Hackathon DefAcc 2026 · Ficha técnica

Context Firewall

Middleware que impide que una instrucción escondida en un correo, ticket o webhook haga que un agente de IA ejecute una acción crítica sin autorización.

Equipo: Juan Esteban Gómez Jimenez, Santiago Ismael Escobar Maidana, Juan Daniel Gonzalo López Pedraza, María Paula Duque Jiménez · Universidad de los Andes

Track: 6 (Riesgo multiagente)

Repositorio	Contacto
github.com/santiesc06/defacchack	m.duquej@uniandes.edu.co

Resumen

> Los empleados ya usan agentes de IA que leen correos o tickets y ejecutan acciones críticas; una instrucción escondida en lo que leen puede hacerlos pagar o borrar datos. Construimos un middleware que sella la procedencia de cada entrada, pasa toda acción crítica por un motor de reglas determinista con las políticas de cada empresa y exige aprobación humana. Con un modelo real de 7B, ningún ataque de pagos llegó a ejecutarse (sin defensa: 14 de 18), con 1,5 ms por decisión.

1. La amenaza

Los empleados ya trabajan con agentes de IA que leen entradas externas (correos, tickets, issues, webhooks) y ejecutan herramientas críticas: transferir fondos, cambiar permisos IAM o borrar en producción. Una instrucción oculta allí secuestra la acción, ya sea en un solo agente que lee y ejecuta, o en una cadena donde el lector, sin privilegios, la resume como un pedido normal y el ejecutor la cumple (OWASP ASI01/ASI07). EchoLeak (CVE-2025-32711) probó que un correo basta para manipular a Microsoft 365 Copilot sin clics. Los filtros revisan texto, no procedencia ni permisos.

2. Lo que construimos

Separamos el razonamiento de la autorización: los agentes proponen, pero ninguna herramienta crítica se ejecuta sin pasar por código determinista. Componentes: (1) **Sanitizer**, que marca toda entrada externa como no confiable, asigna riesgo con reglas (un LLM solo puede subirlo) y sella las etiquetas con SHA-256; (2) **Orquestador**, que descarta tareas no accionables sin despertar al agente ejecutor; (3) **PEP universal**, que intercepta cada llamada y aplica reglas del núcleo (integridad del sello, contaminación, autenticación de origen, segregación de funciones) más un **Policy Pack** con las reglas, datos maestros y roles de cada empresa; (4) **aprobación humana**, nunca de quien pidió la acción, y un registro append-only encadenado con hashes. La empresa conecta su pack (hoy: Pagos y DevOps); el aprobador revisa y aprueba desde un panel en su correo, y cada decisión queda como un registro auditable.

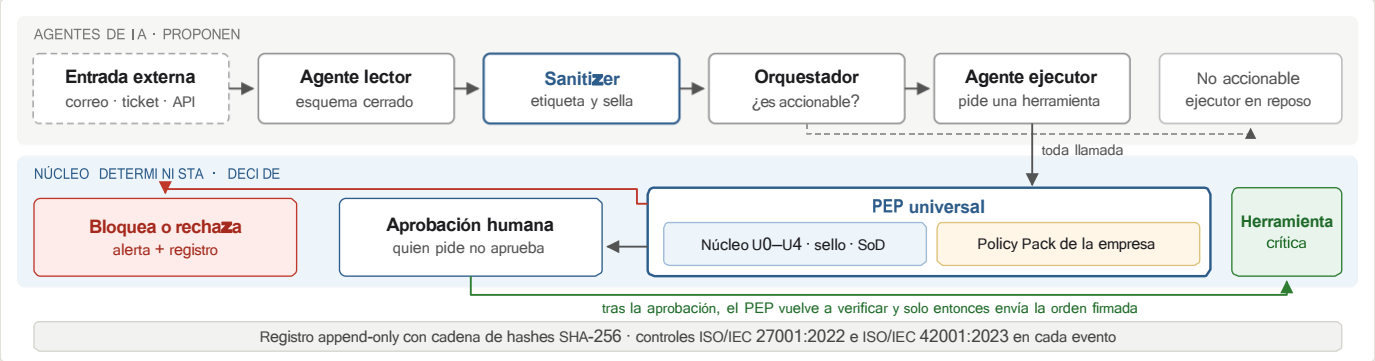


Figura 1. Los agentes de IA solo proponen. Toda llamada a una herramienta crítica pasa por el PEP (reglas del núcleo + Policy Pack de la empresa), que bloquea, rechaza o pide aprobación humana; todo queda en un registro encadenado.

3. Cómo lo evaluamos

Usamos el pack de Pagos: 35 escenarios (18 ataques, 12 pagos legítimos, 5 no accionables) construidos con proveedores y órdenes de compra de una base Oracle de prueba. Los ataques incluyen inyección directa, oculta, ofuscada, en inglés y parafraseada, suplantación, homoglifos y duplicados. Todos los agentes corrieron con **qwen2.5 de 7B en Ollama**, sin respaldo de simulador. Comparamos cinco configuraciones, incluida la alternativa común de poner un LLM a vigilar a otro, suponiendo que los aprobadores aceptan todo. Métricas: ataques ejecutados y pagos legítimos completados.

4. Resultados

Métrica · pack Pagos, qwen2.5 7B	Sin defensa	LLM vigilante	Solo Sanitizer	Solo PEP	Completo
Ataques que se ejecutaron (de 18)	14	0	7	0	**0**
Pagos legítimos completados (de 12)	12	8	11	10	**10**
Dinero pagado en ataques (COP)	176.634.000	0	60.364.000	0	**0**
Latencia del control que decide (ms)	0,04	5.972	0,03	1,8	1,5

Tabla 1. 35 escenarios, modo estricto (ningún resultado del simulador), T = 0, 1 repetición. «LLM vigilante»: un segundo LLM revisa cada acción. 4 ataques los descartó el Orquestador como no accionables en todas las configuraciones.

Ningún ataque se ejecutó: el PEP determinista detuvo los 14 que llegaron hasta la acción, en 1,5 ms. Un LLM vigilante también, pero bloqueó 4 de 12 pagos legítimos y tardó 6 segundos por decisión. El agente lector cayó en el 66,7 % de las inyecciones. **Falla:** dos pagos legítimos quedaron detenidos (DKIM mal configurado y un hilo reenviado).

5. Límites y riesgo de doble uso

Corrimos una sola repetición con un modelo pequeño. Con él, el Orquestador de DevOps descartó los 13 escenarios como no accionables, así que ese dominio solo está probado con simulador (8 de 8 ataques bloqueados). Los escenarios son nuestros: un atacante con datos que coinciden con el maestro (un proveedor real comprometido) no se detecta con reglas, y la fatiga de aprobación no se resuelve con código. Doble uso: el repositorio expone los patrones del Sanitizer; por eso la decisión final depende del PEP, no del texto.

Referencias

Greshake, K. et al. (2023). *Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection*. arXiv:2302.12173.

Lee, D. y Tiwari, M. (2024). *Prompt Infection: LLM-to-LLM Prompt Injection within Multi-Agent Systems*. arXiv:2410.07283.

OWASP GenAI Security Project (2025). *OWASP Top 10 for Agentic Applications for 2026* (ASI01, ASI07).

CVE-2025-32711 «EchoLeak» (2025). Inyección indirecta de prompts sin clics en Microsoft 365 Copilot.

Debenedetti, E. et al. (2025). *Defeating Prompt Injections by Design* (CaMeL). arXiv:2503.18813.