Lakera Guards Inputs Without Governing System Confidence

by Nick Clark | Published March 28, 2026 | PDF

Lakera provides real-time detection of prompt injection attacks, data leakage attempts, and toxic content targeting LLM applications. The platform evaluates each input for adversarial patterns and blocks threats before they reach the model. The threat detection is fast, accurate, and addresses a genuine security need. But defending against individual adversarial inputs does not govern the system's overall operational confidence. A system under sustained attack, where threat detection is blocking an increasing proportion of inputs, should reduce its execution authority rather than continuing to process the inputs that pass through the filter. Confidence governance provides this: persistent state that integrates threat detection patterns into a computation that modulates execution authority based on the threat landscape trajectory.

What Lakera provides

Lakera's platform evaluates each input to an LLM application for adversarial patterns. Prompt injection detection identifies attempts to override system instructions through crafted inputs. Data leakage prevention detects when inputs are designed to extract training data or system prompts. Toxic content filtering catches harmful inputs before they influence model behavior. The detection operates at the input layer, blocking threats before they reach the model.

Each input is evaluated independently. Threats are blocked. Clean inputs pass through to the model. The per-input evaluation does not maintain state about the pattern of threats over time or the system's operational context. A system that has blocked fifty prompt injection attempts in the last hour is evaluated identically to one that has blocked zero.

The gap between threat detection and confidence governance

A system under active adversarial attack should behave differently than one operating in a benign environment, even for the inputs that pass through threat detection. The reason is that sustained adversarial pressure increases the probability that some attacks will evade detection. No threat detection system achieves perfect recall. Under normal conditions, the small false negative rate is acceptable. Under sustained attack, the absolute number of undetected adversarial inputs increases even if the detection rate remains constant.

Confidence governance addresses this by integrating threat detection signals into a persistent state computation. When attack frequency increases, system confidence decreases. The rate-of-change detection identifies escalating attack patterns. The differential alarm triggers when the threat landscape shifts suddenly. The system's response is not to block more aggressively, which is the threat detection system's responsibility, but to reduce its execution authority for the inputs that do pass through, treating them with greater scrutiny because the operating environment has become less trustworthy.

What confidence governance enables

With confidence as a persistent state variable, the system adapts its operational posture to the threat environment. Under sustained attack, even inputs that pass threat detection are processed with reduced execution authority: tighter output constraints, more conservative generation, and lower tolerance for ambiguous inputs. The non-executing mode provides a graduated response for escalating threat conditions, from cautious execution through inquiry mode to full deferral to human oversight.

The multi-input confidence computation integrates threat detection patterns with other operational signals. A system under attack whose output quality is also declining faces compounding confidence reduction. The trajectory projection identifies whether the current threat pattern is likely to intensify or subside, informing whether to maintain reduced execution authority or prepare for further escalation.

The structural requirement

Lakera provides effective per-input threat detection for LLM applications. The structural gap is the operational governance layer: the persistent confidence computation that integrates threat patterns into a system-wide execution authority computation. Confidence governance as a computational primitive transforms per-input threat detection into threat-aware operational governance. The AI system that maintains confidence state does not merely block individual threats. It governs its own execution authority based on the trajectory of the threat environment.

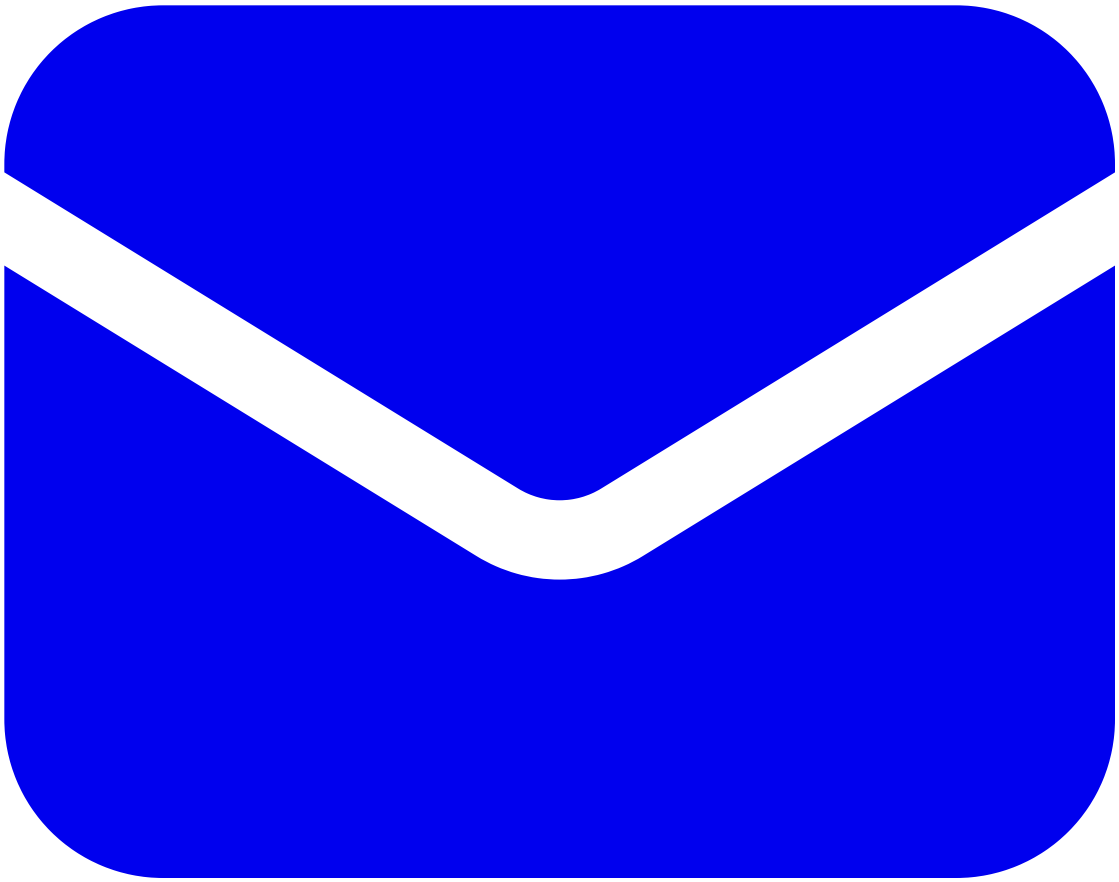
Confidence Governance All 21 steps →

Execution is a revocable permission, not a default.

Primary Technical Disclosure
○ Confidence-Governed Execution: When Agents Pause, Reassess, and Resume Safely
Secondary Technical
○ Execution as Revocable Permission○ Confidence as First-Class Computed State Variable○ Composite Admissibility Evaluator○ Confidence Trajectory Projection○ Non-Executing Cognitive Mode○ Task Class Differentiation Under Confidence Interruption○ Confidence-Integrity Feedback Loop○ Differential Rate Alarm Conditions○ Hysteretic Authorization Recovery○ Confidence Computation Function○ Confidence-Driven Inquiry Mode○ Curiosity as Confidence Modulator○ Affect-Modulated Confidence Sensitivity○ Effort Analysis and Path Optimization○ Confidence-Modulated Discovery Traversal○ Biological Signal to Confidence Coupling○ Multi-Agent Confidence Propagation○ Confidence-Governed Embodied Execution○ Deferred Execution and Temporal Reauthorization○ Execution Authorization Recovery○ Confidence Contagion in Delegation○ Confidence History Calibration○ Attention Field
Applications (General)
○ Autonomous Vehicle Execution Safety Through Confidence Gating○ Clinical AI That Pauses When It Should Not Act○ Confidence Governance for Nuclear Operations○ Confidence Governance for Aviation Autopilot Systems○ Confidence Governance for Pharmaceutical Dosing Systems○ Confidence Governance for Bridge Structural Monitoring○ Confidence Governance for Food Safety Inspection○ Confidence Governance for Chemical Plant Operations
Applications (Specific)
○ Agentforce Executes by Default○ Microsoft Copilot Has No Confidence State○ OpenAI Operator Cannot Govern Its Own Execution Authority○ Claude's Safety Has No Computed Confidence Variable○ Gemini's Multimodal Confidence Is Not Computed○ Cohere Command Generates Without Computed Confidence○ AWS Bedrock Guardrails Filter Output Without Governing Confidence○ Azure Content Safety Classifies Harm Without Governing Execution○ Google Vertex AI Safety Filters Without Confidence State○ NVIDIA NeMo Guardrails Constrains Dialogue Without Governing Confidence○ Guardrails AI Validates Output Without Governing Execution Authority● Lakera Guards Inputs Without Governing System Confidence
Confidence Governance overview →
AQ
deterministic
autonomy

Last updated: 2026-03-03

-
- Inventive Steps
- Licensing
- Patents
- Articles
- Legal
- Opportunities
- Sitemap



-
- nick@qu3ry.net
- 72 28 14 36 01

Invented by Nick Clark | Founding Investors: Devin Wilkie