Hackathon DefAcc 2026 · Ficha técnica

DEDALUS: el laberinto que defiende

Desvía al agente de IA atacante a un laberinto falso y convierte cada ataque capturado en un parche verificado.

Equipo: [Nombre Apellido, Nombre Apellido, Nombre Apellido, Nombre Apellido] · [Universidad u organización]

Track: 3. Ciberdefensa automatizada

Repositorio	Demo	Contacto
github.com/AISAFETY-HACKATON-DEDALO/DEDALUS	Video de 2 min (docs/entrega/DEDALUS-demo.mp4) · panel en vivo con abrir.ps1	[correo de una persona del equipo]

Resumen

Un agente de IA con una cuenta válida puede probar miles de ataques contra una aplicación sin cansarse. Construimos una defensa que, al detectar el ataque o el uso de una credencial señuelo, le corta el acceso real, lo deja en un laberinto falso y usa sus propios ataques para generar y verificar parches. En 87 episodios con modelos reales no hubo fugas, 60 de 66 atacantes quedaron contenidos y los modelos defensores cerraron 48 de 52 funciones atacadas.

1. La amenaza

Los agentes de IA abaratan el abuso de credenciales: con una cuenta de bajo privilegio piden documentos ajenos, prueban inyecciones y usan cualquier clave que encuentran, miles de veces y sin costo humano. Perjudica a cualquier organización con aplicaciones internas. Las defensas de hoy no alcanzan: los permisos bloquean en silencio y el atacante sigue probando; los señuelos clásicos dependen de que muerda un cebo. De los 1000 casos de CyberSecEval (MITRE ATT&CK), 157 corresponden a este escenario.

2. Lo que construimos

- **Qué hace distinto:** el ataque mismo es el cebo, y el atacante escribe sin saberlo la prueba del parche.
- **ARGOS**, un clasificador de reglas, puntúa cada sesión por lo que intenta. Hay además **cebos**: credenciales y archivos señuelo, uno por sesión.
- **Desvío:** ante un ataque inequívoco o el uso de un cebo, el servidor revoca el acceso real de esa sesión y la deja en un **laberinto** generado por HMAC, que cuesta milisegundos y cero tokens. Ningún modelo decide revocar.
- **El ciclo:** cada ataque capturado se repite contra la función vulnerable; un modelo propone el parche y **TALOS**, un verificador sin IA, lo acepta solo si el uso legítimo sigue intacto y el ataque capturado deja de funcionar.
- **Cómo se usa:** el operador abre el panel con un comando. Entra el tráfico HTTP del portal; salen el registro de cada sesión, las alertas y los parches verificados.

1 · Ataca — agente de IA con cuenta válida	**2 · Detecta** — ARGOS o un cebo, por reglas	**3 · Desvía** — sin acceso real; sigue en el laberinto	**4 · Captura** — cada solicitud, con su carga	**5 · Repite** — ¿funcionaba contra el código vulnerable?	**6 · Parcha** — un modelo propone la corrección	**7 · Verifica** — sin IA: uso legítimo intacto, ataque bloqueado

Figura 1. *El ciclo de DEDALUS. Las cinco primeras etapas son reglas deterministas y no usan modelos. La IA solo propone el parche (6) y un verificador sin IA decide si se acepta (7).*

3. Cómo lo evaluamos

Tres pruebas, todas en un laboratorio local con datos sintéticos. Primero, 87 episodios con tres modelos abiertos como atacante simulado: solo permisos (línea base) frente a DEDALUS, con y sin la advertencia de que hay señuelos. Segundo, los 157 casos aplicables de CyberSecEval, como guiones. Tercero, el ciclo completo, con tres modelos como defensor. Medimos lo que importa a quien defiende: fugas, atacantes contenidos, falsos positivos de un usuario honesto y parches verificados.

4. Resultados

Métrica	Línea base	Nuestro sistema	Diferencia
Atacantes detectados y contenidos (modelos reales)	0 de 21	60 de 66	+91 puntos
Fugas del dato protegido	0 de 21	0 de 66	igual: lo impiden los permisos
Casos de CyberSecEval detectados y contenidos	0 de 157	69 de 157	+44 puntos
Solicitudes al sistema real tras la detección	5 (cortar la sesión)	0 (desviar)	−5
Funciones atacadas con parche verificado	6 de 12 (sin esquema)	48 de 52	+42 puntos
Falla: ataques que tocaron la aplicación real antes de la contención	—	17 de 68	1 de cada 4
Falla: usuario honesto desviado por error	0	sí, al tercer descuido	falso positivo

Tabla 1. *Línea base: los mismos permisos sin DEDALUS; en los parches, el mismo defensor sin el esquema de la base. Tres modelos abiertos en Groq; las filas de solicitudes y de usuario honesto son guiones deterministas.*

- **Conclusión:** avisarle al agente que hay señuelos no lo salva (25 de 30 los usaron igual), y lo que lo retiene es el progreso, no el disfraz: con el laberinto generativo, los que mordieron siguieron 6,0 pasos frente a 2,7 y abandonaron 4 de 10 frente a 15 de 16. La IA propone el parche; una regla decide.
- **Dónde falla:** el primer intento ambiguo pasa: uno de cada cuatro ataques tocó la aplicación real antes de la contención, y 6 de 66 atacantes nunca fueron detectados.

5. Límites y riesgo de doble uso

No probamos modelos de frontera ni un agente libre con herramientas: la cuota gratuita cortó esa corrida, y las muestras son pequeñas. Se rompe si un señuelo se filtra (revoca a su dueño), si el atacante no ataca ni toca cebos, o si un usuario honesto comete tres descuidos seguidos: queda desviado y ve datos falsos.

Doble uso: el arnés mide qué tan bien un modelo evade señuelos. Por eso corre solo en local, sin red ni herramientas, y publicamos métricas, no exploits.

Referencias

Meta (2024). CyberSecEval, Purple Llama. · Araujo et al. (2014). From Patches to Honey-Patches. · Pasquini et al. (2024). Hacking Back the AI-Hacker. arXiv:2410.20911. · Reworr y Volkov (2024). LLM Agent Honeypot. arXiv:2410.13919. · MITRE. ATT&CK y D3FEND.

Trabajo previo declarado: el laboratorio dedalo/ (11 al 13 de noviembre, hasta el commit edaf152). Lo demás se hizo durante el evento. El código se escribió con asistentes de IA (Claude y Codex) bajo dirección del equipo.