Antes de Ejecutar: Una Compuerta Tipada y Calibrada para Acciones de Agentes

Ricardo Martínez

Cloud Security & IA Aplicada — NullGhost Research

Septiembre 2026

Resumen

Un agente de IA que decide qué acción tomar mediante texto libre no produce, junto con esa decisión, ninguna señal confiable de qué tan seguro está el modelo de haber elegido bien — y pedirle a otro modelo conversacional que revise esa salida es demasiado lento y costoso para ir delante de cada paso de un agente. Este texto describe un patrón de arquitectura — una compuerta de salida tipada con confianza calibrada, evaluada aquí a través del caso de TypeSafe AI y su modelo Jev [1, 2, 3] — y separa con cuidado lo verificable de lo autoreportado por el proveedor. Cierra con principios de diseño para adoptar el patrón independientemente del vendor, y un archivo de código ilustrativo que lo implementa de forma mínima.

1. La decisión sin señal de confianza

Un agente construido sobre un modelo de lenguaje conversacional decide su siguiente acción generando texto: una elección de herramienta, una clasificación, un veredicto sobre un hallazgo. Esa salida puede estar equivocada, puede venir manipulada por contenido adversarial que el agente procesó como parte de su contexto, y — este es el punto que importa — no trae consigo una medida confiable de cuánto debería confiarse en ella.

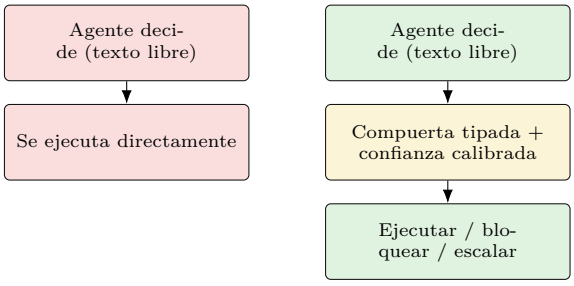

Figura 1: Sin compuerta, la decisión del agente se ejecuta directamente (izquierda). Con una compuerta tipada y calibrada de por medio, la ejecución depende de una segunda verificación estructural, independiente del modelo que decidió (derecha).

Pedirle a un segundo LLM conversacional que audite la salida del primero es una solución obvia pero costosa: 3 a 30 segundos de latencia por verificación, y un costo por token comparable al de la decisión original — inviable si un agente ejecuta docenas de pasos por tarea.

2. Un caso de estudio: modelos “System One”

TypeSafe AI, laboratorio fundado en 2024 que salió de stealth el 15 de septiembre de 2026 con $40M de ronda semilla liderada por DCVC [3], propone una categoría distinta: modelos que devuelven **valores tipados de un esquema predefinido** — clasificar, enrutar, puntuar — en vez de texto libre, con un puntaje de confianza calibrado por campo. Su primer producto, Jev, se entrena con un método propio (RLCD, *Reinforcement Learning for Calibrated Decisions*) que optimiza porque la confianza reportada coincida con la precisión real, en vez de con preferencia humana.

Verificable	**Autoreportado**
Precio publicado, latencia medible, conformidad de esquema (garantía constructiva)	“193.6× más rápido, 444.6× más barato” — evals propios, sin reproducción independiente

Figura 2: Qué tan verificable es cada tipo de afirmación sobre Jev. El análisis técnico de TrueFoundry [2] marca explícitamente la comparativa de rendimiento como sesgada por el propio vendor.

La distinción de la Figura 2 no es un detalle menor: adoptar un patrón de arquitectura por sus propiedades estructurales (salida acotada, confianza calibrada, latencia baja) es una decisión de ingeniería; adoptarlo por la cifra de marketing de un vendor es apostar a un número que nadie más ha medido.

3. El patrón: compuerta antes de ejecutar

Lo que hace viable a este patrón como capa de seguridad para agentes son tres propiedades, independientes de qué modelo específico las implemente:

- **Conformidad de esquema.** Una compuerta que solo puede devolver un valor de un conjunto predefinido no puede "alucinar" una acción fuera de ese conjunto — es una restricción estructural, no una heurística.
- **Confianza calibrada.** Un umbral de confianza permite una regla operativa simple: por debajo de cierto nivel, la acción se bloquea o se escala a revisión humana en vez de ejecutarse a ciegas.
- **Latencia baja.** Una compuerta de 70–500ms puede ir delante de cada acción de un agente sin destruir su presupuesto de latencia; una de 3–30s no puede.

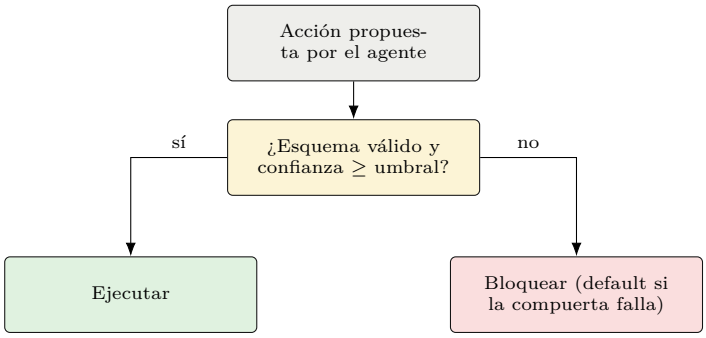

Figura 3: El patrón implementado en `patterns/01_typed_calibrated_gate.py`: si la compuerta misma falla (excepción, esquema desconocido), el resultado por defecto es bloquear, nunca ejecutar en silencio.

El archivo `patterns/01_typed_calibrated_gate.py` de este repositorio implementa exactamente la Figura 3 de forma mínima y ejecutable, sin dependencias externas — escrito desde cero para este texto, no extraído de ningún sistema en producción.

4. Dónde no ayuda

- **No sustituye un control de alcance determinista.** Validar que una acción está dentro del esquema permitido es distinto de validar que el objetivo de esa acción está autorizado; ambos controles son necesarios, ninguno reemplaza al otro.
- **Cardinalidad limitada.** Un esquema de decisión con cientos de categorías posibles puede requerir un enfoque de dos etapas más lento que el caso simple.
- **Sin benchmark independiente de detección de intenciones adversariales.** Que un vendor liste esto como caso de uso no equivale a una tasa de acierto medida por un tercero.
- **Riesgo de vendor temprano.** Un proveedor de dos años en acceso anticipado, sin opción self-hosted todavía, es una dependencia a evaluar con el mismo criterio que cualquier proveedor externo nuevo.

5. Principios de diseño

1. Una compuerta que falla abierta no es una compuerta — el default ante cualquier error debe ser bloquear, nunca ejecutar.
2. Confianza sin calibración es teatro: un número que no coincide con la precisión real no informa ninguna decisión.
3. Salida tipada es una garantía estructural, no una promesa de acierto — elimina el formato incorrecto, no la categoría incorrecta.
4. No adoptes el benchmark del proveedor como si fuera el tuyo; mide la calibración con tus propios casos conocidos antes de confiar en ella.
5. Esto complementa el control de alcance determinista existente; no lo sustituye.

Nota metodológica: este texto evalúa un producto comercial de terceros (TypeSafe AI / Jev) con base en fuentes públicas citadas abajo. No describe ni depende de ningún sistema de producción de NullGhost o KIRUX; el archivo en `patterns/` es código ilustrativo original, no extraído de un sistema real.

Referencias

[1] TypeSafe AI. *Introducing System One Models & Jev*. typesafe.ai/blog, septiembre 2026.

[2] TrueFoundry. *TypeSafe AI's Jev: What "System One Models" Actually Are*. truefoundry.com/blog, 2026.

[3] SiliconANGLE. *TypeSafe AI exits stealth with $40M to build AI for use by software*. siliconangle.com, 16 de septiembre de 2026.