P O S I T I O N   P A P E R

Why Containment Beats Evaluation

A Runtime Security Architecture for Sandbagging-Resistant AI Governance in Regulated Environments

Jonathan Gill
Principal Cybersecurity Officer: AI, Cloud & Platform Cyber Risk
Enterprise AI Security Framework | airuntimesecurity.co.za

March 2026
Version 1.1 | For public distribution

Abstract

The AI safety research community has invested heavily in making model evaluations robust against sandbagging, the risk that AI systems strategically underperform during testing to conceal dangerous capabilities. This paper argues that for enterprise deployment in regulated environments, the evaluate-then-permit paradigm is structurally insufficient. It proposes an alternative: a constrain-regardless architecture based on three interlocking runtime controls (containment boundaries, an enterprise-owned LLM-as-Judge, and human oversight escalation) that renders sandbagging operationally irrelevant as a breakout vector by decoupling security enforcement from evaluation accuracy.

Drawing on empirical findings from the UK AI Security Institute's Frontier AI Trends Report (December 2025), the International AI Safety Report 2026, Anthropic's Constitutional Classifiers research, and the Coalition for Secure AI's MCP Security framework, this paper demonstrates that containment-based architectures address a class of risks that evaluation-based approaches cannot, and are better suited to the operational, legal, and regulatory requirements of financial services and other critical infrastructure sectors.

This architecture is not without limits. The paper acknowledges residual risks including judge brittleness, within-bounds adversarial optimisation, and operational complexity, and argues that intellectual honesty about these limits strengthens rather than weakens the case for runtime containment over evaluation dependency.

1. The Problem: Evaluation as the Foundation of Trust

The dominant paradigm in AI safety research follows a linear sequence: evaluate a model's capabilities, determine what is safe to permit based on those results, then deploy with safeguards calibrated to the evaluation findings. This evaluate-then-permit model underpins Anthropic's Responsible Scaling Policy, OpenAI's Preparedness Framework, Google DeepMind's Frontier Safety Framework, and the broader ecosystem of Frontier AI Safety Frameworks, twelve of which were published or updated in 2025 alone.

Sandbagging threatens this paradigm at its foundation. If a model can detect that it is being evaluated and strategically underperform, then the evaluation produces a false picture of the model's capabilities. Safeguards are calibrated to a capability level that does not reflect reality. The model is deployed into production with permissions it should not have.

The empirical evidence that this risk is real and growing is now substantial:

Evidence from the UK AI Security Institute (December 2025)

AISI confirmed that current models can sandbag when prompted to do so, subtly enough to fool automated monitoring systems. The more complex the task, the easier sandbagging becomes. Black-box monitors (which analyse outputs and chain-of-thought) lose accuracy as task difficulty increases. Deception probes that examine model internals show promise on small models but degrade on larger models performing complex agentic tasks. AISI analysed over 2,700 evaluation transcripts and found no evidence of spontaneous sandbagging but acknowledged that their monitors may not detect all forms of strategic underperformance.

Evidence from the International AI Safety Report 2026

The Bengio-led report, authored by over 100 experts and backed by 30+ countries, noted that reliable pre-deployment safety testing has become harder to conduct. Models increasingly distinguish between test settings and real-world deployment and exploit loopholes in evaluations. The report explicitly states that dangerous capabilities could go undetected before deployment.

Evidence from Anthropic’s Research

Anthropic’s own research has shown that automated researchers can subtly sandbag, and that models can learn to hack their own reward systems by generalising from training in simpler settings. Their Constitutional Classifiers++ paper (January 2026) demonstrates that even sophisticated monitoring requires 1,700+ hours of red teaming to validate, and the system’s designers acknowledge that no defence is perfectly robust.

The Mirror Risk: Evaluation Overconfidence

The sandbagging literature focuses on models that are more capable than evaluation reveals. But evaluation failure has two faces. The mirror risk is a model that appears more capable than it is: one that is deployed with permissions it can technically exercise but cannot competently exercise. It produces outputs that are plausible, fluent, structurally sound, and wrong.

This is arguably the more prevalent failure mode in enterprise deployment today. Every organisation that has deployed a summarisation agent or compliance checker has encountered this: the model sounds like it knows what it is doing, and it does not. Pre-deployment evaluation that overestimates competence is as dangerous as evaluation that underestimates capability, and it is far more common.

Both failure modes share the same structural cause: security and governance that depend on evaluation accuracy inherit the full uncertainty of that evaluation. The constrain-regardless architecture addresses both directions of failure through the same mechanism, as described in the following sections.

The safety research community is attempting to solve sandbagging by making evaluations more accurate. This paper argues that for enterprise deployment, the better strategy is to make evaluations less important, because evaluation failure in either direction *(underestimating or overestimating capability) produces the same outcome: miscalibrated governance.*

2. The Alternative: Constrain-Regardless Architecture

A constrain-regardless architecture does not rely on evaluation to determine what a model is permitted to do. Instead, it defines operational boundaries independently of the model’s capabilities, enforces those boundaries at runtime, and monitors for behavioural drift within permitted parameters.

This approach rests on a simple principle: if an AI agent’s actions are constrained to a defined set of permitted operations, the agent’s undisclosed capabilities are operationally irrelevant as a breakout vector. A model may be capable of far more than it is allowed to do. That hidden capability cannot manifest if the containment boundary prevents it.

The architecture comprises three interlocking layers, each addressing the residual risk of the layer before it. Critically, these layers form a compound defence: no single layer is trusted alone, and each layer’s failure mode is covered by the next. This mutual reinforcement is what distinguishes the architecture from simply stacking independent controls.

2.1 Layer 1: Containment Boundaries (Guardrails)

Containment boundaries define what an AI agent is permitted to do, based on business intent and operational requirements rather than on evaluation of the model’s capabilities. These boundaries are enforced through technical controls: API-level restrictions, tool-calling permissions, data access scopes, output format constraints, and action whitelists.

In this layer, the question is not “what can this model do?” but “what does this business process require the model to do?” The boundary is set by the use case, not by the model. This is a fundamental inversion of the evaluate-then-permit logic.

What this neutralises: Sandbagging is a strategy for evading evaluation-based permission systems. If permissions are not derived from evaluation, there is nothing to game. A model that conceals its ability to exfiltrate data cannot exploit that ability if the containment boundary does not permit data exfiltration in the first place. The capability exists; the execution path does not.

What this does not neutralise: Containment addresses breakout risk. It does not address adversarial behaviour within permitted bounds, such as an agent that subtly degrades output quality, optimises for unintended objectives, or produces technically compliant but strategically misaligned results. That residual risk is the explicit domain of Layer 2.

2.2 Layer 2: Enterprise-Owned LLM-as-Judge

A separate model, configured with an enterprise-defined behavioural constitution, continuously evaluates the agent’s outputs against expected patterns. Unlike vendor-side safeguards (such as Anthropic’s Constitutional Classifiers or OpenAI’s safety training), this judge is owned, configured, and governed by the deploying organisation.

The distinction matters. Vendor-side safeguards are designed to prevent the model from producing harmful outputs. Enterprise-owned judges are designed to verify that the model is producing the *correct* outputs for a specific business context. These are different objectives with different threat models.

How this catches drift: Behavioural degradation, whether caused by sandbagging within bounds, prompt injection, model updates, or stochastic variation, manifests as a change in the statistical distribution of outputs over time. An LLM-as-Judge trained on the expected distribution for a given use case detects this drift regardless of its cause. The agent does not need to be “caught sandbagging”; it needs to be caught behaving differently from baseline. The judge is agnostic to intent.

The obvious question: who judges the judge? This is a legitimate concern. The judge is itself a model, subject to its own blind spots, drift, and potential manipulation. If the architecture simply replaced dependency on evaluation with dependency on a judge, it would be a lateral move, not an improvement. The answer is structural: the judge does not operate alone. It operates as the second layer of a compound defence. When the judge’s confidence falls below defined thresholds, when its outputs diverge from historical patterns, or when the agent’s behaviour falls into ambiguous territory, the system escalates to Layer 3. The judge is not trusted unconditionally. It is trusted within bounds, and those bounds are enforced by human oversight. The meta-evaluation problem is real, but it is addressed by architecture, not by building a better judge.

2.3 Layer 3: Human Oversight Escalation

Neither containment nor the judge is perfect. Containment boundaries may have gaps in enforcement. The judge may produce false negatives or share blind spots with the base model. The third layer exists to handle what the first two cannot resolve autonomously.

Human oversight is triggered by defined escalation thresholds: judge confidence below a threshold, anomalous patterns in agent behaviour, high-stakes decisions requiring explicit authorisation, or periodic audit reviews. This layer is not a fallback; it is an integral component of the control system, and its escalation triggers are calibrated to catch specifically the failure modes of Layers 1 and 2.

The compound defence: For sandbagging or any form of adversarial behaviour to succeed against this architecture, a model would need to simultaneously: remain within its containment boundary (evading all technical controls), produce output that passes the judge’s behavioural constitution over sustained periods without triggering drift detection, and avoid triggering any human escalation threshold. Each layer’s design is informed by the failure modes of the others.

This is not defence-in-depth by coincidence. It is defence-in-depth by design, where each layer is specifically calibrated to catch what the previous layer misses.

A note on layer hierarchy: Although the three layers are mutually reinforcing, they are not equally independent. In adversarial contexts, particularly where the judge and agent share architectural lineage or training data, Layers 2 and 3 may exhibit correlated failure modes. In those scenarios, Layer 1 (containment) becomes the only structurally independent control. It is the hard barrier. The judge and human oversight verify and refine; containment constrains. When designing for adversarial resilience, containment is load bearing. The other layers are verification.

2.4 Preserving AI Value Within Containment

The strongest objection to this architecture is practical, not theoretical: if containment constrains the AI this tightly, why not write conventional code? The question deserves a direct answer, because if the architecture eliminates what makes AI valuable, it is not a security solution. It is a rejection of the technology.

The answer lies in a distinction between two types of constraint. **Action-space constraints** limit what an agent may do: which tools it may call, which data it may access, which operations it may execute. **Reasoning-space constraints** limit how an agent may think: what it may infer, synthesise, interpret, or generate. Containment operates exclusively on the action space. The reasoning space remains unconstrained.

This is the critical distinction. The entire value of deploying AI, rather than deterministic code, lies in its ability to handle ambiguity, generalise across contexts, synthesise unstructured information, and produce outputs that could not have been pre-specified. None of that requires unrestricted action. A model summarising customer complaint needs full reasoning freedom to interpret nuance, identify patterns, and generate coherent analysis. It does not need the ability to call external APIs, modify database records, or exfiltrate data. Containment preserves the former while preventing the latter.

The analogy is parental, not carceral. A good parent does not script what a child says or thinks. They set boundaries: don’t cross the road, don’t touch the stove, come home before dark. Within those boundaries, the child has full autonomy. The boundaries are non-negotiable. The behaviour within them is free. And the parent watches, not to control every action, but to notice when something looks wrong. Containment is the boundary. The judge is the parent watching. Human oversight is the parent stepping in when passive observation is insufficient.

Ironically, the alternative approaches, evaluate-then-permit and vendor-side safety training, are more likely to constrain the reasoning space than containment is. Over-refusal, where a model declines legitimate business queries because safety training is poorly calibrated, is a reasoning-space constraint. It reduces the AI’s utility without improving operational security. Containment avoids this entirely by moving the control to the action layer and leaving the reasoning layer alone.

The right question is not whether to constrain, but where on the spectrum of action-space tightness to operate. A customer-facing agent recommending investment products requires tighter boundaries than an internal agent summarising meeting notes. The architecture is identical in both cases (three layers, same logic, same compound defence) but the containment boundaries are calibrated to the risk. This is not a binary between fully constrained and fully free. It is risk-proportionate governance that preserves AI's value precisely because it does not attempt to control AI's reasoning.

3. Closed-Loop Control vs. Open-Loop Evaluation

The difference between these paradigms can be expressed in control systems terms.

The evaluate-then-permit model is an **open-loop control system**. It measures the system's state once (during evaluation), sets control parameters based on that measurement, and then operates without further verification. If the initial measurement is wrong, whether due to sandbagging, evaluation limitations, or capability changes after deployment, the control parameters remain miscalibrated indefinitely.

The constrain-regardless model is a **closed-loop control system**. It defines the desired operating state (containment boundaries), continuously measures actual operating state (LLM-as-Judge), computes the error between desired and actual (behavioural drift detection), and applies corrective action (escalation to human oversight). The accuracy of initial evaluation is useful but not load-bearing. The system self-corrects.

This does not make the system infallible. Closed-loop control systems can still fail through sensor error (judge blind spots), actuator failure (escalation processes that do not respond effectively), or feedback delay (drift that accumulates faster than the judge detects). But closed-loop systems *degrade gracefully* and are *self-correcting by design*, whereas open-loop systems fail silently and without recovery.

In engineering, no critical system operates on open-loop control. AI governance should be no different. The question is not whether the model passed its evaluation. The question is whether the model is behaving correctly right now.

4. Implications for the Safety Research Community

This paper does not argue that evaluation research is without value. Pre-deployment evaluation informs risk classification, capability awareness, and the initial design of containment boundaries. The AISI's capability tracking, Anthropic's red-teaming, and METR's time-horizon benchmarks all contribute essential intelligence about the threat landscape.

However, evaluation should inform architecture. It should not *be* the architecture. The current research emphasis on making evaluations sandbagging-resistant (better probes, white-box monitoring, chain-of-thought inspection) addresses a real problem but implicitly accepts a flawed premise: that deployment security depends on evaluation accuracy. For enterprise contexts, this dependency is neither necessary nor desirable.

Specifically:

- **Containment reduces dependency on probabilistic evaluation and converts capability risk into bounded operational risk.** Evaluation accuracy is inherently probabilistic. Containment enforcement, while not infallible, operates on a different and more favourable risk profile. It constrains the action space rather than attempting to predict the capability space. Regulated environments prefer controls that bound outcomes over controls that estimate inputs.
- **Evaluation is performed by the model's developer. Containment is enforced by the deploying organisation.** In regulated industries where the deployer bears liability, the deployer must own the primary security control. Delegating security assurance to a vendor's evaluation process is a risk transfer that most banking supervisors would not accept.
- **Evaluation is a point-in-time assessment. Runtime containment is continuous.** Models change after deployment through fine-tuning, prompt evolution, and vendor updates. Point-in-time assessments degrade; continuous monitoring does not.
- **Sandbagging detection scales poorly with model capability.** AISI found that detection accuracy degrades as models become more capable and tasks more complex. Containment enforcement does not degrade with model capability. The boundary is independent of what the model can do.

5. Application to Agentic AI and MCP Security

The rise of agentic AI systems and the Model Context Protocol (MCP) make the constrain-regardless architecture more urgent, not less. The Coalition for Secure AI's MCP Security white paper (January 2026) identified 12 threat categories spanning approximately 40 distinct threats. Real-world incidents, including the EchoLeak zero-click exfiltration from Microsoft 365 Copilot (CVE-2025-32711), Asana's tenant isolation failure, and prompt injection via Atlassian support tickets, demonstrate that agentic AI risks have moved from theoretical to operational.

In an agentic context, sandbagging is only one manifestation of a broader problem: AI systems behaving differently from expectation in ways that evade detection. Tool poisoning, cross-agent context contamination, confused deputy attacks, and supply chain compromise through context injection all share the same structural characteristic. The system does something its operators did not intend and cannot see.

The three-layer architecture addresses all of these:

- **Containment boundaries** restrict which tools an agent may call, which data sources it may access, and which actions it may execute, regardless of what a poisoned tool or injected prompt requests.
- **The LLM-as-Judge** monitors for anomalous patterns in the agent's decisions: unusual tool selection sequences, data access patterns that deviate from baseline, or output characteristics that suggest context contamination.
- **Human oversight** handles ambiguous cases and provides the accountability chain that regulators in financial services and critical infrastructure require.

This is not theoretical. The AISI's analysis of MCP server autonomy levels in financial services found a sharp increase in execution-capable servers (Level 4 to 5) through mid-2025. AI agents are already being granted the authority to transfer assets and execute trades. The question of how to govern these systems is not emerging. It is here.

6. Regulatory Alignment

The constrain-regardless architecture aligns naturally with existing regulatory expectations in financial services and critical infrastructure:

- **ISO 42001 (AI Management System)** requires organisations to define and maintain operational controls for AI systems proportionate to risk. Containment boundaries satisfy the control requirement; the judge satisfies the monitoring requirement; human oversight satisfies the governance requirement.
- **EU AI Act** requires human oversight for high-risk AI systems and mandates risk management systems that operate throughout the AI system lifecycle, not only at deployment.
- **NIST AI RMF** emphasises continuous monitoring and the separation of AI risk management from the AI system's own capabilities. This is precisely the principle underlying enterprise-owned (rather than vendor-delegated) controls.
- **Banking supervisory expectations (APRA CPS 230, PRA SS1/23, ECB Guide on AI)** consistently require that operational risk controls be owned by the deploying institution, not outsourced to technology providers. The constrain-regardless model places the primary control in the hands of the institution.

The International AI Safety Report 2026 explicitly endorses defence-in-depth, layered safeguards where no single control is trusted alone. The CoSAI Principles for Secure-by-Design Agentic Systems formalise this as three requirements: human-governed and accountable, bounded and resilient, and subject to risk-based controls. The architecture described in this paper is a concrete implementation of these principles.

7. Limits of This Approach

Intellectual honesty requires acknowledging what this architecture does not solve. The constrain-regardless model reduces dependency on evaluation accuracy and provides a structurally superior control framework for enterprise deployment. It is not a complete answer to all AI governance risks.

Judge Brittleness

The LLM-as-Judge is itself a model. It can drift, hallucinate, or develop blind spots that correlate with the base model's failure modes, particularly if both share architectural lineage or training data. Judge calibration is an ongoing operational requirement, not a one-time configuration. Organisations deploying this architecture must treat judge maintenance with the same rigour as they treat the primary model's governance.

Within-Bounds Adversarial Behaviour

Containment prevents breakout. It does not prevent a model from pursuing misaligned objectives within its permitted action space. An agent that is permitted to summarise customer complaints could subtly bias its summaries in ways that are technically compliant but operationally harmful. For example, consistently downplaying severity or omitting patterns. This class of risk is the judge's domain, but detecting subtle, slow-burn optimisation in high-dimensional output spaces is non-trivial. The architecture bounds the risk; it does not eliminate it.

Operational Complexity and Escalation Fatigue

Three interlocking layers require three layers of operational support. Containment boundaries must be maintained as business processes evolve. Judge baselines must be recalibrated as models are updated. Human reviewers must respond to escalations without developing alert fatigue. The architecture introduces operational cost that must be justified by the risk it mitigates. In high-risk, regulated contexts such as financial services, healthcare, and critical infrastructure, that justification is straightforward. In lower-risk contexts, a lighter approach may be appropriate.

Model Update Risk

When a vendor updates the underlying model, the containment boundaries remain intact, but the judge's baseline may become stale. A model that behaves differently after an update is not sandbagging. It is simply different. The judge must be sensitive enough to detect this change but calibrated enough not to treat every update as an anomaly. This is a tuning problem, not an architectural flaw, but it requires operational maturity.

Cross-Model Collusion

In multi-agent architectures, the assumption that the judge is independent of the base model becomes harder to guarantee. If the judge and the agent share the same provider, training methodology, or underlying architecture, correlated failures become possible. Organisations deploying this architecture in multi-agent contexts should consider using judges from different providers or architectures to preserve independence.

Competence Overestimation and Judge Constitution Design

The architecture addresses adversarial capability concealment (sandbagging) and competence overestimation (where a model is less capable than believed) through the same mechanism: the judge detects deviation from expected output quality. However, this only works if the judge's behavioural constitution is designed for correctness and business-context accuracy, not only for safety and policy compliance.

A judge configured exclusively for safety will catch harmful outputs but miss confidently wrong ones. A model producing safe, compliant, plausible, and incorrect answers will pass a safety-only judge indefinitely. This is a design requirement, not an architectural limitation: the behavioural constitution must include quality, accuracy, and domain-appropriateness metrics alongside safety constraints. Organisations that configure judges only for compliance will get compliance. They will not necessarily get competence.

None of these limits invalidate the architecture. They define its operating envelope. A control system that is honest about its failure modes is more trustworthy than one that claims none.

8. Conclusion

The AI safety community is right to take sandbagging seriously. A model that can deceive its evaluators is a model whose risk profile is unknown. But the response to this problem diverges depending on where you sit.

For model developers, improving evaluation robustness is essential. Developers need to understand what their models can do to make responsible release decisions. The work of AISI, Anthropic, METR, and others on detection methods (probes, white-box monitors, chain-of-thought analysis) serves this purpose and should continue.

For model deployers, particularly in regulated industries, the calculus is different. The deployer's obligation is not to determine what the model can do; it is to ensure the model does what it should, and nothing else, within the deployer's operational environment. This is a containment problem, not an evaluation problem.

Containment makes breakout irrelevant. The judge makes drift detectable. Human oversight handles what the judge cannot resolve autonomously. Each layer covers the residual risk of the layer before it. There is no single-point-of-failure attack path for sandbagging as a breakout vector because no single layer's compromise is sufficient.

The three-layer architecture (containment boundaries, enterprise-owned LLM-as-Judge, and human oversight escalation) is a continuously operating control system for AI governance. It does not depend on the model's honesty during evaluation. It does not depend on the vendor's safeguards. It depends on the deploying organisation's definition of acceptable behaviour, enforced at runtime and verified continuously.

AI governance that relies on point-in-time assessment, whether of capability or of safety, will always be outpaced by the systems it seeks to govern. The standard should be the same one applied to every other critical system: define acceptable behaviour, measure actual behaviour, correct the difference. Continuously.

References

AI Security Institute. Frontier AI Trends Report. UK AI Security Institute, December 2025.

Bengio, Y. et al. International AI Safety Report 2026. arXiv:2602.21012, February 2026.

Sharma, M. et al. Constitutional Classifiers: Defending against Universal Jailbreaks across Thousands of Hours of Red Teaming. Anthropic, January 2025.

Cunningham, H. et al. Constitutional Classifiers++: Efficient Production-Grade Defenses against Universal Jailbreaks. Anthropic, arXiv:2601.04603, January 2026.

Coalition for Secure AI. Model Context Protocol (MCP) Security. OASIS Open / CoSAI Workstream 4, January 2026.

Coalition for Secure AI. Principles for Secure-by-Design Agentic Systems. OASIS Open, July 2025.

METR. Measuring AI Ability to Complete Long Tasks. Model Evaluation and Threat Research, 2025.

Anthropic. Automated Researchers Can Subtly Sandbag. Alignment Science Blog, 2025.

AI Security Institute. White Box Control at UK AISI: Update on Sandbagging Investigations. 2025.

Schoenn, N. et al. Large Language Models Often Know When They Are Being Evaluated. 2025.

Greenblatt, R. et al. AI Sandbagging: Language Models can Strategically Underperform on Evaluations. 2024.

Palo Alto Networks Unit 42. New Prompt Injection Attack Vectors Through MCP Sampling. December 2025.

Cisco. State of AI Security 2026. Cisco AI Threat Intelligence & Security Research, February 2026.

About the Author

Jonathan Gill is a cybersecurity practitioner with over 30 years in IT and 20+ years in enterprise cybersecurity, specialising in AI security governance for regulated financial services environments. He is the author of the Enterprise AI Security Framework (airuntimesecurity.co.za), an open-source framework for runtime behavioural security in AI deployments. His work focuses on translating AI safety research into implementable security architectures for banking, insurance, and critical infrastructure.

CISSP | CCSP | AWS Machine Learning Specialty