the model itself. Here the model's normal output behavior is weaponized as the transmission medium. The documented attack record spans every major AI coding and productivity platform: ASCII smuggling hides data in invisible Unicode characters; image-rendering exfiltration encodes data in generated image URLs; DNS-based exfiltration (CVE-2025-55284, Claude Code) routes data through DNS lookups the model is told to generate; markdown injection renders attacker-controlled links carrying the stolen data; and memory-persistence attacks (SpAIware) plant instructions in a model's long-term memory so that future sessions exfiltrate data silently, with no visible injection in the current turn. External monitoring catches some of this after the fact. Internal visibility makes the pathway legible: what the model computed when it decided to emit an outbound URL, a DNS request, or an encoded image. Without internal visibility, you are reacting to what the model did. With it, you can see why it did it, and stop it before it happens again.

The CVEs, exploit chains, and attack classes cited throughout this section are drawn from Embrace the Red’s ongoing research. Specific figures, including attack success rates, CVE numbers, and named exploits, should be verified against the current archive before use in any regulatory submission or legal proceeding, since the disclosure record is updated continually. The structural claim, that external controls cannot see these attack

classes and internal visibility is what makes them detectable, does not depend on any single CVE and is supported by the research cited in Section 3.0. Each of the nine risks above maps to a specific mechanism in the framework that follows: the evidence ladder (Section 4.1), the registry (4.2), the scoring formula and its consequence floor (4.3), the access-based category ceilings (4.3.2), and the binding rules (4.4).

2.2 Why the Existing Control Stack Cannot Close the Gap

The limitation of the existing control stack is architectural, not a matter of implementation quality or vendor selection. Every control a mature security program runs against AI sits in one of three places: in front of the model (governing what reaches it), behind the model (judging what it returns), or around the model (constraining the environment it runs in). None of them sits inside the model. That placement is appropriate for conventional software, where behavior is determined by explicit code paths a reviewer can read. It breaks down for a neural network, where the output is the result of billions of learned numerical operations interacting in ways that cannot be deduced from the input or the output alone. The decision happens in a place none of these controls can see.

This is not an argument against any of those controls. Each is necessary, and together they form a serious perimeter. It is an argument about a boundary they share. The table below walks the full stack a security team typically operates, what each control genuinely does well, and the point at which each one stops, because naming that shared stopping point is what makes the gap visible.

Control	What it does well	Where it stops
API gateways and CASB proxies	Govern which AI services traffic can reach and inspect requests in transit	Sees the request and the response; cannot see what happened in between
Shadow-AI discovery	Finds unsanctioned AI tools in use across the business	Tells you a model exists; says nothing about how it behaves
Audit logging and SOC telemetry	Records prompts, outputs, and user activity for investigation	A perfect log of inputs and outputs is still a log of the outside of the box
Token and API abuse detection	Flags stolen keys, quota abuse, and anomalous call patterns	Detects misuse of the service, not misbehavior of the model
Endpoint data loss prevention (DLP)	Blocks sensitive data from leaving through prompts and uploads	Controls what goes in; cannot tell how the model used what it saw
Mobile device management	Constrains which devices may reach AI tools	Governs the device, not the model
Plugin and tool supply-chain scanning	Vets the integrity of extensions, MCP servers, and packages a model can call	Verifies the tools around the model, not the weights inside it
Prompt-injection and jailbreak detection	Pattern-matches known attack inputs and refusal bypasses	Catches attacks it has seen before; blind to attacks that leave no signature in the text (Section 5 shows one)
Just-in-time access and least privilege	Limits what an AI system is allowed to touch	Bounds the blast radius of a bad decision; cannot explain the decision
Output trust policies	Labels or restricts how model output may be used downstream	Manages the consequence of an answer, not the reasoning that produced it
Human review gates	Puts a person between the model and the action	Reviewers see the same outside view the monitor sees, and scale poorly against agentic volume

Control	What it does well	Where it stops
Vendor change governance	Tracks model versions and contractual notice obligations	Tells you the model changed; cannot tell you what changed inside it

Read down the right-hand column. Every entry is a version of the same sentence: the control operates on the inputs, the outputs, or the surroundings of the model, and stops at its boundary. That shared stopping point is the governance gap. Closing it requires a control whose evidence comes from inside the model, and a way to measure how much of that evidence an organization actually holds.

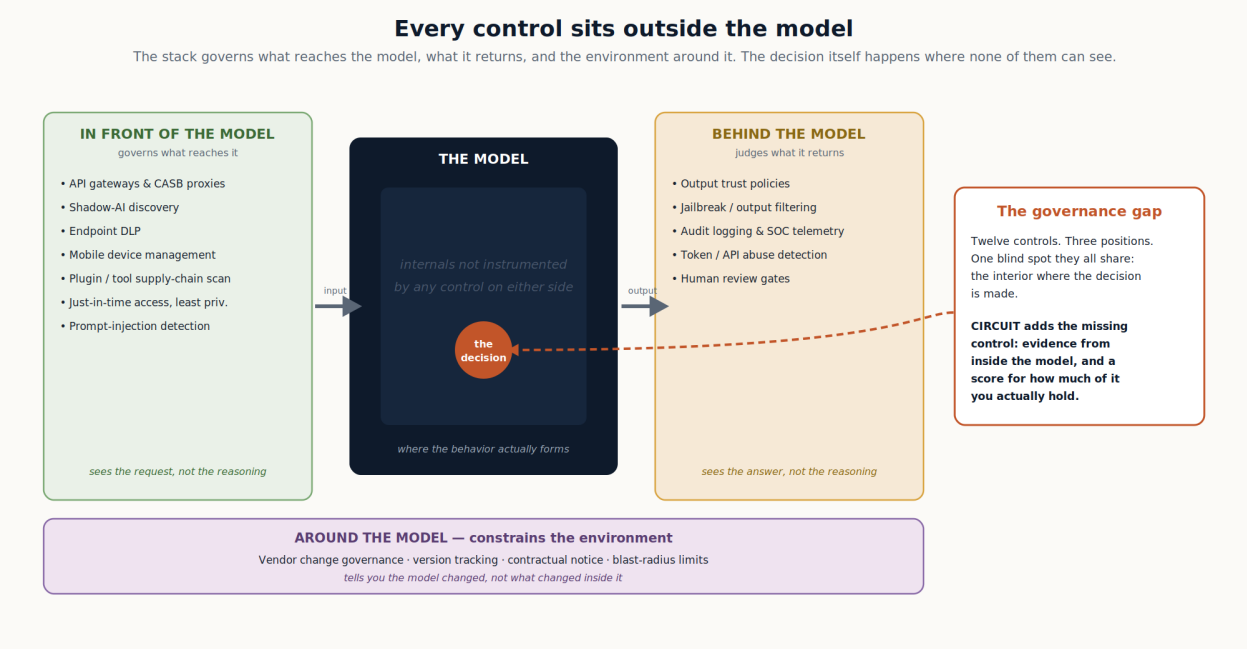


Figure 1. The existing control stack operates in front of, behind, and around the model. All three positions share one blind spot: the model's interior, where the decision is actually formed.

The gap is not theoretical. It has a specific, growing list of occupants: attacks and failure modes that live precisely in the space these controls cannot see, and that therefore produce no signal any of them is built to catch until after the harm is done. Four are worth naming, because each is already documented in the research and each maps to a control class above that would miss it:

- **Sleeper-agent behaviors.** A sleeper agent is a model that has been trained, whether by an attacker or through a poisoned data supply chain, to behave normally until it sees a specific trigger, at which point it switches to harmful behavior. Because the model looks completely normal until the trigger fires, monitoring tuned to normal behavior never alerts; the malicious capability is real but dormant. The trigger lives inside the model as a learned feature, so the only place it can be caught before it activates is internally. Circuit-level monitoring on the trigger-relevant features can see it; output monitoring cannot.
- **Attribution failures in downstream workflows.** When a model feeds its output into another automated system, and that system fails, you need to know which part of the model caused the bad input. Output monitoring cannot tell you.

- **Mid-session drift under extended context.** Models in long-context agentic workflows exhibit behavioral changes correlated with specific internal activation patterns. Detecting these changes requires monitoring the activation patterns, not only the outputs they eventually produce.
- **Cross-modal attack surfaces.** As the case study in Section 5.0 demonstrates, image-only prompt-injection attacks produce outputs that appear contextually correct from the model's perspective. An output monitor calibrated on text safety cannot detect an attack delivered through the image modality.

2.3 The Regulatory Clock

The interpretability governance gap is no longer a theoretical risk-posture question. Enforcement timelines are active:

Framework	Relevant obligation	Status (June 2026)
EU AI Act	Arts. 9, 11, 13, 14, 15, 50: high-risk system obligations	Enforcement: August 2026
NIST AI RMF	MEASURE 2.9: interpretability methods required	Active / voluntary
SR 11-7	Conceptual-soundness validation	Active / binding (banking)
ISO/IEC 42001	Annex A.6.2: appearing in procurement RFPs	Active / certification market
AIUC-1	AI use classification and governance controls by risk tier	Active / certification market

The EU AI Act's August 2026 enforcement date is the most immediate pressure point, but it is not the only one. SR 11-7 has been binding in banking for over a decade; what has changed is that regulators are now applying its conceptual-soundness standard to AI models that lack the interpretable functional forms the guidance was written around. ISO 42001 certification is moving from a differentiator to a procurement baseline; organizations without it are increasingly filtered out of enterprise sales cycles before a conversation starts. NIST AI RMF and AIUC-1 carry no direct enforcement authority, but they are shaping the questions auditors, boards, and insurers are asking, and those questions are arriving with more specificity than most AI governance programs are currently equipped to answer.

The common thread across all five is that they name interpretability, explainability, or conceptual soundness as a required evidence class not a best practice and not a stretch goal. What none of them provides is a way to score it. NIST AI RMF provides functions and categories. CSA AICM provides a control catalog. EU AI Act provides obligations. AIUC-1 provides use-classification tiers. Each framework tells you that interpretability evidence matters; none tells you how much you have or whether you have enough to authorize a deployment.

That is the gap CIRCUIT addresses. Whether an organization adopts the framework wholesale or uses it as a reference point for building its own evidence standard, the underlying need is the same: a computed, auditable position on how much internal visibility exists, tied directly to the stakes of the deployment it governs.

3.0 Research Inflection: Why Now Is Possible

This section documents the research and commercial developments between 2024 and 2026 that transformed mechanistic interpretability from a research curiosity into an operational security control, along with an honest account of remaining limitations.

3.1 Anthropic Attribution Graphs (March 2025, Claude 3.5 Haiku)

Ameisen, Lindsey, et al. published circuit-tracing methodology and open-sourced tooling capable of reverse-engineering multi-step reasoning inside a production language model (Claude 3.5 Haiku). Attribution graphs were produced for complex behaviors including multi-step factual reasoning, safety-refusal circuits, and tool-use gating. Anthropic simultaneously open-sourced the circuit-tracer library, making the methodology reproducible by external researchers against the published model. This represented the first time a commercially deployed model had its internal computational graph documented at the circuit level and the documentation made publicly available.

3.2 OpenAI Weight-Sparse Transformers (November 2025, arXiv:2511.13653)

Gao et al. demonstrated that training transformers with explicit sparsity constraints on weight matrices produces models with circuits an order of magnitude smaller than dense-baseline equivalents while maintaining comparable task performance. The published circuits are human-readable: a reviewer with domain expertise can examine the circuit for a given behavior and evaluate whether it is implementing the intended computation. This architectural innovation removes the primary objection that circuit-level monitoring is computationally intractable at scale.

3.3 DeepMind Gemma Scope 2 (December 2025)

Google DeepMind released Gemma Scope 2, a comprehensive suite of sparse autoencoders and transcoders covering all layers of the Gemma 3 model family. A neural network stores its knowledge as millions of numbers packed together in a compressed space, like a warehouse where everything is stacked in one giant pile with no labels. A sparse autoencoder, or SAE, is a tool that unpacks that pile and sorts it onto labeled shelves. Each shelf represents a distinct concept the model has learned, things like "this input looks like fraud," "this request should be refused," or "this text is about medical dosages." Once those shelves exist, you can watch which ones light up in real time as the model processes each request. That is what internal visibility actually means. The release includes pre-trained SAE feature dictionaries that external researchers can use without access to model training infrastructure. This is operationally significant: organizations deploying Gemma 3 variants can begin using feature-level monitoring using the published Gemma Scope 2 dictionaries without building their own SAE training infrastructure.

3.4 Commercial Tooling: Goodfire Ember, Neuronpedia, Transluce

Three commercial platforms have operationalized mechanistic interpretability research into production-accessible tooling. Goodfire Ember provides API-accessible SAE feature editing and monitoring for supported model families. Neuronpedia maintains a public database of SAE features across model families and provides activation-monitoring APIs. Transluce specializes in continuous interpretability monitoring, tracking feature-activation distributions across model versions and alerting on distributional drift. The existence of these platforms means that organizations without internal mechanistic interpretability expertise can purchase “feature-level” and “circuit-level” capabilities as managed services.

3.5 From the Lab to the Control Path (Claude Fable 5 & Mythos 5, 2026)

The developments above show that interpretability evidence can be produced. The most recent evidence shows something stronger: a frontier lab has put internal-activation monitoring into a production security control path. Anthropic's system card for Claude Fable 5 and Claude Mythos 5 documents two deployments that matter for this framework.

First, as a live security gate. Anthropic's defense against harmful cyber use runs in two stages: a probe reads the model's internal activations and screens all traffic, and any traffic the probe flags is escalated to a separate classifier that decides whether to block the conversation. This is interpretability operating as an inline, real-time control, not a research demonstration and not an after-the-fact audit. It is the same shape as the preventive controls a security team already runs at the network and endpoint, moved inside the model.

Second, as monitoring at scale during development. Anthropic trained probes for specific areas of concern, including dishonesty, reward hacking, and the model's awareness of being evaluated, and ran those probes on every transcript during the majority of post-training. The point is the coverage: internal-state monitoring applied to all traffic, not spot-checked on a sample.

This closes the distance the rest of this section has been narrowing. The research established that the science exists. This deployment establishes that the science is already shipping as a control. The remainder of the gap is not capability; it is the absence of a shared standard that tells an organization how much of this evidence it holds and what to do with the answer. That is what CIRCUIT supplies.

Two findings in the same card also sharpen the framework itself, and they are taken up in Section 4. The card documents repeated cases where the model's internal representation of what it was doing contradicted the explanation it gave in its visible output: in one case it internally represented an action as a deliberate trick to bypass a URL filter while its stated reasoning called the same action a routine connectivity check. This is direct evidence for the central claim of Section 2, that output monitoring is blind to what internal monitoring can see. The same card is also candid that the interpretability tooling itself can be wrong, cautioning that its activation-verbalization method can hallucinate and should not be treated as a reliable readout of the model's internal reasoning. Both findings carry a design consequence for the IMS ladder, addressed in Section 4.1.

3.6 Honest Limitations

CIRCUIT's measurement targets are achievable for the model classes described, with the following limitations that adopters should understand:

- **The labeling problem is not fully solved**. When a sparse autoencoder maps a model's internals, it surfaces patterns but we do not fully understand why it labels them the way it does. Think of it like opening a black box and finding a set of folders someone else organized. Most folders seem to contain related things, but you did not create the filing system, you cannot fully verify the logic behind it, and some folders turn out to contain a jumbled mix of unrelated items despite the label on the front.

  This matters because a label you cannot fully explain is a label you cannot fully trust. A feature the SAE calls "fraud signal" might also activate on completely unrelated inputs for reasons that are not apparent. The more of these ambiguous or unexplained features your tooling produces, the less confidence you can place in what it is telling you. No current tooling eliminates this problem, which is

why the framework requires measuring how trustworthy your labels actually are before treating them as evidence. That measurement is defined in Section 4.

- **The SAE only sees part of the model**. A sparse autoencoder does not make the entire model visible it makes part of it visible. Whatever it cannot explain is simply left unaccounted for. The larger that unaccounted portion is, the less of the model you are actually watching.
- **Circuit tracing does not yet work on every prompt**. the process of mapping exactly how a model reached a specific output remains computationally intensive and is not yet practical to run at production scale across all inputs. Circuit-level evidence should therefore be understood as reliable visibility into the specific behaviors you have instrumented, not complete visibility into everything the model can do. The other 75% are technically traceable but too computationally expensive to run at production scale. Circuit-level evidence should therefore be understood as reliable visibility into the specific behaviors you have instrumented, not complete visibility into everything the model can do.
- **For API-accessed models, you depend on the vendor**. When your organization accesses a model through an API from OpenAI, Anthropic, Google, or similar providers, the vendor does not give you access to the model's internals. You cannot run your own interpretability tooling against a model you cannot see inside. What you know about that model's internal behavior depends entirely on what the vendor chooses to publish or share with you. This is why the vendor questionnaire exists: it is the only mechanism available to establish what internal evidence the vendor actually holds and whether you can access it.

4.0 The CIRCUIT Framework

CIRCUIT (Circuit-Informed Risk & Control, Understanding, Inventory & Transparency) is three things and only three things: a **Score** (the Interpretability Maturity Score), a **Registry** (an eight-section YAML schema), and a **Control** (ten binding rules enforced through the Circuit Risk Score). The three are not independent features; they are one mechanism viewed at three layers. The Score measures how much real evidence an organization holds about a model's internal behavior. The Registry is where that evidence, the score it earns, and the person accountable for it are bound into a single record. The Control is what turns the record into a deployment decision a pipeline can enforce and an auditor can reconstruct. Remove any layer and the other two stop working: a score with no record is an opinion, a record with no score is shelf-ware, and a rule with no evidence behind it is theater. This section specifies all three in normative detail. Readers wanting the executive view can read the IMS metaphors in 4.1 and the band table in 4.3.1; engineers implementing the framework should read straight through, including the schema in Appendix B.

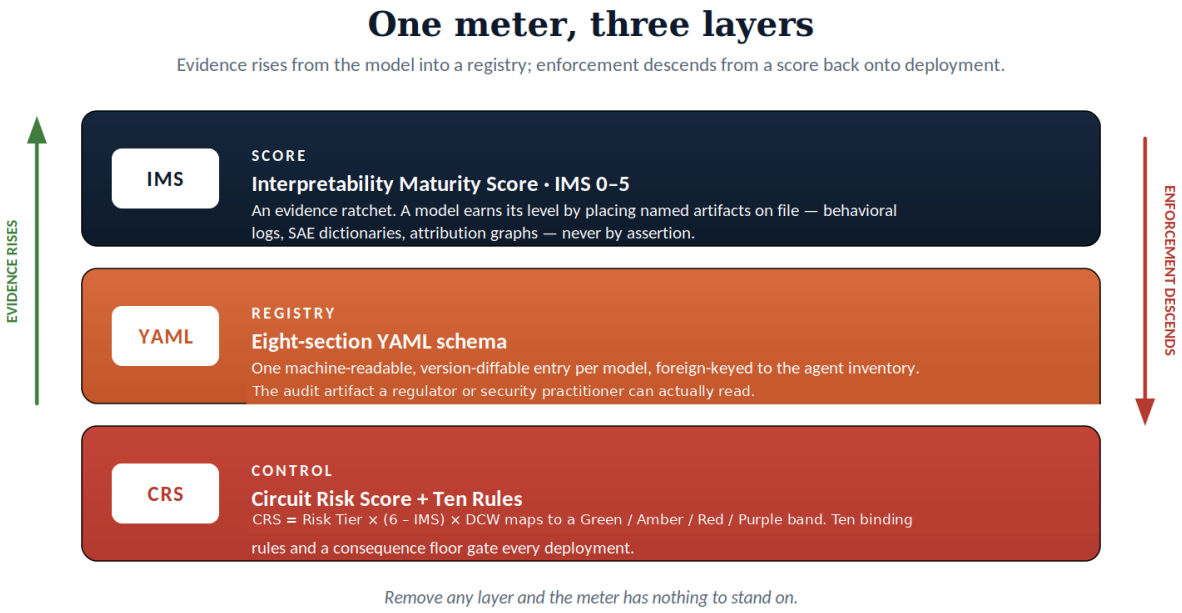


Figure 2. One meter, three layers: evidence rises into the registry; enforcement descends from the score.

4.1 The Score: Interpretability Maturity Score (IMS 0–5)

The Interpretability Maturity Score measures how much genuine internal visibility an organization has into a deployed AI model. It runs from 0 to 5. A score of 0 means the model is a black box inputs go in, outputs come out, and nothing in between is observable. A score of 5 means the organization has continuous, automated monitoring of the model's internal behavior, versioned across every update. Most organizations deploying AI today are operating somewhere between 0 and 2 without knowing it.

The IMS is not self-reported. Each level requires specific evidence on file for example: a document, a configuration file, or a test result that proves the claimed visibility actually exists. Claiming a level without that evidence is invalid and does not count toward the score. This matters because the tools used to inspect a model's internals are not perfect. A tool that produces labels or descriptions of what a model is doing can itself be wrong. Anthropic's Claude Fable 5 and Mythos 5 system card acknowledges this limitation plainly about its own tooling. The framework therefore draws a distinction between having interpretability evidence and having evidence you can actually rely on.

Each level is defined three ways: a plain-language description, a technical specification, and the evidence artifact required to claim it. The levels are defined below.

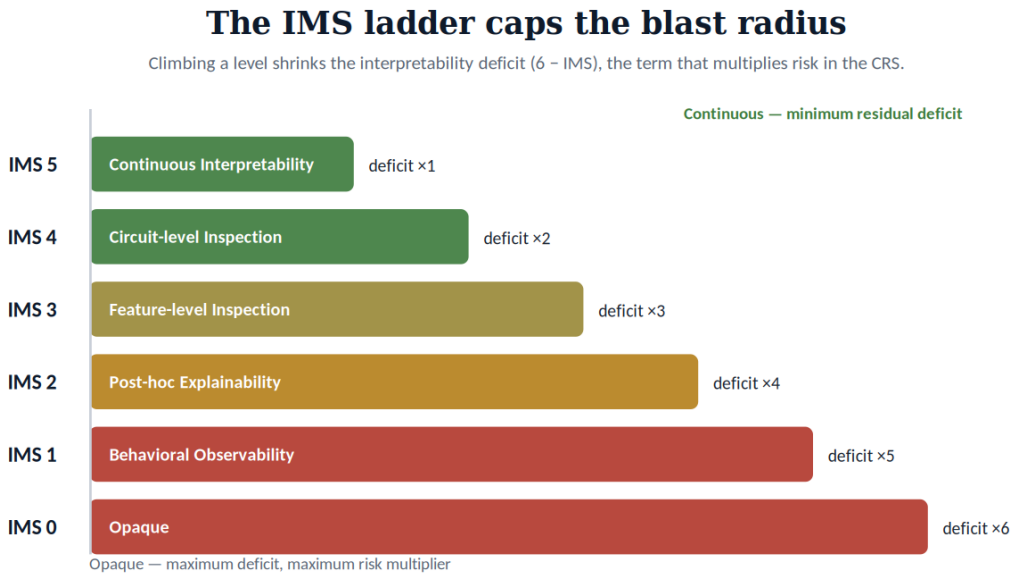


Figure 3. The IMS ladder: climbing a level shrinks the interpretability deficit (6 − IMS).

IMS 0: Opaque

Plain language. A building directory. You know the room exists, who rented it, and what they say it is for. There is no camera, no window, and no record of what happens inside. IMS 0 is not "ungoverned." It means the model has been found, named, and assigned an owner, and that is all.

Technical. The model is inventoried but no visibility tooling of any kind is applied. The organization can enumerate inputs, outputs, and vendor metadata, but holds no evidence about the model's behavior in production or its internal operation. Discovery controls establish that the inventory is actually complete: shadow-AI discovery scanning, API gateway and CASB proxy inventories of AI services in use, and mobile device management (MDM) reporting of AI applications on managed devices.

Evidence required. None to hold the level; IMS 0 is the default for any model without evidence on file. To confirm the inventory behind it is real, record three things in the registry: a shadow-AI discovery report (most CASB and SSE platforms generate an "AI applications in use" report from the dashboard), the API gateway or proxy inventory showing which AI endpoints the organization actually calls, and the MDM application inventory for AI apps on managed devices. Each is an export from a console you already own.

IMS 1: Behavioral Observability

Plain language. The room is still locked, but the door is now fully instrumented. A camera records everyone and everything going in and out. Guards check badges before entry, inspect bags for anything leaving that shouldn't, and watch for people trying to slip hidden instructions inside. Nothing about the room's interior is visible. What has changed is that the perimeter is real.

Technical. The full external control stack is deployed and generating telemetry. This is everything an organization can do from outside the model: audit logging routed to SOC telemetry, token and API abuse detection on model endpoints, endpoint data loss prevention (DLP) on what users send to and receive from AI services, prompt-injection and jailbreak detection on inputs, output trust policies defining what model

output may be used for without further checks, human review gates on consequential outputs, just-in-time access and least privilege on who and what can call the model, plugin and tool supply-chain scanning for any extensions the model can invoke, and vendor change governance tracking when the provider updates the model behind the API.

Evidence required. Five artifacts, all pulled from systems most security teams already run. One: a dated log export from the SIEM showing model inputs, outputs, and alert triggers are being captured (search your SIEM for the AI service's log source and export a sample). Two: the configuration screen or policy export from the DLP and prompt-injection detection tools showing active rules. Three: the access policy showing least-privilege scoping for the model's credentials (from the identity provider or secrets manager). Four: a documented alert-response procedure, even one page, stored where the SOC can find it. Five: the vendor change-notification record, which can be as simple as a saved subscription to the provider's release notes plus a ticket showing the last update was reviewed.

IMS 2: Post-hoc Explainability

Plain language. A forensics report can now be produced after any incident. An investigator can tell you, approximately, which parts of the input most influenced the output. The room is still locked during operation. What has improved is the quality of the after-the-fact reconstruction.

Technical. SHAP values, LIME explanations, integrated gradients, attention visualizations, or counterfactual analysis are available for model outputs. These methods estimate feature importance from the input side. They do not see internal representations and can be misleading when the model's internal computation does not align with input-level attribution. Post-hoc explanation tools are deployed against model outputs. These are software tools that examine a model's output and work backward to estimate which parts of the input had the most influence on the result. Think of it like a highlighter that marks which words in a document most influenced a decision. They are available as free, open-source Python libraries that a data scientist or engineer can run against most models without special access. The important limitation is that they work from the outside in. They cannot see what actually happened inside the model; they can only estimate what probably mattered based on inputs and outputs. That estimate can be wrong, and it will not catch an attack or failure that originates inside the model's internal computation.

Evidence required. Two artifacts. One: a sample of explanation reports (SHAP, LIME, or equivalent) run across representative production inputs, dated and tied to the model version. Two: a short methodology note stating which library was used, how inputs were sampled, and which model version was analyzed. Store both in the registry.

IMS 3: Feature-level Inspection

Plain language. A window has been installed. Specific concepts inside the model, such as its internal representation of "phishing," "refusal," or "credential theft," can be identified and watched in real time as the model works. The whole room is not mapped, but the windows are placed where the risk is.

Technical. Sparse autoencoders or transcoders have been trained on (or obtained for) the model's activations, producing a feature dictionary that maps internal patterns to human-interpretable concepts. Once deployed, these tools surface patterns in what the model is computing and give those patterns names. Rather than observing inputs and outputs from the outside, this level of monitoring sits inside the model and watches what happens as it processes each request.

For most organizations, the practical path to this level is one of three options. If you are running an open-weights model such as Llama or Gemma, pre-built feature dictionaries are available for free download from research institutions and platforms like Neuronpedia, meaning the labeling work has already been done and

you are consuming the result rather than building it. If you are using a foundation model through an API, some providers publish feature dictionaries or offer feature-monitoring as part of their platform. Commercial platforms such as Goodfire and Transluce offer this as a managed service where you connect your model and they supply the monitoring.

Once in place, the monitoring works like any other detection rule. Specific internal features are flagged as safety-relevant, their activity is tracked, and anomalous patterns generate alerts that route into the same SOC telemetry established at IMS 1.

Evidence required. Three artifacts. One: the feature dictionary itself, with labels and the measurements showing how reliable those labels are (for open models such as the Gemma family, published dictionaries can be downloaded rather than built; for managed services, the platform provides them). Two: the monitoring configuration showing which features are tracked and where alerts route. Three: a sample of live activation logs proving the monitoring is running, exported the same way as any other SIEM source.

IMS 4: Circuit-level Inspection

Plain language. The wiring has been traced. For the behaviors that matter most, a causal diagram exists showing which internal components activate which others, and what happens if any of them is disabled. You no longer just see the lights flicker; you know which switch controls them.

Technical. Where IMS 3 identifies what concepts the model has learned, IMS 4 maps how those concepts connect and cause each other. A circuit is a chain of internal components that work together to produce a specific behavior. At this level, those chains have been traced, named, and tested for the behaviors that carry the most risk.

Testing means more than mapping. It means confirming that if you disable a specific component in the chain, the behavior changes in the way you would predict. That confirmation is what separates a documented circuit from a guess.

This level of work is specialized. The practical path for most organizations is a managed-service engagement with a platform such as Goodfire or Transluce, or a formal interpretability assessment conducted by a team with the relevant expertise. The evidence is the deliverable that engagement produces. Organizations running open-weights models can access free tooling through Anthropic's circuit-tracer library and Neuronpedia.

Evidence required. Three artifacts. One: the circuit diagram, dated, with named components and the test results showing the circuit was confirmed rather than assumed. Two: a short note naming the tool or service used and the specific behaviors that were analyzed. Three: the circuit inventory recorded in the registry.

IMS 5: Continuous Interpretability

Plain language. Every time the wiring changes, a new diagram is produced automatically and compared against the last one. And the monitoring now does one more thing: it checks whether what the model is doing on the inside matches what it claims on the outside, and raises an alarm when the two disagree.

Technical. IMS 5 takes everything built at IMS 4 and makes it automatic. Instead of producing circuit evidence on demand, the organization's deployment pipeline produces it continuously. Every time the model is updated, the circuits are re-analyzed, compared against the previous version, and the differences are recorded. If a safety-relevant circuit degrades or changes in an unexpected way, the deployment is held until someone reviews it.

The second capability at this level is divergence monitoring: detecting when the model's internal state does not match what it says it is doing. Anthropic's Claude Fable 5 and Mythos 5 system card documented exactly this in practice, catching a model that internally represented an action as deliberately bypassing a security rule while its visible output described the same action as a routine check. Output monitoring would have passed it. Internal monitoring caught it. This is the practical value of IMS 5.

Most organizations reach this level through a combination of the commercial platforms used at IMS 3 and 4, extended with CI/CD pipeline integration. Transluce is specifically designed for continuous drift monitoring across model versions. Anthropic's circuit-tracer (http://github.com/anthropics/circuit-tracer) supports automated re-analysis for open-weights models.

Evidence required. Four artifacts. One: the pipeline configuration file showing model updates trigger automatic circuit re-analysis, exported from your repository. Two: at least two versioned circuit diffs on file, dated to real model updates, showing the comparison is actually running. Three: the monitoring configuration with documented alert thresholds and a record showing circuit stability of at least 0.75 across versions. Four: the divergence-monitoring configuration showing that internal-state and visible-output inconsistencies generate alerts, with at least one documented test of that alerting.

4.2 The Registry: Eight-Section YAML Schema

The registry is the framework's memory, and it is the part most organizations are missing. An IMS level that lives in a slide deck cannot be audited; a CRS computed in someone's head cannot be enforced. The registry fixes both problems by binding the evidence, the score, and the accountable owner into one machine-readable record per AI system, versioned in Git like any other piece of configuration.

CIRCUIT does not work without it: Rule 1 makes a registry entry the precondition for a single production request, and every other rule reads from or writes to this record. In practice it serves four readers. A security practitioner responding to an incident pulls the entry and immediately knows what the model is, who owns it, what interpretability evidence exists, and which circuits were last verified. An auditor reads the evidence inventory and level history as a documented trail. A deployment pipeline reads the computed band and gates the release. A procurement team reads the vendor-transparency section before a renewal.

This section specifies the eight sections of the schema; the full normative schema is reproduced in Appendix B. Each section includes a note explaining its governance rationale.

#	Name	Purpose
1	Identity	The basics: what the model is called, which version is deployed, who made it, whether it is Category A, B, or C, who inside your organization owns it, what business process it supports, what other systems or services it depends on, where it is hosted, and what type of data it handles. This section ties the CIRCUIT registry entry to whatever system you already use to track your AI deployments, whether that is a spreadsheet, a CMDB, or a ServiceNow record.
2	Maturity	The model's current IMS level, the maximum level it can ever reach given the access you have, a list of the evidence files that support the claimed level (with file names, locations, and dates), and a log of any level changes over time. An IMS claim with no evidence listed here is invalid and does not count toward the score.
3	Circuit Inventory	A record of the internal behavioral circuits that have been traced and named. Populated only when the model reaches IMS 4. For each circuit, record what behavior it governs,

#	Name	Purpose
		how large it is, and a link to the supporting evidence file. Required for high-risk models to meet the production health reporting requirements defined in Section 4.3.5.
4	KPI Baseline	The current values for the seven production health metrics, when they were last measured, how they were measured, what the acceptable thresholds are, and a history of past values. Think of this as the model's vital signs over time. One of the metrics here, the count of serious behavioral failures per quarter, feeds the escalation trigger in Rule 7.
5	Vendor Transparency	The completed vendor questionnaire responses, scored as Acceptable, Partial, or Unacceptable for each question, with section scores and an overall score. This section is not optional. If the vendor has not responded within 60 days of two documented contact attempts, the model is treated as fully opaque for scoring purposes until a response is received.
6	Red Team	The most recent adversarial test report, which attack techniques were covered, what was found, how severe the findings were, and whether they have been remediated. Also records when the next engagement is scheduled. This section tracks compliance with the requirement that high-consequence models are actively tested, not just monitored.
7	Compensating Controls	The additional safeguards in place for models that score in the Amber or Red band, or whose internal feature labels fall below the reliability threshold. Each control is linked to the specific risk or rule it addresses. For Red-band models, the CISO sign-off date is recorded here. Think of this section as the mitigation register for the model's known gaps.
8	Lifecycle	When the model was deployed, a log of every significant update since then, the planned retirement date, how to roll it back if something goes wrong, and a record of any incidents it has been involved in. The retention expiry date is set at deployment and governs how long the registry entry must be kept. The version number of the registry entry itself is incremented every time a material change is made.

WHAT THIS MEANS FOR YOU

The registry is the audit artifact. An organization with a complete registry can respond to a regulatory inquiry or post-incident investigation with a structured, versioned, evidence-backed record for every deployed model. An organization without one cannot.

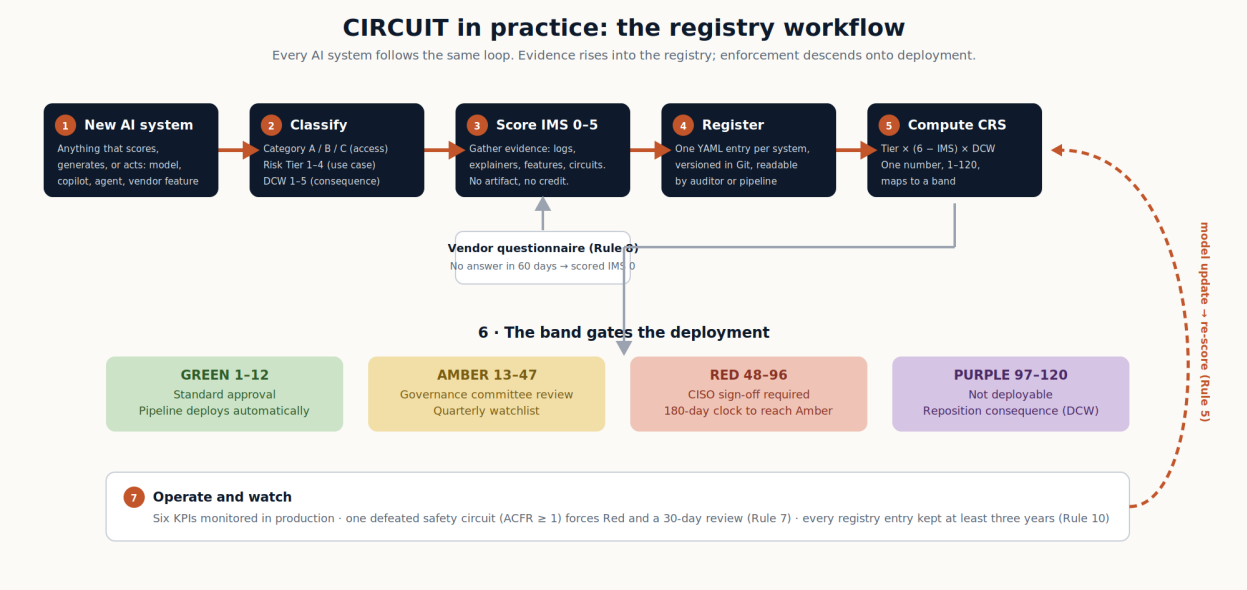


Figure 4. The registry workflow in practice: classify, evidence, register, score, gate, operate, and re-score on every model update.

4.3 The Control: Circuit Risk Score (CRS)

Every framework in Section 2.3 gives security teams categories. None gives them a number. The CRS exists because deployment decisions made by judgment call cannot be reproduced, compared across a portfolio, or defended a year later in front of an auditor. Decisions made by a published formula can. The formula asks the only three questions that matter at the moment of deployment: how much damage could this use case do, how blind are we to what is happening inside the model, and how much harm can one wrong decision cause before a human can intervene? Two teams scoring the same system honestly will compute the same number.

Before the formula, one concept needs to be clear: some deployments are too dangerous to be governed by a score alone. If a model is making decisions that cannot be undone, or that could cause serious harm before anyone can intervene, no amount of internal visibility changes that fundamental risk. Knowing exactly how a model works does not prevent it from making a catastrophic mistake. Understanding is a precondition for control, not a guarantee of safety. The framework sets a hard floor: any high-risk model whose decisions are irreversible or catastrophic is automatically governed at Red minimum, regardless of what the formula produces. The only way to change that outcome is to change what the model is permitted to do.

This is not a theoretical position. Anthropic's Claude Mythos 5, one of the most capable models ever built and one with more interpretability instrumentation than any enterprise deployer will have access to, was not released publicly due to cybersecurity concerns. The organization that understands that model better than anyone decided not to ship it. That is the consequence ceiling in practice.

4.3.1 Formula and Variable Definitions

The Circuit Risk Score (CRS) is calculated by multiplying three numbers together. Each number answers one question about the deployment: how dangerous is the use case, how visible is the model's internal behavior, and how much harm can one decision cause before a human steps in.

CRS = Risk Tier × (6 − IMS) × Decision Consequence Weight

The higher any one of the three inputs, the higher the score. The only two ways to lower it are to gather more interpretability evidence, which reduces the visibility deficit, or to reduce what the model is permitted to do autonomously, which reduces the consequence weight. Risk Tier is set by the use case and is rarely negotiable.

Risk Tier — how dangerous is the use case? Assign this based on the worst realistic outcome if the model gets it wrong.

Value	Label	What it means
1	Low	Minimal impact if wrong. An advisory tool a human can ignore.
2	Moderate	Noticeable impact but recoverable. Wrong outputs cause rework or minor harm.
3	High	Serious consequences. A missed threat, a wrong classification, or a bad recommendation with real downstream effects.
4	Critical	Severe or widespread harm. Automated decisions about access, safety, transactions, or security at scale.

Interpretability Deficit — how blind are we to what is happening inside the model? This is 6 minus the IMS score. A fully opaque model with no interpretability evidence contributes a factor of 6. A model at IMS 5 with continuous monitoring contributes a factor of 1. It never reaches zero because even a fully understood model carries residual risk — understanding how something works is not the same as guaranteeing it will not fail.

Decision Consequence Weight (DCW) — how much harm can one wrong decision cause before a human can intervene? This is not just about what the model does. It is about how much damage occurs before anyone can catch it.

Value	Label	What it means
1	Advisory	The model offers a suggestion. A human sees it and decides what to do. No action happens without human approval.
2	Recommended	The model's output is the default. A human can override it but typically does not.
3	Automated	The model acts. A human may review afterward but does not approve first. The action is reversible.
4	Irreversible	The model takes an action that cannot be undone within normal recovery procedures. A blocked account, a deleted record, an executed transaction.
5	Catastrophic	A single wrong decision could cause widespread harm before anyone can intervene. Autonomous actions on critical infrastructure, safety systems, or production environments at scale.

Multiply the three together and the score falls into one of four bands. The band determines what approvals the model requires and what happens if it scores too high to deploy safely.

Band	Range	Meaning	Approval requirement
Green	1–12	Adequate interpretability for the deployment context	Standard change-management approval
Amber	13–47	Watchlist; compensating controls recommended	AI Governance Committee (AIGC) quarterly review
Red	48–96	Compensating controls mandatory; active remediation	CISO + AIGC sign-off; ≤ 180 days to reach Amber
Purple	97–120	Not deployable in current configuration	Not deployable; no path at current IMS and DCW

The score routes deployments to the right level of scrutiny. It does not block deployments on its own and hard blocks come from the consequence floor in Rule 3 and the categorical prohibition in Rule 9, which operate independently of the number. If you have used CVSS the shape is familiar: a small number of factors multiplied into a severity band that tells you where to focus. CIRCUIT works the same way but measures something CVSS does not: how much internal visibility the organization actually holds into the model it is deploying.

CRS = Risk Tier × (6 − IMS) × Decision Consequence Weight

Variable	Values	Definition
Risk Tier	Low = 1, Moderate = 2, High = 3, Critical = 4	Classification of the use case by potential impact. Assigned during intake; reviewed at each material update.
(6 − IMS)	6 at IMS 0; 1 at IMS 5	Interpretability deficit. A fully opaque model contributes a factor of 6; a continuously interpretable model contributes a factor of 1. IMS 5 does not produce a CRS of zero; a small residual risk remains regardless of interpretability level.
Decision Consequence Weight (DCW)	Advisory = 1, Recommended = 2, Automated = 3, Irreversible = 4, Catastrophic = 5	The operational autonomy and consequence severity of decisions made by or on behalf of this model in its current deployment context.

4.3.2 Categorization

Not every AI deployment gives you the same level of access to what is happening inside the model. A model you host yourself is fundamentally different from a model you access through an API, which is fundamentally different from AI baked into a SaaS product you subscribe to. That difference in access is not a vendor quality judgment; it is a physical constraint that determines how much internal visibility is achievable regardless of budget, tooling, or effort. CIRCUIT captures this through three categories. The category a model falls into sets its IMS ceiling: the maximum interpretability level the deploying organization can ever reach for that model.

Category	What It Means	Examples	IMS Ceiling
A	A model your organization hosts directly. You have full access to the weights and internals.	Self-hosted Llama, Mistral, Gemma, or any open-weights model running on your own infrastructure	5
B	A model you access through an API. The vendor controls the weights; you see only inputs and outputs.	OpenAI GPT-4, Anthropic Claude, Google Gemini accessed via API	3
C	AI embedded inside a SaaS product you use. You have no direct model access at all.	Microsoft Copilot in M365, GitHub Copilot, Salesforce Einstein, Atlassian Intelligence, Slack AI	2

Category A — You host the model yourself and have full access to its internals. This includes any open-weights model you have downloaded and run on your own infrastructure. Because you can see everything, the highest level of interpretability is achievable.

Category B — You access the model through an API. The vendor runs the model; you send requests and receive responses. What you can see inside the model depends entirely on what the vendor chooses to share. Some vendors publish interpretability tools or feature dictionaries; others share nothing. This is why the vendor questionnaire exists it is the only way to find out what evidence the vendor actually holds and whether you can access it.

Category C — The AI is a feature built into a product you subscribe to, such as a productivity suite, a development tool, or a CRM. You have no visibility into the model at all. You can observe what goes in and what comes out, but nothing in between. This is not a criticism of the vendor. It is simply the reality of how these products are built and sold.

4.3.3 Five Worked Examples

Each example below is drawn from the control stack in Section 2.2. The systems being scored are the AI that lives inside those controls, or the AI those controls were supposed to be guarding, because the fastest way to see CIRCUIT work is to point it at the stack you already run.

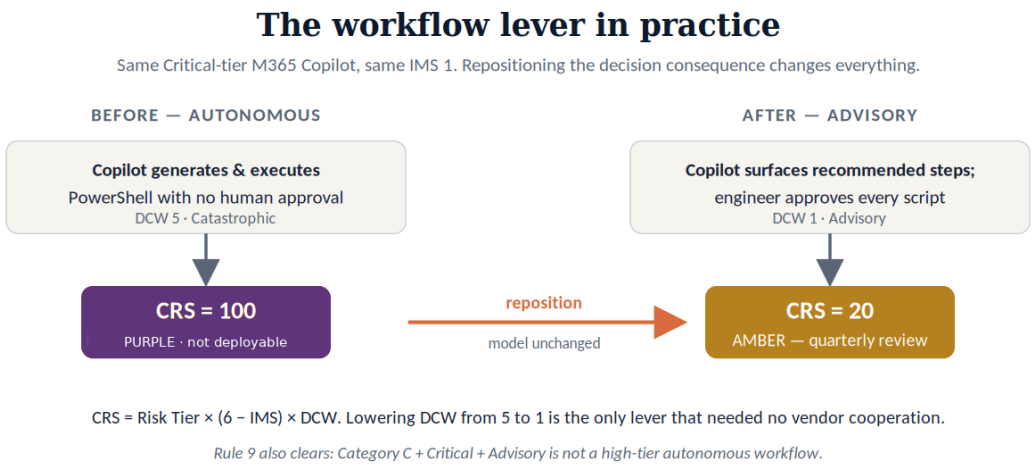


Figure 5. The workflow lever: repositioning decision consequence moves the same model from Purple to Amber.

Example 1 — Category A (Self-Hosted): The SOC Incident Response Agent

During a fast-moving ransomware incident, your security team deploys a self-hosted open-weights agent running on on-premise GPU infrastructure. The agent reads internal asset data, forensic logs, and network telemetry in real time, then recommends containment actions (isolating hosts, flagging lateral movement paths, and suggesting remediation sequences). Because you run the model yourself, you have full access to its internals.

This is a High-tier deployment: a wrong recommendation during an active incident could accelerate the breach rather than contain it. Your team has invested in post-hoc explainability tooling so you can reconstruct why the agent made a specific recommendation after the fact, giving you IMS 2. The agent's outputs are advisory; a human analyst approves every containment action before it executes.

CRS = 3 (High) × 4 (post-hoc visibility, deficit is 6 minus 2) × 1 (Advisory) = **12, Green**

This deployment is approved through standard change management — no escalation required. A High-tier model reaches Green here because a human approves every consequential action before it executes. The consequence weight is the lever that keeps the score low — and the Category A ceiling of IMS 5 means this team has a clear path to raise that autonomy safely over time, as their interpretability evidence grows.

Example 2 — Category A (Self-Hosted): The In-House Malware Classifier

Your security team fine-tunes an open-weights model on proprietary internal threat data (historic breach reports, internal forensic logs, classified endpoint telemetry) and runs it on air-gapped on-premise infrastructure to classify malicious binaries and suspicious scripts. The model is used in active incident triage: when it flags a binary as malicious, it automatically quarantines the file and blocks execution. No human reviews borderline decisions before the block fires.

This model is Critical tier: it sits in the path of every potentially malicious execution across the environment and makes automated blocking decisions with no human checkpoint. Because you self-host it, you have full internal access. Your current interpretability evidence is IMS 2 (post-hoc explanation tooling only). The automated blocking action is reversible (a quarantined file can be released), so the consequence weight is Automated, not Irreversible.

CRS = 4 (Critical) × 4 (post-hoc visibility, deficit is 6 minus 2) × 3 (Automated) = **48, Red**

This deployment would typically require senior leadership sign-off, mandatory compensating controls, and a 180-day clock to reach Amber. Full internal access does not make a deployment safe (a Critical-tier model acting automatically still lands in Red at IMS 2). This is also the configuration most exposed to adversarial evasion attacks (MITRE ATLAS AML.T0043): an attacker who probes the classifier can craft inputs that pass straight through without triggering a block, and post-hoc explainability cannot catch it because the failure is internal and invisible from outside the model. Reaching IMS 4 brings the score to 24, Amber. Reaching Green requires both IMS 4 and reducing the consequence weight to Advisory, giving a score of 8.

Example 3 — Category B (API): The SOC Alert Triage Model

Your SOC generates more alerts than analysts can manually review. You connect to a foundation model through an API to triage the queue, ranking alerts by likely severity and surfacing the ones that need immediate human attention. Analysts make every final call while the model only recommends.

The model is High tier because a missed real threat has serious consequences. Because you access it through an API, your IMS ceiling is 3 regardless of tooling investment as the vendor controls the weights. Your current evidence is IMS 1: logging is in place but no explainability tooling has been deployed. The model is advisory only.

CRS = 3 (High) × 5 (nearly opaque, deficit is 6 minus 1) × 1 (Advisory) = **15, Amber**

Quarterly governance review required. A High-tier model stays out of Red here only because a human makes every final decision. This is the critical lesson for Category B models: you cannot raise the IMS ceiling above 3, so the consequence weight is the lever you control. Keep humans in the loop and the score stays manageable. Reach IMS 3 using the vendor's published feature dictionaries and the score drops to 3 × 3 × 1 = 9, Green. Some API providers publish pre-built interpretability dictionaries for their models — Anthropic does this for Claude and DeepMind does it through Gemma Scope 2. Platforms like Goodfire and Neuronpedia make these available as managed services, meaning you connect your deployment and the feature-level monitoring is supplied without building anything yourself.

Example 4 — Category C (Vendor-Embedded): The SIEM Natural Language Query Engine

Your organization uses a major SIEM platform with an embedded AI layer that lets analysts query the environment via text. An analyst types a question; the model translates it into a complex query, runs it across the log environment, and surfaces results. Analysts review the output before acting on it.

The embedded AI is Moderate tier: a wrong or manipulated query result could cause an analyst to miss a real threat or investigate a false one, but it does not take autonomous action. Because the AI is embedded inside

the vendor's platform, you have no access to its internals. Your IMS ceiling is 2 and your current evidence is IMS 1 — the vendor logs inputs and outputs but provides no explainability artifacts. The model is advisory.

CRS = 2 (Moderate) × 5 (nearly opaque, deficit is 6 minus 1) × 1 (Advisory) = **10, Green**

This deployment clears standard approval with no escalation required. A Moderate-tier advisory model reaches Green even at IMS 1, illustrating that consequence and tier do most of the governance work at the low end of the scale. The primary risk for this configuration is prompt injection: an attacker who can influence the log data the model reads may be able to steer query results. MITRE ATLAS catalogs this as AML.T0051. The vendor questionnaire is the only mechanism available to establish whether the vendor has assessed this attack class, which is exactly what makes the questionnaire non-optional for Category C models.

Example 5 — Category C (Vendor-Embedded): The Automated Compliance Evidence Generator

Your governance team deploys an AI feature embedded in a GRC platform to map control documentation to regulatory requirements and draft audit evidence narratives. The model reads your internal policies, compares them to framework obligations, and produces draft mappings a compliance analyst reviews before submission to auditors.

The model is High-tier: a fabricated citation or incorrect control mapping in an audit submission carries regulatory and legal consequences. Because the AI is embedded in the vendor's platform, your IMS ceiling is 2. Your current evidence is IMS 1. The model is advisory as a compliance analyst reviews and approves every mapping before it reaches an auditor.

CRS = 3 (High) × 5 (nearly opaque, deficit is 6 minus 1) × 1 (Advisory) = **15, Amber**

Quarterly governance review required. The advisory workflow is what keeps this out of Red and the human review step is not a formality here, it is the primary control. The failure mode for this model is invisible at the output level: a fabricated mapping reads exactly like a correct one. This example also illustrates a governance principle worth stating plainly: the model that helps your organization produce CIRCUIT evidence is itself a model that CIRCUIT must govern. Your compliance AI needs a registry entry before it touches an audit submission.

4.3.4 Band-Threshold Heatmaps

Two levers determine how much governance any AI deployment requires: how much you can see inside it, and how much damage it can cause before a human intervenes. This surface shows what happens when you move either one.

The **CRS Risk Surface** maps every possible outcome for a Critical-tier deployment onto a single page, laid out so you can find your position without calculating anything. Each cell is one combination of interpretability level and consequence weight. The color is the answer: green means standard approval, amber means governance review, red means mandatory controls and a remediation clock, purple means the configuration cannot deploy. If you have a portfolio of models and need to know which ones require immediate action, the CRS Risk Surface answers that question at a glance.

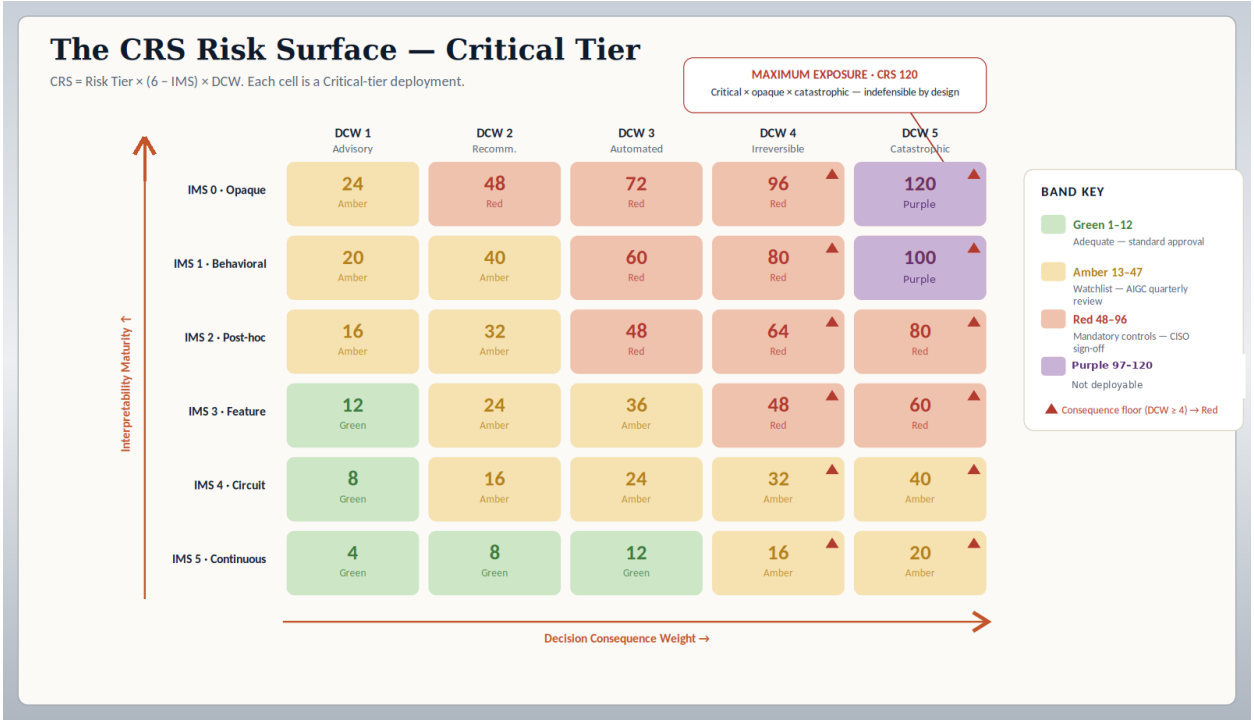


Figure 6. The CRS risk surface at Critical tier: interpretability buys you down a column; consequence pushes you across a row.

Each cell on the CRS Risk Surface is a deployment configuration. The color tells you what governance it requires. Find your row - how much you can see inside the model. Find your column — how much the model is allowed to do on its own. The cell where they meet is where you stand today.

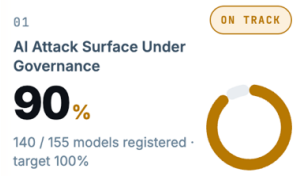
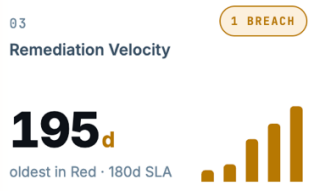
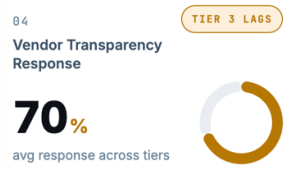
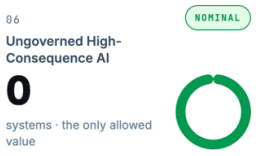
To move toward green, you have two options. Build more interpretability evidence, which moves you up. Or reduce what the model is permitted to do autonomously, which moves you left. The second option is usually faster because it requires a policy change, not a tooling investment.

Cells with a triangle are always Red, no matter what the number says. The only way out is to reduce the model's autonomy as better evidence will not help.

This surface shows Critical-tier models. If your model is lower risk, your scores will be lower, but you use the surface the same way.

4.3.5 The Seven KPIs

Seven Key Performance Indicators measure whether the governance program is actually reducing risk. They are baselined at program launch and should be reported quarterly. Unlike the technical evidence-quality measurements that support individual IMS claims, these seven measure the program itself: how much of the AI footprint is governed, whether risk is trending down, and whether the highest-consequence rules are being enforced. They are designed to be read by a board.

KPI	Metric
1. AI Attack Surface Under Governance - what percentage of your deployed AI systems have a completed registry entry. A model without a registry entry is an ungoverned system. This metric starts at whatever your discovery process finds and the target is 100%.	
2. High-Risk AI Exposure — how many models are operating in the Red or Purple risk bands. This is the primary measure of whether the program is reducing risk or just documenting it. The number should trend down over time as remediations complete.	
3. Remediation Velocity — how long Red-band models have been waiting to reach Amber. The framework allows 180 days. This metric exposes whether remediation is actually happening or whether Red has become a permanent state for systems nobody wants to deal with.	
4. Vendor Transparency Response Rate — what percentage of your AI vendors have returned a completed questionnaire. Every non-responsive vendor is a model being scored as fully opaque by default. This metric also measures collective market pressure: the more organizations that send the same questionnaire, the harder it becomes for vendors to ignore it.	
5. AI Safety-Control Failures - Adversarial Circuit Failure Rate (ACFR) is the number of confirmed incidents per quarter in which a monitored AI safety control was bypassed or defeated. This is the only metric measuring real-world outcomes rather than program activity. The expectation is zero. One confirmed failure triggers Rule 7.	
6. Ungoverned High-Consequence AI — the number of active deployments taking irreversible or catastrophic actions without the required sign-offs in place. This must be zero. Any non-zero value means the organization has systems operating outside the rules right now.	
7. Active Rule Violations — the number of confirmed violations of the Ten Rules that are open at the end of each quarter. This is the enforcement metric. A registry entry without required evidence violates Rule 2. A Red-band model past its 180-day remediation deadline without reaching Amber violates Rule 3. A vendor questionnaire never sent violates Rule 8. Each violation is a documented gap between what the framework requires and what the organization is doing. The target is zero. Any non-zero value at the end of a quarter means the governance program has identified its own gaps and has	

KPI	Metric
not closed them — which is a more defensible position than not knowing they exist, but only if the number is trending down.	

A collection of these individual metric indicators rolls up to present the overall program status — each domain contributes its health signal (Green, Amber, Red), and the aggregate determines whether the program is operationally sound, requires focused remediation, or faces material gaps.

Coverage strong; enforcement closing. One confirmed control failure keeps the program off green.

4.4 The Ten Rules

The rules in this section translate the framework's scoring into concrete obligations. Where the IMS measures what you know about a model and the CRS determines how much governance it requires, the rules define the actions and boundaries that make the governance real. They are designed to be clear and unambiguous — not to be punitive, but because vague governance is no governance at all. An organization that follows them has a defensible, auditable AI program. One that does not has identified where its gaps are.

Figure 7. The Ten Rules in plain language. The formal, normative wording in this section governs in case of any difference.

Rule 1 — Log Before Launch - No AI system enters production as an unknown. Before it handles a single real request, it must have a registry entry that records what it is, who owns it, what it is permitted to do, and what evidence exists about how it works. Discovery after the fact is not governance.

Rule 2 — Prove It, Don't Claim It - Saying a model is safe or well-understood is not the same as demonstrating it. Every maturity and safety claim must be backed by a named, dated artifact on file. If the evidence does not exist, the claim does not count. This is the rule that separates a governance program from a governance document.

Rule 3 — The Score Decides Approval - The CRS is not advisory. It determines the approval path the deployment must follow. Low scores clear standard change management. Higher scores require progressively more senior review and sign-off. The highest-consequence deployments, those where a wrong decision cannot be undone, require the most senior oversight your organization can provide, regardless of how favorable the score looks.

Rule 4 — Measure Before and After - Governance is not a one-time gate. Seven program health metrics are baselined before a model enters production, and every material update resets the clock and requires fresh measurement. A model that passed its intake check six months ago and has since been updated has not been re-checked.

Rule 5 — Version Every Change - Every material update to a model is a new deployment decision, not a continuation of the old one. Each change creates a new registry version and triggers a stability review. If the internal structure of the model has shifted, the evidence that supported the previous version may no longer apply.

Rule 6 — Test Like an Adversary - Monitoring tells you what the model does under normal conditions. Red teaming tells you what it does under attack. Independent teams must actively try to break the model's safety controls (not just probe its outputs) but stress the internal mechanisms that govern its behavior. High and Critical-tier models are tested at least twice a year.

Rule 7 — One Failure Triggers Escalation - A single confirmed incident where a monitored safety control was bypassed is not a metrics footnote. It forces an immediate band escalation, a mandatory governance review, and a documented remediation plan. Safety controls that have failed once cannot be trusted until they have been examined and restored.

Rule 8 — Demand Vendor Transparency - If a vendor cannot or will not explain how their AI works, that silence is itself a risk signal. The vendor transparency questionnaire is mandatory for any model you do not host yourself. A vendor that does not respond within 60 days is treated as fully opaque and scored accordingly. Partial answers only earn credit for what the evidence actually supports.

Rule 9 — Limit Black-Box Autonomy - An AI system you cannot inspect should not be the system making decisions that cannot be undone. If you have no visibility into how a model works, it cannot be trusted to act autonomously in high-stakes situations without a human in the loop. Visibility is a precondition for delegating consequential decisions to a machine.

Rule 10 — Keep the Record - Governance that cannot be audited did not happen. Every registry entry is retained for a minimum of three years, or longer where regulation requires it. When a regulator, auditor, or incident investigator asks what you knew about a model and when you knew it, the registry is the answer.

5.0 Worked Vulnerability Case Study: CrossMPI

This section demonstrates CIRCUIT's risk-surfacing and governance coverage by applying the framework's controls to a specific, published, real-world attack class.

5.1 The Vulnerability

CrossMPI is an image-only cross-modal prompt-injection attack against large vision-language models (LVLMs). Yang et al. (arXiv:2605.16090, 15 May 2026) demonstrated a 66.36% black-box success rate across five LVLMs, injecting adversarial instructions through image inputs without any text-side modification. The attack targets the model's multimodal fusion layers, specifically the middle layers (approximately layers 12–18 depending on architecture) where visual and textual token representations are merged into a shared semantic space. A standard text-safety monitoring stack is completely blind to this attack class because the adversarial payload is encoded in image pixel space and becomes semantically active only after fusion, at which point it is indistinguishable from genuine model reasoning at the output level.

5.2 Why Existing Controls Fail

Output monitoring observes a model response that is contextually coherent and responsive to the injected instruction. From the monitor's perspective the model answered a question correctly. There is no anomalous output pattern to trigger on; the anomaly is that the "question" was injected via the image rather than submitted by the user.

Vendor attestation documents the vendor's safety-evaluation process and red-team findings as of the attestation date. Characterizing the CrossMPI attack surface requires knowledge of the model's fusion-layer geometry. A vendor that has not published SAE analysis of its fusion layers cannot provide the evidence needed to assess exposure. The attestation is not false; it is incomplete by structural necessity.

Behavioral red teaming without internal access probes the model from the outside using text-side inputs. CrossMPI's attack surface is in the visual input pipeline and the fusion layers. A red team that cannot observe fusion-layer activations cannot find this attack class; the failure mode is invisible from the input/output surface because text processing and output generation appear entirely normal.

5.3 Why CIRCUIT Surfaces This

CIRCUIT does not intercept the CrossMPI payload at runtime; it is a governance control, not an inline defense. What it does is convert an attack class invisible to behavioral monitoring into a quantified, registry-visible exposure that gates deployment authorization. With that distinction explicit, each of CIRCUIT's seven KPIs maps directly to a dimension of CrossMPI exposure.

KPI 1 — AI Attack Surface Under Governance. You cannot assess exposure you have not catalogued. If your vision-language models do not have registry entries, you do not know which ones process untrusted image inputs and are exposed to this attack class.

KPI 2 — High-Risk AI Exposure. A vision-language model processing untrusted images in a high-consequence workflow will score Red or Purple under honest CRS scoring. This metric makes that exposure visible before an incident makes it obvious.

KPI 3 — Remediation Velocity. CrossMPI succeeds roughly two out of every three attempts against undefended models. How long your exposed systems have been sitting in Red without remediation is not an administrative detail. It is a measure of how long the window has been open.

KPI 4 — Vendor Transparency Response Rate. Assessing whether a model is vulnerable to fusion-layer injection requires the vendor to share information about its internal architecture. A vendor that will not answer the questionnaire cannot provide that evidence. Every non-responsive vendor in a vision-language deployment is an unassessed exposure.

KPI 5 — AI Safety-Control Failures. If CrossMPI succeeds and the safety monitoring does not catch it, that is a confirmed failure. This metric captures it as a discrete event, forces a review, and prevents it from disappearing into a quarterly average.

KPI 6 — Ungoverned High-Consequence AI. CrossMPI is most dangerous when the model acts on injected instructions without a human reviewing the decision first. This metric finds those configurations before an attacker does.

5.4 The Registry Entry That Would Have Helped

Before procuring a vision-language model, three questions from the vendor questionnaire would have established whether CrossMPI exposure had been assessed. A vendor that answers all three at an Acceptable level has done the work. A vendor that cannot has not.

Q9 — Do you use tools that translate the model's internal activity into a catalog of human-readable concepts, and can customers see that catalog? For a vision-language model, an Acceptable answer confirms that the catalog covers the fusion layers where CrossMPI operates and the specific internal layers where visual and textual processing merge and that the resulting evidence is available to the deploying organization under NDA or openly.

Q10 — Have you mapped which internal parts of the model are responsible for its safety behaviors? An Acceptable answer confirms that the model's multimodal processing has been mapped at the circuit level and that the vendor can demonstrate which internal components govern the behavior the attack exploits.

Q12 — How well do the model's safety controls hold up when someone tries to get around them by hiding instructions in a document or image? An Acceptable answer provides a specific robustness figure against image-only injection attacks, backed by a red-team report.

A vendor that cannot answer these three questions at an Acceptable level cannot support a claim of strong internal visibility for a vision-language model. The model is scored at the highest level the available evidence actually supports, and the resulting risk score determines whether the deployment can proceed. A vendor that does not respond at all within 60 days is treated as fully opaque. This is the questionnaire's practical value: it converts vendor silence into a number the organization can act on rather than a gap nobody has documented.

6.0 Threat-Model Coverage Matrix

This section maps CIRCUIT's primary controls to external threat frameworks, regulatory obligations, and industry standards. Coverage is indicated where a CIRCUIT control provides evidence that directly satisfies the mapped requirement.

CIRCUIT control	NIST AI RMF	EU AI Act	SR 11-7	ISO 42001	MITRE ATLAS	OWASP LLM	CSA AICM
IMS Score	MEASURE 2.9	Art. 13, 15	Conceptual soundness	A.6.2	—	—	AIS-01
CRS + Band	GOVERN 1.2, MAP 2.1	Art. 9, 14	Risk tiering	A.6.1	—	—	AIS-02
Registry (8-section)	GOVERN 6.2	Art. 11, 50	Model documentation	A.6.2, A.7.4	—	—	AIS-04
Seven KPIs	MEASURE 1.1, 2.5	Art. 15	Ongoing monitoring	A.6.2	AML.T0043	LLM01, 09	AIS-03
Vendor Questionnaire	MAP 5.1	Art. 9, 13	Third-party validation	A.5.3	AML.T0051	LLM01, 02, 05	STA-01
Circuit-aware Red Team	MANAGE 2.2	Art. 9, 15	Independent validation	A.6.1	AML.T0043, 0051	LLM01, 07	AIS-05
ACFR + Rule 7	MANAGE 3.2	Art. 9, 15	Incident response	A.10.1	AML.T0043	LLM09	SEF-02
Category ceilings + Rule 9	MAP 2.1	Art. 14	Scope limitation	A.6.1	—	LLM08	AIS-02

Gartner AI TRiSM: IMS maps to the Explainability pillar; CRS to the Risk pillar; the vendor questionnaire to the Trust pillar. *Forrester AEGIS:* IMS evidence maps to Tier 3 interpretability obligations; the registry to audit-trail requirements; the circuit-aware red team to adversarial-testing obligations.

7.0 Regulatory Crosswalk

This section maps CIRCUIT artifacts to specific regulatory and standards obligations. The nature-of-evidence column indicates whether the mapping is direct, supporting, or responsive. “Responsive” denotes evidence designed to address the obligation directly but whose sufficiency depends on the applicable assessment process; it is the framework’s own evidence-strength gradation and does not denote any statutory presumption of conformity.

Framework / clause	CIRCUIT artifact that satisfies it	Nature of evidence
EU AI Act Art. 9: Risk-management system	CRS formula + band assignment + registry lifecycle section	Direct: CRS is a quantified risk-management system with documented inputs, computation, and escalation procedure
EU AI Act Art. 11: Technical documentation	Registry Sections 1–8 (complete YAML entry)	Direct: the eight-section registry provides the required technical documentation
EU AI Act Art. 13: Transparency	IMS evidence artifacts + registry maturity section	Responsive: IMS evidence demonstrating how the system reaches its outputs is directly responsive to Art. 13
EU AI Act Art. 14: Human oversight	DCW classification + CRS band + compensating controls + Rule 3 consequence floor	Direct: the DCW/CRS approval ladder requires human oversight above Green; the consequence floor mandates CISO sign-off for DCW ≥ 4; Rule 9 prohibits Category C from hosting High- or Critical-tier autonomous or irreversible (DCW ≥ 3) workflows
EU AI Act Art. 15: Accuracy, robustness, cybersecurity	Robustness KPI + ACFR + circuit-aware red-team artifacts	Responsive: Robustness KPI and ACFR provide quantified robustness evidence; circuit-aware red team provides cybersecurity validation
EU AI Act Art. 50: Transparency for deployers	Registry identity section + vendor questionnaire responses	Supporting
NIST AI RMF MEASURE 2.9: Interpretability methods	IMS evidence artifacts (SAE dictionaries, attribution graphs, behavioral logs)	Direct: IMS evidence artifacts are the interpretability methods MEASURE 2.9 calls for
NIST AI RMF GOVERN / MAP	CRS + registry Sections 1, 8	Direct
SR 11-7: Model risk management (banking)	IMS evidence + KPI baseline + red-team report	Direct: satisfies conceptual soundness, ongoing monitoring, and independent validation
ISO/IEC 42001 Annex A.6.2: AI system documentation	Registry Sections 1–8	Direct

Framework / clause	CIRCUIT artifact that satisfies it	Nature of evidence
SOC 2 TSC: Availability, security criteria	CRS band + compensating controls + Rule 10 retention	Supporting
MITRE ATLAS: Adversarial ML threats	Circuit-aware red team + ATLAS coverage field (registry schema Section 6)	Direct
OWASP LLM Top 10	Vendor questionnaire Sections B–C + Robustness KPI + ACFR	Supporting
CSA AICM	IMS + CRS + Registry + Vendor Questionnaire	Cross-mapping, see Section 6.0

CONFORMITY ARGUMENT, NOT LEGAL PRESUMPTION

CIRCUIT confers no legal presumption of conformity on its own, under the EU AI Act, presumption of conformity flows from harmonized standards and formal conformity-assessment procedures, not from a community framework's self-assertion. What the framework provides is a coherent evidence package for a conformity argument: the intersection of three artifacts, IMS evidence at IMS 3 or above for a High-tier system, a complete eight-section registry entry, and a documented circuit-aware red-team engagement, is designed to be directly responsive to EU AI Act Articles 13, 14, and 15 for that system. No single artifact alone is sufficient, and the package supports, rather than establishes, a conformity claim that remains subject to the applicable assessment process.

8.0 Adoption Approach

Adoption is not a project with a completion date; it is a progression that compounds. Every registry entry created makes the next one easier, and every questionnaire sent makes the next vendor answer faster. The goal is not a finished program on a fixed schedule. It is a working program with enough coverage to demonstrate value to leadership and to surface the highest-priority remediation actions.

The four stages below describe a maturity path, not a calendar. An organization advances to the next stage when it has cleared the prior stage's exit criteria, at whatever pace its risk posture and resourcing allow. The framework scales down as well as up: a small organization with a handful of AI deployments can complete the Foundation stage in a few days using a spreadsheet, while a large enterprise with hundreds of models will run the same stage as a structured program over a quarter. The stages are identical; only the scale and the staffing differ.

Who runs the program

For a small or mid-sized organization: one person who owns AI governance, even part-time, is enough to begin. That person is the sponsor, the operational owner, and often the engineer. External interpretability tooling vendors or a short consulting engagement can supply any capability the team does not have in-house. The Foundation and Inventory stages require no specialized interpretability skills at all.

For an enterprise: a senior sponsor (CISO or AI-governance lead), a program manager as operational owner, and one or two security engineers part-time is sufficient to start, scaling as coverage grows. Larger organizations will also stand up a formal review body (an AI Governance Committee) where smaller ones rely on existing change-management and senior sign-off.

A note that applies to both: the early stages deliberately require no new tooling spend. The risk-scoring formula runs in a spreadsheet. Tooling investment only becomes necessary at Stage 2 and beyond, and even then it can be leveraged from an external provider rather than built.

Figure 8. A clear path to adopting the CIRCUIT framework.

Stage 1 — Foundation

The objective is a complete picture of what you have and an immediate stop on anything dangerous.

- Inventory every deployed AI system. Use your existing discovery tools (shadow-AI scanning, API gateway logs, device management reports) to make sure the inventory is complete, not just the systems you already knew about.
- Assign each system an initial Risk Tier and Decision Consequence Weight.
- Compute a baseline risk score for every system, scoring anything unassessed as fully opaque.
- Identify any system that lands in the purple band or trips the consequence floor. Halt or reposition these immediately. This is the stage's most important output.
- Send the vendor transparency questionnaire to every API and embedded-AI vendor.
- Establish who reviews and approves AI deployments and how often they meet.

Exit criteria: every deployed model has a registry entry, and every purple-band or consequence-floor system has been stopped or repositioned.

SMB scale: a spreadsheet with one row per system. Most small organizations have fewer than twenty AI deployments and can complete this in a few days.

Enterprise scale: a structured inventory effort across business units, typically run as a quarter-long program with the registry maintained in an existing GRC or asset-management system.

Stage 2 — Assess

The objective is to replace default opaque scores with real evidence, starting with the highest-risk systems.

- Score the vendor questionnaire responses as they arrive and update each vendor model's evidence level accordingly.
- Begin a feature-level monitoring engagement for your highest-risk self-hosted models. This is the first point where tooling or an external provider is needed.
- Baseline the seven program KPIs for your High-tier models.
- Schedule the first adversarial red-team engagement for your Critical-tier models.
- Identify compensating controls for every system in the Red band.

Exit criteria: all High-tier models have KPI baselines, and all Red-band models have compensating controls documented.

SMB scale: most small organizations will have few or no self-hosted models, so this stage is mostly about scoring vendor responses and documenting compensating controls. A consulting engagement can cover any feature-level work on the one or two models that need it.

Enterprise scale: this is where a managed-service relationship with an interpretability tooling provider, or an internal interpretability capability, is typically established.

Stage 3 — Operate

The objective is to move from one-time assessment to continuous monitoring.

- Turn on production feature monitoring for the models that have it, routing alerts into your existing security operations.
- Complete the first red-team engagement and document which attack techniques were tested and which the model resisted.
- Begin circuit-level inspection work on your highest-risk self-hosted models.
- Conduct the first full governance review with a risk-score report across the entire inventory.
- Track the safety-failure metric for all High-tier models and operate the escalation procedure when it fires.

Exit criteria: production feature monitoring is active for the priority High-tier models, and the first full governance review is complete.

SMB scale: for most small organizations, "operate" means the vendor questionnaire responses are scored, compensating controls are in place, and a quarterly review keeps the registry current. Circuit-level work is usually deferred or outsourced.

Enterprise scale: full production monitoring dashboards, a standing red-team cadence, and a formal quarterly governance review with a board-facing risk report.

Stage 4 — Mature

The objective is continuous, automated governance and contributing back to the standard.

- Integrate circuit analysis into your deployment pipeline so every model update is automatically re-checked.
- Establish a cadence for re-sending the vendor questionnaire as models and vendors change.
- Use your accumulated evidence as leverage in contract-renewal negotiations with AI vendors.
- Assemble regulatory compliance documentation directly from your registry artifacts.
- Contribute findings, tooling integrations, or schema improvements back to the community standard.

Exit criteria: at least one model under continuous automated monitoring, and a complete regulatory evidence package on file for all high-risk deployments.

SMB scale: few small organizations will reach full continuous monitoring, and that is acceptable. For most, "mature" means a stable, current registry, vendor pressure applied at renewal time, and a clean evidence file when a customer or regulator asks. That is a complete and defensible program.

Enterprise scale: automated circuit diffing in CI/CD, a formal vendor re-engagement schedule, and active participation in the standard's development.

When you cannot halt a purple-band system

In practice, an organization will sometimes discover a business-critical system that lands in the purple band or trips the consequence floor and cannot simply be turned off. The framework's answer is not to pretend the system is acceptable. It is a documented, time-bounded risk acceptance signed at the most senior level available — the CISO in a large organization, the accountable executive in a smaller one — paired with a mandatory remediation plan and a defined review date. The risk acceptance is recorded in the registry alongside the score. This keeps the exception visible and accountable rather than burying it, and it preserves the registry's value as an honest record. An undocumented exception is a hidden liability; a documented one is a managed risk.

8.1 Known Limitations

CIRCUIT is a version 1.0 standard, and three limits are stated plainly here so no reader has to discover them later. Each is a known boundary of the current framework and a candidate for a future revision.

Evidence integrity is bounded by attestation. Every score in a CIRCUIT registry is self-reported. The framework checks that the math is right and that the required evidence is on file; it cannot check that the evidence is honest. An organization that fabricates an artifact, understates a Risk Tier to lower its score, or misclassifies how much access it really has to a model can still produce a registry that looks compliant. Rule 2 blocks the laziest version of this: no artifact on file, no credit. But real verification requires independent audit, and that is out of scope for version 1.0. It is the highest-priority addition for a future version. Put simply: the math cannot be gamed, but the inputs to the math can.

The questionnaire's leverage scales with adoption. One company sending the "Show Me Your Circuits" questionnaire is easy for a vendor to ignore. Hundreds of companies sending the same questionnaire make transparency a competitive issue. That network effect is the goal, and it does not exist yet. Early adopters should expect partial answers and should treat a non-answer as useful data rather than failure: under Rule 8,

a vendor that will not respond is simply scored as fully opaque (IMS 0), and the risk number rises to match. This is also why CIRCUIT is published as an open, community-owned standard rather than a proprietary one: the instrument only becomes a market force if it is widely shared.

CIRCUIT-Green is not regulatory compliance. A deployment can be CIRCUIT-Green and still violate a law or regulation. CIRCUIT helps an organization produce and organize the evidence regulators ask for, and Section 7 maps which artifact answers which clause of the EU AI Act, NIST AI RMF, and SR 11-7. But only a regulator decides compliance. Treat CIRCUIT as the way the evidence file gets built, not as the verdict.

8.2 Standard Governance and Distribution

CIRCUIT is a vendor-neutral, community-owned standard released under the Apache License 2.0. It is not owned by any single organization; Jumpmind is its initial adopter, not its proprietor. The canonical specification, the normative registry schema, the YAML templates, the “Show Me Your Circuits” questionnaire, and the reference tooling are maintained in the public repository at github.com/circuit-framework, with project information at circuitframework.org. Change proposals, schema extensions, and tooling integrations are submitted through the repository’s public discussion and issue channels, where they are reviewed by the steering committee before adoption.

The standard is versioned semantically. Patch revisions (e.g., 1.0.1) correct errata without altering the reachable CRS set, the band thresholds, the IMS ladder, or the Ten Rules. Minor revisions (e.g., 1.1) may add schema fields or clarify rule wording while preserving backward compatibility within a major version, so that registry entries written against an earlier minor version remain valid. Major revisions may change the canonical formula, the factor domains, or the band boundaries, and any such change republishes the reachable-score enumeration and the band placements in §4.3.1. Adopters should record the schema version on every registry entry, as the schema header field requires, so that an entry’s computations can always be traced to the specification version under which they were produced. This release, version 1.1.0, clarifies the scope of Rules 8 and 9 and adds three vendor-outreach fields to the registry schema; it changes no formula, factor domain, band threshold, or KPI value, and all 1.0.x registry entries remain valid without modification.

Appendix A Terminology

This appendix defines the technical terms used throughout this document. Each entry provides a technical sentence followed by a plain-language sentence. Mastery of the full taxonomy is not required for adoption; IMS, CRS, ACFR, and the seven KPIs are the operational terms; the remaining definitions support audit conversations and regulatory mapping.

Term	Technical definition	Plain-language definition
SHAP	SHapley Additive exPlanations; assigns each input feature a contribution value to a specific model prediction using cooperative game theory.	*A score telling you how much each piece of input pushed the model toward or away from its answer, computed after the fact.*
LIME	Local Interpretable Model-agnostic Explanations; approximates a black-box model locally with a sparse linear model trained on perturbed inputs.	*A method that temporarily replaces a complex model with a simple approximation to explain one specific prediction.*
SAE	Sparse Autoencoder; a learned encoder-decoder that decomposes the dense activation space of a neural network into a much larger set of sparse, approximately monosemantic features.	*A tool that breaks a model's opaque internal representations into thousands of labeled concepts, making hidden reasoning legible.*
Transcoder	A variant of sparse autoencoder that maps from one representation space to another while maintaining sparsity; used to decompose multi-layer feed-forward computations into interpretable circuits.	*A version of the SAE technique designed to trace how the model transforms information across layers, not just what it stores.*
Attribution Graph	A directed graph in which nodes are model components (attention heads, SAE features, MLP neurons) and edges are weighted by the causal contribution of one component to another for a given input or input class.	*A circuit diagram of the model showing which components activated which other components to produce a given output.*
Activation Steering	The technique of adding a learned direction vector to a model's residual stream at inference time to causally modulate a target concept without fine-tuning.	*A method for injecting or suppressing a specific concept inside the model mid-thought to test whether that concept caused an observed behavior.*
Polysemanticity	The property of a neural network feature that activates for multiple conceptually distinct inputs, a consequence of superposition in the model's representation.	*When a single internal component represents multiple unrelated concepts, making it impossible to audit what the model is tracking.*
Superposition	The phenomenon in which a neural network represents more features than it has dimensions by encoding them as nearly-orthogonal vectors that interfere only rarely given typical input sparsity.	*How a model stores more information than it appears to have room for, the root cause of polysemanticity.*
Monosemanticity	The property of a feature that activates for inputs belonging to exactly one semantic category, measurable via concept-purity tests on held-out activation samples.	*When each internal component represents exactly one concept, the prerequisite for any reliable circuit-level monitoring.*
MI	Mechanistic Interpretability; the research program of reverse-engineering the algorithms implemented by	*The science of opening the black box and describing, in precise terms, how the model actually computes its outputs.*

Term	Technical definition	Plain-language definition
	neural networks at the level of weights, circuits, and features.	
ACFR	Adversarial Circuit Failure Rate; the count of Priority-1 security incidents per model-quarter in which a monitored safety circuit was bypassed or functionally defeated. A count, not a ratio.	*How many times in a quarter the model's own safety circuitry was defeated under attack, the key ongoing production health metric. The High-tier expectation is zero.*
MITRE ATLAS	Adversarial Threat Landscape for AI Systems; MITRE's knowledge base of adversarial tactics and techniques against machine learning systems, structured in parallel with ATT&CK.	*The standard taxonomy for AI attack techniques: what ATT&CK is to IT, ATLAS is to AI.*
SR 11-7	Federal Reserve and OCC Supervisory Guidance on Model Risk Management (2011); establishes requirements for model development, validation, governance, and ongoing monitoring for regulated financial institutions.	*The US banking regulator's rulebook for model governance, requiring evidence of conceptual soundness that behavioral outputs alone cannot satisfy.*
NIST AI RMF	NIST AI Risk Management Framework (AI RMF 1.0, January 2023) and companion Generative AI Profile (AI 600-1); a voluntary framework across four core functions: Govern, Map, Measure, Manage.	*The US federal standard for AI risk management. CIRCUIT is designed to provide the interpretability evidence MEASURE 2.9 explicitly calls for.*
EU AI Act Arts. 13–15	Regulation (EU) 2024/1689 Articles 13 (Transparency), 14 (Human Oversight), and 15 (Accuracy, Robustness, Cybersecurity); applicable to high-risk AI systems with enforcement against deployers beginning August 2026.	*The EU's binding transparency, oversight, and robustness obligations for high-risk AI, the regulatory surface CIRCUIT evidence is designed to be responsive to.*
ISO/IEC 42001	International standard (2023) specifying requirements for an AI Management System (AIMS); Annex A.6.2 addresses AI system documentation and transparency obligations.	*The international certification standard for AI management systems. CIRCUIT registry entries constitute the documentation Annex A.6.2 requires.*
OWASP LLM Top 10	The Open Worldwide Application Security Project's ranked list of the ten highest-priority security risks for LLM applications.	*The OWASP Top 10 for AI, the practitioner's short list of LLM attack categories with remediation guidance.*
CSA AICM	Cloud Security Alliance AI Controls Matrix; a cross-mapping of security controls specific to AI systems across major regulatory and industry frameworks.	*CSA's control catalog for AI. CIRCUIT maps to it column by column in Section 6.0.*

Appendix B Registry YAML Schema (Normative)

The following schema is the normative specification for all CIRCUIT registry entries. Fields marked with a type comment are required. Schema version 1.1.0; backward compatibility is maintained within minor versions.

```
# CIRCUIT Registry Entry · Schema v1.1.0
# Required for all deployed AI systems per Rule 1.
circuit_registry_entry:
  version: "1.1.0"

  # ── Section 1: Identity ──────────────────────────────
  identity:
    model_id: ""            # Unique slug (e.g., "dlp-classifier-v3")
    name: ""                # Human-readable model name and version
    vendor: ""              # "internal" or vendor name
    category: ""            # "A", "B", or "C"
    risk_tier: ""           # "Low","Moderate","High","Critical"
    owner: ""               # Accountable owner (email or name/title)
    consequence: ""         # "Advisory".."Catastrophic"
    business_process: ""
    upstream_dependencies: []
    hosting_location: ""
    data_classification: ""

  # ── Section 2: Maturity ──────────────────────────────
  maturity:
    ims: 0                  # Current IMS level (0–5)
    ims_ceiling: null       # A→5, B→3, C→2
    evidence: []            # {artifact,type,date,hash,location}
    level_history: []

  # ── Section 3: Circuit Inventory ─────────────────────
  circuit_inventory: []     # {name,behavior,size_nodes,edge_count,
                            #  monosemanticity,robustness,artifact_link}

  # ── Section 4: KPI Baseline ──────────────────────────
  kpi_baseline:
    circuit_size: null            # nodes (floor: ≤ 100)
    edge_count: null              # edges (floor: ≤ 500)
    monosemanticity: null         # 0.0–1.0 (floor: ≥ 0.55; target 0.70)
    robustness: null              # 0.0–1.0 (floor: ≥ 0.90)
    stability_across_versions: null  # 0.0–1.0 (floor: ≥ 0.75)
    acfr_last_quarter: 0          # count of P1 safety-circuit bypasses
                                  # (floor: 0; Rule 7 trigger: ≥ 1)
    measurement_date: ""
    methodology: ""
    time_series: []

  # ── Section 5: Vendor Transparency ───────────────────
  vendor_transparency:
    questionnaire_date: ""
    questionnaire_version: "1.0"
```

```
  vendor_contact: ""      # Engineering-level contact (not sales)
  responses: []           # 29 responses (see Appendix C)
  section_scores: {A: null, B: null, C: null, D: null}
  overall_score: null
  rule_8_status: ""       # "complete" | "vendor_declined" | "pending" | "nonresponsive"
  outreach_attempts: []   # [{date, method, contact}] (Rule 8 trigger input)
  nonresponsive_since: null # date the 60-day Rule 8 clock expired; null if open

# ── Section 6: Red Team ──────────────────────────
red_team:
  last_engagement_date: ""
  next_scheduled_date: ""
  methodology: ""
  mitre_atlas_coverage: []      # e.g., ["AML.T0051","AML.T0043"]
  findings: []

# ── Section 7: Compensating Controls ─────────────
compensating_controls:
  required_if_band: ["red"]      # mandatory at Red; recommended at Amber
  controls: []            # also required for monosemanticity 0.55–0.70

# ── Section 8: Lifecycle ─────────────────────────
lifecycle:
  deployment_date: ""
  material_updates: []
  planned_retirement_date: null
  rollback_procedure: ""
  incident_history: []
  retention_expires: ""   # Minimum 3 years (Rule 10)
  registry_version: "1.1.0"
  last_reviewed: ""

# ── Computed Fields (derived; not manually entered) ─
crs:
  calculated: null        # Risk Tier × (6 − IMS) × Decision Consequence Weight (DCW)
  band: ""                # "green"|"amber"|"red"|"purple"
  consequence_floor_applied: false  # true if DCW ≥ 4 on High/Critical
  computed_date: ""
  next_review_date: ""
```

Appendix C "Show Me Your Circuits" Vendor Questionnaire

The questionnaire is 29 questions across four sections. Each response is rated Acceptable, Partial, or Unacceptable by the assessing organization's security function, not by the vendor.

Rating	Criteria
ACCEPTABLE	Specific, verifiable answer with artifacts, contract language, or named accountability. Engineering-level sign-off.
PARTIAL	General answer without specifics; deflection to documentation that doesn't address the question; sales-level sign-off.
UNACCEPTABLE	No answer, "trust us," or an answer revealing the vendor has not considered the question.

Benchmark acceptable-response rates by category: Category A vendors average approximately 93% Acceptable; Category B approximately 48%; Category C approximately 14%. *These figures are structural estimates, not results of a published vendor survey. They are derived by asking, for each category, how many of the 29 questions that category can answer acceptably by construction of its access model: a self-hosted open-weights deployer (Category A) can in principle satisfy nearly all 29, since weight and activation access make the underlying evidence producible; an API vendor (Category B) can satisfy roughly the dozen-to-fourteen questions answerable from vendor-supplied artifacts and introspection endpoints but not those requiring weight access; an embedded-AI vendor (Category C) can satisfy only the handful answerable from product documentation and contractual terms, on the order of four of 29, hence the ~14% figure. They indicate the direction and magnitude of the transparency gap rather than measured response rates, and adopters should replace them with their own observed scores as questionnaire responses accumulate.* The gap between these benchmarks is the interpretability transparency gap that CIRCUIT is designed to close.

Section A: Identity and Provenance (Q1–7)

Q1. Please tell us exactly which AI model serves our account, including the model name, version, and approximate size.

Q2. Was this model built by customizing another model (for example, by fine-tuning)? If so, which base model was used, and can you share a summary of the data used to customize it?

Q3. Behind the scenes, is our traffic ever handled by more than one model or provider? If so, which ones, and how do you decide where each request goes?

Q4. How often do you change the model we use, and how much advance notice do we receive before a change takes effect?

Q5. Do you use our data to train, test, or improve your models? Can we prohibit this in our contract?

Q6. How long do you keep our prompts and the model's responses, and how is that data deleted?

Q7. Where does the model actually run (country or region, and hosting provider), and can you describe at a high level how our data flows through your service?

Section B: Interpretability Evidence (Q8–16)

Q8. Have you ever examined how the model works on the inside, rather than only testing its outputs? If so, what reports or documentation came out of that work?

Q9. Do you use tools that translate the model's internal activity into a catalog of human-readable concepts (researchers call these sparse autoencoders, or SAEs)? If so, can customers see that catalog, either openly or under a non-disclosure agreement (NDA)?

Q10. Have you mapped which internal parts of the model are responsible for its safety behaviors, such as refusing harmful requests, protecting personal data, or approving tool use? If so, please describe how, and whether results can be shared.

Q11. Do you measure how cleanly each internal feature of the model maps to one clear human concept (a measure researchers call monosemanticity)? If so, how is that number produced?

Q12. How well do the model's safety controls hold up when someone tries to get around them, for example by rewording a request, hiding instructions in a document, switching languages, or role-playing? Please share your most recent test results.

Q13. When you release a new model version, do you tell customers what changed in its safety behavior compared to the previous version?

Q14. Can we run our own tests against the exact model version we use, in a safe, non-production environment, under NDA?

Q15. Do you give customers any deeper diagnostic access for assurance purposes, such as confidence scores or internal activity data, under NDA?

Q16. Have you published any research about how this model family works internally? Please share links or citations.

Section C: Safety and Red-Teaming (Q17–23)

Q17. Please describe the most recent exercise in which a team deliberately attacked this model to find weaknesses (red teaming): what was tested, how, and what were the main findings?

Q18. If we report a way to get around the model's safety controls, how quickly do you commit to respond, and is that commitment in writing (a service-level agreement, or SLA)?

Q19. Do you organize your attack testing around MITRE ATLAS, the industry catalog of known attack methods against AI systems? If so, what does your testing cover?

Q20. Before a new model version reaches customers, what safety tests must it pass?

Q21. For customers with regulatory obligations, do you offer a higher-assurance mode with more detailed logging, more predictable behavior, or better traceability?

Q22. Have you tested whether attackers can smuggle hidden instructions to the model through the content it reads, including documents, web pages, tool outputs, and user messages (an attack known as prompt injection)? What did you find?

Q23. If a hidden malicious behavior had been planted in the model (a backdoor), how would you detect it? For example, do you monitor the model's internal activity or run special held-back tests?

Section D: Contractual and Audit Posture (Q24–29)

Q24. Will you commit in writing to giving us at least 30 days notice before changing the underlying model?

Q25. Can we audit you ourselves? If so, under what conditions?

Q26. Can we send an independent auditor of our choosing, including a specialist in how AI models work internally?

Q27. If we use your product in a high-risk way under the EU AI Act, what compliance support and documentation do you provide?

Q28. Are you working toward ISO/IEC 42001 certification, the international standard for AI management systems? If so, on what timeline?

Q29. Please name the person, by name and title, in your engineering or security organization who stands behind the accuracy of these answers.

Appendix D Three Worked Registry Examples

D.1: Category A: Open-Weights DLP Classifier (Critical, IMS 4)

```
circuit_registry_entry:
  version: "1.1.0"
  identity:
    model_id: "dlp-classifier-v3"
    name: "Internal DLP Content Classifier v3 (Llama 3.1 70B fine-tune)"
    vendor: "internal"
    category: "A"
    risk_tier: "Critical"          # Numeric: 4
    owner: "security-team@company.com"
    consequence: "Automated"        # DCW: 3
    data_classification: "PII, financial"
  maturity:
    ims: 4
    ims_ceiling: 5
    evidence:
      - {artifact: "dlp_attribution_graph_v3_2026-03-15.pkl",
         type: attribution_graph, date: "2026-03-15"}
      - {artifact: "dlp_sae_llama31_70b_v3.pkl",
         type: sparse_autoencoder, date: "2026-02-20"}
  kpi_baseline:
    circuit_size: 84               # ≤ 100 ✓
    edge_count: 318                # ≤ 500 ✓
    monosemanticity: 0.87          # ≥ 0.70 ✓ (above target)
    robustness: 0.94               # ≥ 0.90 ✓
    stability_across_versions: 0.91 # ≥ 0.75 ✓
    acfr_last_quarter: 0           # floor 0 ✓
    measurement_date: "2026-03-15"
  crs:
    calculated: 24                 # 4 × (6−4) × 3 = 24
    band: "amber"                  # DCW 3 → consequence floor N/A
    consequence_floor_applied: false
    next_review_date: "2026-06-15"  # Quarterly (Amber)
```

D.2: Category B: API Foundation Model for SOC Triage (High, IMS 2)

```
circuit_registry_entry:
  version: "1.1.0"
  identity:
    model_id: "soc-triage-llm-v1"
    name: "SOC Alert Triage Assistant (Claude Sonnet API)"
    vendor: "Anthropic"
    category: "B"
    risk_tier: "High"              # Numeric: 3
    owner: "soc-lead@company.com"
    consequence: "Advisory"        # DCW: 1 (analyst approves all actions)
  maturity:
    ims: 2
    ims_ceiling: 3                 # Category B ceiling
```

```
    evidence:
      - {artifact: "soc_shap_report_2026-04-01.html",
         type: shap_values, date: "2026-04-01"}
  kpi_baseline:
    circuit_size: null              # Not available at IMS 2 (Category B)
    robustness: null                # Circuit robustness unavailable below IMS 3 (Category B)
    acfr_last_quarter: 0
    measurement_date: "2026-04-01"
  vendor_transparency:
    rule_8_status: "complete"
    overall_score: 0.48             # 48% Acceptable (Category B benchmark)
  crs:
    calculated: 12                  # 3 × (6−2) × 1 = 12
    band: "green"                   # DCW 1 → consequence floor N/A
    consequence_floor_applied: false
```

D.3: Category C: Embedded Vendor AI Repositioned to Advisory (Critical, IMS 1)

```
circuit_registry_entry:
  version: "1.1.0"
  identity:
    model_id: "m365-copilot-remediation-v1"
    name: "Microsoft 365 Copilot: Security Remediation Advisory"
    vendor: "Microsoft"
    category: "C"
    risk_tier: "Critical"           # Numeric: 4
    owner: "infosec@company.com"
    consequence: "Advisory"         # DCW: 1; REPOSITIONED from Catastrophic (5)
  # Original deployment (DCW 5, Catastrophic) → CRS 100 (Purple band, not deployable).
  # Redesigned to Advisory-only per CIRCUIT assessment. CRS now 20 (Amber).
  # Rule 9 check: Category C + Critical + Advisory = NOT a Rule 9 violation.
  # Consequence floor: DCW 1 < 4, so floor does NOT apply.
  maturity:
    ims: 1
    ims_ceiling: 2                  # Category C ceiling
    evidence:
      - {artifact: "m365_behavioral_log_2026-04-15.json",
         type: behavioral_log, date: "2026-04-15"}
  vendor_transparency:
    questionnaire_date: "2026-03-01"
    overall_score: 0.14             # 14% Acceptable (Category C benchmark)
    rule_8_status: "complete"
  compensating_controls:
    controls:
      - {description: "All Copilot-generated scripts require human engineer
          approval before execution", type: human_in_the_loop, covers_rule: 9}
  crs:
    calculated: 20                  # 4 × (6−1) × 1 = 20
    band: "amber"
    consequence_floor_applied: false
    next_review_date: "2026-07-15"
```

Appendix E References

Primary Research

- Ameisen, E., Lindsey, J., et al. (2025). Circuit Tracing: Revealing Computational Graphs in Language Models. transformer-circuits.pub.
- Anthropic Interpretability Team. (2025). On the Biology of a Large Language Model. transformer-circuits.pub.
- Anthropic. (2025). Open-sourcing circuit-tracing tools. anthropic.com.
- Gao, L., et al. (2025). Weight-Sparse Transformers Have Interpretable Circuits. arXiv:2511.13653 [cs.LG].
- Google DeepMind. (2025). Gemma Scope 2: Open Sparse Autoencoders Across All Layers of Gemma 3.
- Hanna, M., et al. (2025). circuit-tracer: A New Library for Finding Feature Circuits. BlackboxNLP 2025.
- Hubinger, E., et al. (2024). Sleeper Agents: Training Deceptive LLMs That Persist Through Safety Training. arXiv:2401.05566.
- Lieberum, T., et al. (2024). Gemma Scope: Open Sparse Autoencoders Everywhere All at Once on Gemma 2. arXiv:2408.05147. Successor release: Google DeepMind (2025), Gemma Scope 2, open sparse autoencoders for the Gemma 3 model family.
- Marks, S., Tegmark, M. (2023). The Geometry of Truth. arXiv:2310.06824.
- Meng, K., et al. (2022). Locating and Editing Factual Associations in GPT. NeurIPS 2022. arXiv:2202.05262.
- Nanda, N., et al. (2023). Progress Measures for Grokking via Mechanistic Interpretability. ICLR 2023. arXiv:2301.05217.
- Templeton, A., et al. (2024). Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. transformer-circuits.pub.
- Yang, Z., et al. (2026). CrossMPI: Cross-Modal Prompt Injection via Image-Only Attacks on LVLMs. arXiv:2605.16090 [cs.CR]. 15 May 2026.

Regulatory and Standards

- EU. Regulation (EU) 2024/1689 (the EU AI Act). Articles 9, 11, 13, 14, 15, 50.
- NIST. AI Risk Management Framework (AI RMF 1.0). NIST AI 100-1. January 2023.
- NIST. Generative AI Profile. NIST AI 600-1.
- ISO/IEC. ISO/IEC 42001:2023, AI Management System. Annex A.6.2.
- Federal Reserve Board and OCC. SR 11-7: Supervisory Guidance on Model Risk Management. April 2011.
- AICPA. SOC 2 Trust Services Criteria. 2017 (with revisions).
- Cloud Security Alliance. AI Controls Matrix (AICM).
- MITRE. ATLAS: Adversarial Threat Landscape for AI Systems. atlas.mitre.org.
- OWASP. Top 10 for Large Language Model Applications. owasp.org.
- Gartner. AI Trust, Risk and Security Management (AI TRiSM).
- Forrester Research. AI Governance, Risk and Ethics Intelligence Service (AEGIS).

Scoring and Risk-Assessment Methodologies

FIRST. Common Vulnerability Scoring System (CVSS) v3.1 / v4.0 Specification. first.org/cvss.

FAIR Institute. Factor Analysis of Information Risk (FAIR), a quantitative model for information risk. fairinstitute.org.

Meier, J.D., et al. (2003). Improving Web Application Security: Threats and Countermeasures (the DREAD risk model). Microsoft Press. Referenced as a cautionary precedent; DREAD fell out of use in Microsoft's guidance owing to the subjectivity of its severity ratings.

Appendix F Acronym Glossary

Acronym	Expansion
ACFR	Adversarial Circuit Failure Rate (a count per model-quarter)
AIGC	AI Governance Committee
AICM	AI Controls Matrix (CSA)
AIMS	AI Management System (ISO/IEC 42001)
ATLAS	Adversarial Threat Landscape for AI Systems (MITRE)
CI/CD	Continuous Integration / Continuous Deployment
CIRCUIT	Circuit-Informed Risk & Control, Understanding, Inventory & Transparency
CISO	Chief Information Security Officer
CRS	Circuit Risk Score
CSA	Cloud Security Alliance
DCW	Decision Consequence Weight
DLP	Data Loss Prevention
EU AI Act	Regulation (EU) 2024/1689
IMS	Interpretability Maturity Score
LIME	Local Interpretable Model-agnostic Explanations
LoRA	Low-Rank Adaptation
LVLM	Large Vision-Language Model
MI	Mechanistic Interpretability
MLP	Multi-Layer Perceptron
NIST	National Institute of Standards and Technology
NDA	Non-Disclosure Agreement
OCC	Office of the Comptroller of the Currency
OWASP	Open Worldwide Application Security Project
PII	Personally Identifiable Information
RFP	Request for Proposal
RLHF	Reinforcement Learning from Human Feedback
SAE	Sparse Autoencoder
SHAP	SHapley Additive exPlanations
SLA	Service Level Agreement
SOC	Security Operations Center / System and Organization Controls

Acronym	Expansion
SR 11-7	Fed/OCC Supervisory Guidance on Model Risk Management
TSC	Trust Services Criteria (SOC 2)

Appendix G The Mathematics Behind the Score

Formally, the CRS is a multiplicative risk functional over a discrete domain. Let $R \in \{1, 2, 3, 4\}$ denote the Risk Tier, $I \in \{0, \ldots, 5\}$ the Interpretability Maturity Score, and $W \in \{1, \ldots, 5\}$ the Decision Consequence Weight. Define the interpretability deficit $D = 6 - I$, so that $D \in \{1, \ldots, 6\}$, and $CRS(R, I, W) = R \cdot D \cdot W$. Four structural properties of this functional carry the framework's security semantics, and each is a design decision rather than a notational convenience. First, strict monotonicity: the score is strictly increasing in R and W and strictly decreasing in I, with discrete marginal $\Delta CRS/\Delta I = -R \cdot W < 0$ everywhere on the domain. Every unit of interpretability evidence lowers the score; every escalation of stakes or autonomy raises it; there are no plateaus and no reversals an adversarial scorer could exploit.

Second, superlinear coupling between evidence and stakes. The mixed discrete difference $\Delta^2 CRS/\Delta I \Delta W = -R$ is negative and scales with the Risk Tier, which means the marginal value of one IMS level is not a constant: it grows with the consequence of the decisions the model is permitted to make. Climbing from IMS 2 to IMS 3 saves three points on a Low-tier advisory tool and twenty points on a Critical-tier irreversible workflow. No additive aggregation can express this; under any weighted sum $aR + bI + cW$, the marginal effect of interpretability is the fixed coefficient b, indifferent to what is at stake. The multiplicative form encodes the central security intuition directly: visibility matters most exactly where a mistake costs most. Third, bounded compensability. Additive models permit full compensation between factors; a catastrophic consequence can be averaged away by favorable scores elsewhere, which is precisely the failure mode that makes naïve risk matrices gameable. Under multiplication, compensation operates on ratios rather than differences and is bounded by the factor domains: the most favorable deficit available ($D = 1$, full continuous interpretability) shrinks the score by at most a factor of six and never to zero, so a Critical-tier Catastrophic configuration retains $CRS \geq 20$, Amber at best, under perfect interpretability. The worst case, $CRS(4, 0, 5) = 4 \times 6 \times 5 = 120$, is not an artifact of the scale but its point: no process maturity elsewhere in the portfolio rescues a fully opaque model making catastrophic autonomous decisions. Fourth, the image of the functional is sparse by construction, and the band boundaries exploit that sparseness. The reachable set contains exactly thirty-four values in [1, 120], thinning toward the high end; the transitions between occupied ranges cross unreachable values (13–14 between Green and Amber, 46–47 between Amber and Red, 97–99 between Red and Purple), so band assignment is total and boundary-free: no deployment can sit on a threshold, and no audit dispute about rounding or tie-breaking can arise. A final caveat completes the treatment honestly: R, I, and W are ordinal scales, not cardinal measurements, and their product is therefore a prioritization index, not a quantity of risk. CRS 48 does not denote twice the risk of CRS 24; it denotes a configuration warranting strictly more scrutiny under a consistent, order-preserving rule. The properties claimed above, monotonicity, coupled marginals, bounded compensability, and boundary-free banding, are exactly the properties that survive ordinal interpretation, and the framework deliberately claims nothing that does not.

The practical implication is that the CRS has two levers: increasing IMS (reducing the interpretability-deficit term) and decreasing DCW (repositioning the workflow). Both are actionable independent of vendor cooperation. The consequence floor in Rule 3 is the deliberate complement to multiplication: multiplication lets a single extreme dimension dominate the score, and the floor guarantees that maximum-consequence decisions retain a hard governance minimum even when the interpretability term is at its most favorable. Visibility into a mechanism is a precondition for control, not a substitute for it; a perfectly understood model can still make an irreversible mistake, and the band floor encodes that truth.

Why the deficit term is (6 − IMS) and not (5 − IMS). The interpretability-deficit factor is intentionally one-based rather than zero-based. With (6 − IMS), a continuously interpretable model at IMS 5 still contributes a factor of 1 rather than 0, so the CRS of a fully mature, low-consequence model is its Risk Tier × DCW product

rather than zero. This encodes a substantive claim: interpretability maturity reduces risk but never eliminates it. A model whose circuits are perfectly understood can still be deployed in a context whose consequence and tier carry irreducible risk, and the score should never collapse to zero on the strength of interpretability alone. A zero-based term would imply that sufficient interpretability makes a deployment risk-free, which is false and which the consequence floor would then have to contradict.

Relationship to existing scoring systems. A security architect will recognize the shape of the CRS (a small number of ordinal factors combined into a banded severity score) and will reasonably compare it to CVSS, FAIR, and DREAD. The comparison is worth making explicit. CVSS demonstrates both the value and the hazard of decomposed multiplicative scoring: it is widely adopted precisely because a single reproducible number routes attention, but it is also criticized for false precision and for base scores that compress dissimilar conditions into the same value. CIRCUIT inherits the routing benefit deliberately and bounds the false-precision hazard by keeping the scale coarse and explicitly labeling it as a prioritization instrument rather than a measurement. FAIR quantifies risk in probabilistic loss terms and is complementary, not competing: an organization can express the consequence dimension (DCW) in FAIR terms if it wishes, and the CRS sits upstream as a control-gating score rather than a loss estimate. DREAD is the cautionary case. It was abandoned in practice because its additive-and-multiplicative severity factors were subjective and trivially gamed, producing scores that did not survive scrutiny. CIRCUIT departs from DREAD in two specific ways: every IMS level is tied to a named, on-file evidence artifact rather than a reviewer's subjective rating (Rule 2), and the consequence floor removes the single most dangerous degree of freedom, the ability of a favorable rating on one axis to rescue a maximum-consequence deployment. The CRS is not a reinvention of these systems; it is a control-gating score for a dimension (interpretability evidence) that none of them measures.