Agente Auditor de Confianza Cero: defensa contra inyección indirecta en contratación pública

Frente 01: Agent Security · Equipo No Prompt, No Party · Sergio Castaño, Angie Gutiérrez, Camilo Murcia, Santiago Gómez

Un asistente de auditoría de contratación pública que sigue siendo seguro aunque su LLM obedezca al atacante. Todos los números salen de `RESULTS.md`, que cita el archivo de cada uno. † = el número está en `ML_ARCHITECTURE.md` §5d, pero sus datos crudos no se subieron al repo.

1. Vector de amenaza

Inyección indirecta de prompts: la orden del atacante llega dentro de un dato que el agente lee (HTML oculto, imágenes, documentos de oficina, facturas de contratistas). Se suma el *confused deputy*: un agente con acceso a expedientes reservados y a la web puede ser usado para sacar datos de ciudadanos o alterar archivos internos. Un agente único sin defensas cae en 0.96 de 338 ataques (atacante oráculo).

2. Arquitectura defensiva

- **3 enclaves** con privilegios disjuntos: Orquestador (planea, sin herramientas), Trabajador de Bóveda (lee expedientes, sin red) y Aduana (lee internet y documentos externos, sin bóveda).
- **CapabilityBroker** determinista, único camino a las herramientas: tokens HMAC por paso, validación de rutas, taint L0/L1/L2 (lo reservado no sale a la red), monitor de egreso con canaries y canaries de system prompt.
- **Aduana**: el plan se congela antes de leer datos externos; el texto libre va a cuarentena como handle opaco; cada valor extraído se verifica contra la fuente; escáner de doble chequeo (regex + Galvanize-3); datamarking.
- Idea central: la seguridad no depende de detectar la inyección. El LLM que lee el documento no tiene herramientas para hacer daño, y el que tiene herramientas nunca lee el documento.

3. Metodología de evaluación

- **Extremo a extremo** (`eval/benchmark/e2e.py`): 614 casos (338 ataques) por configuración, contra el sistema completo. Atacante **oráculo**: un LLM simulado que obedece toda inyección que llega a ver. Es el peor caso.
- **Línea base**: A, un agente monolítico con todas las herramientas. Ablación: D-arch (sin filtro), D-regex, D.
- **Jueces automáticos sobre efectos observables**: canary en el log del atacante, `app.py` modificado, lectura no autorizada de la bóveda, monto envenenado en el acta, system prompt filtrado.
- **Escáner aislado** (`eval/benchmark`): 611 casos (338 inyecciones, 273 legítimos).
- **LLM real**: llama3 en Ollama, muestra de 16 casos. Todo corre en local con datos sintéticos.

4. Resultados

Extremo a extremo, 614 casos, atacante oráculo	ASR	Utilidad	Falso bloqueo
A: agente sin defensas	0.96	1.00	0.00
D-arch: solo arquitectura, sin filtro	0.00	1.00	0.00
D: arquitectura + filtro	0.00	0.87	0.13

- En D-arch el LLM obedeció la inyección en los 338 ataques y ninguno causó daño. La arquitectura contiene sola.
- El filtro bloquea 0.84 de los ataques antes de que un LLM los vea, y cuesta utilidad: de 1.00 a 0.87. Con envoltura neutral (sin el título del proceso), 0.94 de utilidad y 0.06 de falso bloqueo.
- Escáner aislado: detección 0.81 †, falsas alarmas 0.08 † (antes de calibrar: 0.85 / 0.16).
- **LLM real, muestra pequeña**: A 2/8 ataques con daño †, D 0/8 †. Latencia de D: p50 1.7 s †, máx 2.2 s †.

Reproducir: `python -m pytest -q` (383 tests) y `python main.py --mock --scenario atk_multihop_app`.

5. Limitaciones

- Todo corre en un mismo proceso Python; no hay sidecar que aísle el broker.
- La prueba con LLM real es pequeña (16 casos, un solo modelo). El resto usa un atacante simulado.
- El umbral del escáner se calibró sobre el mismo benchmark que se reporta.
- El filtro bloquea 13 % de los documentos legítimos, y es sensible al contexto de la página.
- Los modelos de Hugging Face se cargan sin revisión fija, y WARD-2b con `trust_remote_code`.
- Canales laterales de tamaño y tiempo sin mitigar. Un valor falso pero bien formado pasa esquema y procedencia.

- La variante E (`ml_d`) no completa las tareas benignas de la matriz oficial por un desajuste con su planner.

6. Uso dual

El banco de ataques sirve para probar defensas, y también describe cómo atacar agentes. Mitigaciones: los payloads son genéricos y de demostración; solo apuntan a `127.0.0.1` (una allowlist lo impone en código); no hay técnicas de evasión contra productos reales; los datos y canaries son sintéticos. El entregable es la defensa.

Demo de 60 s

`python main.py --mock --scenario atk_multihop_app` — una "nota de migración" en un changelog pide modificar `app.py`. A: `app.py modificado: SI`. D y E: bloqueado, con la regla que negó. Evidencia: el `audit.jsonl` que imprime la demo. Usa el LLM mock; con Ollama y `.env` configurado, el mismo comando con `--live`.