So you gave an LLM a shell.

Guion de ensayo: **49:55**, 59 diapositivas principales y 38 de apoyo. Las preguntas quedan fuera del tiempo de exposición. Diapositivas en inglés; notas y transiciones en español.

Los tiempos son objetivos de ensayo, no una grabación. Los replays se narran saltando a llamadas concretas; no se reproducen sus duraciones completas. Para acortar, recortar la explicación de la matriz PMOX y el segundo ejemplo de controladores; conservar el método, el caso real y las conclusiones.

01 · 00:00 · So You Gave an LLM a Shell

02 · 00:20 · “LLMs are text calculators.”

03 · 01:05 · From autocomplete to vulnerability discovery

04 · 01:50 · Cyber capability is now gated

05 · 02:30 · What does “good at cyber” actually measure?

06 · 03:15 · What we wanted to know

07 · 03:55 · Six settings. Different claims.

08 · 04:25 · Q&A: isolate the model.

09 · 05:15 · Docker: a complete, bounded loop.

10 · 06:05 · The model is not the agent.

11 · 07:00 · A command is only the beginning.

12 · 07:40 · Vulnerable machines: the target remembers.

13 · 08:30 · Active Directory: five hosts, shared identity.

14 · 09:20 · PMOX: a chain with four objectives.

15 · 10:20 · Four contrasts an operator actually faces.

16 · 10:55 · Twelve models. Several families.

17 · 11:25 · Same model. Different ways to work.

18 · 12:05 · ReAct: one step, then listen.

19 · 13:00 · OODA: update the map before acting.

20 · 13:55 · What did we actually tell the agent?

21 · 15:00 · Hermes: the agent around the model.

22 · 15:50 · Instructions can shape the failure mode.

23 · 16:40 · Eight categories. One challenge in each.

24 · 17:30 · Grok and Kimi led the knowledge test.

25 · 18:10 · Codex solved all 24 attempts.

26 · 19:10 · The exploit chain separated the models.

27 · 20:05 · Choose for the job in front of you.

28 · 20:50 · The prompt can help. Or get in the way.

29 · 21:40 · More instructions. Less answer.

30 · 22:20 · 2.1× tokens per solve. Little extra coverage.

31 · 23:25 · Extra structure helped one run and hurt another

32 · 24:25 · Give the harness a specific job.

33 · 25:15 · Same total. Different winners.

34 · 26:20 · Chinese models were serious contenders.

35 · 27:15 · A refusal and a failed request are different.

36 · 28:10 · 8B can solve CTFs. Size still matters.

37 · 29:10 · The economics of a verified effect

38 · 30:10 · Failures consumed most of the tokens.

39 · 30:45 · Spend capability where it is needed.

40 · 31:30 · Choose the model. Then tune the workflow.

41 · 32:30 · Failure often did more work.

42 · 33:35 · Track progress. Make it visible.

43 · 34:30 · Getting in was easier than finishing.

44 · 35:25 · The wall was keeping the operation moving.

45 · 36:15 · Same target. Different decisions.

46 · 38:35 · Root at call 58. Still running at call 100.

47 · 39:50 · A foothold is an identity on a host.

48 · 41:05 · A callback is not the finish line.

49 · 42:55 · A real pentest. A human in the loop.

50 · 43:35 · What filled the 31 hours?

51 · 44:15 · The dangerous part was joining the steps.

52 · 45:10 · The operator kept the chain connected.

53 · 46:20 · Identity control and data exposure.

54 · 46:55 · The operator steered. The agent did the legwork.

55 · 47:40 · The human still has a job.

56 · 47:55 · Components first. Operational systems next.

57 · 48:25 · Next: complete operation frameworks.

58 · 49:00 · Useful attackers. Uneven operators.

59 · 49:55 · Questions

---

01 / 00:00–00:20 / 20 S

So You Gave an LLM a Shell

Soy Pau Muñoz. Investigo IA y seguridad, y fundé Lobera. Hoy quiero responder a una pregunta bastante concreta: cuando damos herramientas a un modelo, ¿qué parte del resultado viene del modelo y qué parte viene del sistema que hemos construido alrededor? Vamos a pasar de preguntas de ciberseguridad a un pentest real asistido por una persona.

Empecemos por lo que estamos conectando a esa shell.

Abrir diapositiva ↗

02 / 00:20–01:05 / 45 S

“LLMs are text calculators.”

La frase de que las LLM son calculadoras de texto describe una parte del mecanismo. El modelo propone una secuencia de tokens. El runtime interpreta parte de esa secuencia como una acción, ejecuta una herramienta y devuelve su resultado. Ese resultado entra en la siguiente decisión.

El cambio operativo está en cerrar este circuito. Los permisos, la ejecución, el estado del objetivo y la decisión de detenerse pertenecen al sistema que rodea al modelo. Por eso dos agentes con el mismo modelo pueden comportarse de manera muy distinta. Todavía no necesitamos una arquitectura enorme: modelo, herramienta, entorno y feedback.

¿Cómo llegamos de completar texto a ejecutar una tarea de sistemas?

Abrir diapositiva ↗

---

03 / 01:05–01:50 / 45 S

From autocomplete to vulnerability discovery

Pasamos de completar una línea a generar funciones, navegar repositorios, ejecutar pruebas y revisar resultados. En seguridad, esa misma secuencia permite encadenar hipótesis, reproducción y corrección. El efecto relevante es cuánto se comprime el ciclo de investigación completo.

El caso de PHP ilustra esa transición: descubrimiento asistido, reproducción y coordinación de una corrección. Está aquí como contexto externo y con fuentes identificadas, no como parte de nuestros datos. No voy a construir una colección de titulares: los siguientes resultados proceden de nuestros experimentos.

La discusión dejó de ser solamente qué puede generar el modelo.

Abrir diapositiva ↗

04 / 01:50–02:30 / 40 S

Cyber capability is now gated

Buena parte de la conversación sobre Anthropic, modelos chinos y modelos abiertos mezcla tres preguntas. ¿Puede resolver la tarea? ¿La petición llega al despliegue que queremos evaluar? ¿Ese despliegue acepta actuar en este contexto? Son variables diferentes.

Glasswing y Daybreak muestran que el acceso a capacidades de ciberseguridad se está convirtiendo en una condición de producto. Las controversias sobre destilación tampoco convierten la nacionalidad en una medida de capacidad. En esta charla separaré origen, disponibilidad de pesos, política, ruta de proveedor y resultado observado. Un filtro del proveedor no demuestra que el modelo carezca de conocimientos.

Veamos primero qué miden los benchmarks que ya existen.

Abrir diapositiva ↗

---

05 / 02:30–03:15 / 45 S

What does “good at cyber” actually measure?

Antes de contar nuestro experimento, situemos el mapa. CyberMetric pregunta por conocimiento de seguridad. Cybench da retos CTF con herramientas. CVE-Bench enfrenta al agente a aplicaciones con vulnerabilidades reales. CyberGym entrega un código vulnerable y una descripción, y pide un PoC que reproduzca el fallo.

No voy a comparar sus porcentajes: cada uno le da al agente un trabajo distinto. Nosotros queremos responder una pregunta de operador: si cambio el modelo, el flujo de trabajo o el entorno, ¿qué cambia en el resultado y en la factura? Por eso necesitamos varias capas y mirar también las trayectorias. En los materiales hay ocho referencias que conectan con lo que vamos a ver.

Con ese mapa, estas son nuestras preguntas de operador.

Abrir diapositiva ↗

06 / 03:15–03:55 / 40 S

What we wanted to know

Queríamos saber quién funciona mejor, en qué tareas, y con qué barrera de dificultad. Después, cuánto cambia el resultado al cambiar la estrategia o las herramientas. También si los despliegues chinos o con pesos disponibles se acercan a los servicios de frontera. Y, finalmente, cuánto cuesta producir un efecto verificable y qué podemos decir del tamaño del modelo.

No todas estas preguntas tienen la misma calidad de respuesta. Iré distinguiendo comparaciones controladas, trazas individuales y el caso de campo asistido.

Necesitamos distintos laboratorios para responderlas.

Abrir diapositiva ↗

---

07 / 03:55–04:25 / 30 S

Six settings. Different claims.

Vamos a subir cinco escalones: preguntas de seguridad, retos en Docker, máquinas vulnerables, Active Directory y una cadena completa en Proxmox. Cada escalón exige algo nuevo: saber, actuar, conservar acceso, entender identidades y mantener una misión con varios objetivos.

Después cambiaremos el agente: probaremos Hermes con una metodología cyber. Y terminaremos fuera del laboratorio, con un pentest real dirigido por una persona. Así veremos qué cambia cuando el sistema tiene más herramientas y alguien conserva el rumbo.

El primer escalón mide conocimiento sin ejecutar nada.

Abrir diapositiva ↗

08 / 04:25–05:15 / 50 S

Q&A: isolate the model.

Aquí no hay una shell. Entregamos un escenario de seguridad, sus evidencias y una pregunta. El modelo responde y un evaluador compara esa respuesta con una rúbrica y una referencia que el modelo examinado no ha visto.

Es el equivalente a ponerle un caso técnico encima de la mesa: qué ha ocurrido, qué comprobarías después y qué no puedes afirmar todavía. También cambiamos cómo le pedimos contestar: directamente, con una estructura o escribiendo un borrador que después critica y revisa. Queremos saber si pedirle más trabajo produce una respuesta mejor.

Ahora sí: vamos a darle un entorno donde sus acciones tengan consecuencias.

Abrir diapositiva ↗

---

09 / 05:15–06:05 / 50 S

Docker: a complete, bounded loop.

Aquí ya existe un agente que puede actuar. El controlador habla con el modelo, el agente ejecuta herramientas desde su contenedor y el objetivo vive en una red privada del reto. La salida vuelve al controlador y la comprobación final valida el objetivo.

Separar estos componentes nos permite mantener fijos el target, los límites y el criterio de éxito mientras cambiamos el despliegue o el controlador. El gateway y el ejecutor también forman parte del experimento: un comando mal interpretado o una herramienta ausente cambia lo que recibe el modelo. La arquitectura sirve para definir una comparación, no solamente para dibujar cajas.

Entre el modelo y ese entorno hay una pieza que tenemos que definir.

Abrir diapositiva ↗

10 / 06:05–07:00 / 55 S

The model is not the agent.

El modelo elige una acción. Para que esa acción ocurra hace falta software: llamar al modelo, interpretar la respuesta, ejecutar una herramienta, recoger la salida y preparar el siguiente turno. A ese conjunto lo llamamos harness.

También mantiene lo que ya sabemos, las sesiones y los archivos del trabajo, y decide cuándo terminar. Una herramienta hace una tarea concreta: una petición HTTP, un comando o una consulta. El harness conecta esas tareas en una operación. Modelo más ese sistema es el agente.

Veamos una sola acción para entender quién hace qué.

Abrir diapositiva ↗

---

11 / 07:00–07:40 / 40 S

A command is only the beginning.

El modelo propone una petición HTTP con curl. Todavía no ha hablado con el objetivo: ha producido texto. El harness comprueba el alcance, ejecuta ese comando, recoge la respuesta y conserva el resultado.

El servidor devuelve una redirección a admin. Esa es evidencia del entorno. Si la siguiente llamada al modelo recibe esa evidencia, puede decidir inspeccionar el login. Si el runtime pierde la salida o la sesión, el mismo modelo está trabajando con otro problema. Esa es la diferencia entre tener una herramienta y tener un agente que sabe continuar.

La siguiente dificultad es que el objetivo conserve estado.

Abrir diapositiva ↗

12 / 07:40–08:30 / 50 S

Vulnerable machines: the target remembers.

En las máquinas vulnerables, el target es una VM completa en VirtualBox. El ejecutor es un subprocess filtrado del host, no el contenedor del benchmark anterior. La máquina conserva archivos, procesos, sesiones y cambios producidos por las acciones anteriores.

Tenemos que comprobar el estado inicial, restablecer snapshots y conservar las observaciones por ejecución. El objetivo ya puede exigir descubrimiento, acceso inicial, identificación de credenciales, escalada y una prueba final. La lista de herramientas y el tratamiento de la salida son parte del contrato: si el prompt promete herramientas que no existen, el agente desperdicia acciones por un fallo del laboratorio.

Con varios hosts aparece además el problema de la identidad.

Abrir diapositiva ↗

---

13 / 08:30–09:20 / 50 S

Active Directory: five hosts, shared identity.

Aquí hay cinco máquinas: tres controladores de dominio y dos servidores miembro. El modelo parte del reconocimiento y trabaja con herramientas desde el ejecutor del laboratorio.

El cambio es que el mapa ya no se entiende solo con IPs y puertos. Hay usuarios, grupos, credenciales y relaciones de confianza. Conseguir ejecución en un servidor te da una posición; entender qué puede hacer su identidad abre el siguiente movimiento.

Y en PMOX conectamos varios objetivos en una sola misión.

Abrir diapositiva ↗

14 / 09:20–10:20 / 60 S

PMOX: a chain with four objectives.

En Proxmox montamos cuatro objetivos y una VM atacante. El agente ejecuta por SSH desde esa VM: entra por una superficie DMZ, descubre una aplicación interna, se enfrenta a Windows y busca la bóveda final.

Las máquinas comparten la VLAN del laboratorio. Lo que dibujamos es la cadena lógica de la misión. Cada flag confirma una etapa, pero el agente tiene que descubrir cómo llegar: las respuestas del evaluador no forman parte de sus instrucciones.

Ya sabemos dónde los probamos. Ahora, por qué elegimos estos modelos.

Abrir diapositiva ↗

---

15 / 10:20–10:55 / 35 S

Four contrasts an operator actually faces.

¿Por qué elegimos estos modelos? El conjunto representa cuatro decisiones que tiene cualquier equipo: especialistas de código frente a asistentes generales; pesos accesibles frente a servicios cerrados; rutas económicas frente a otras más caras; y modelos pequeños frente a familias de mayor tamaño.

No es una selección de todas las novedades del mercado. Queríamos comparar contrastes útiles bajo las mismas tareas y herramientas. Estos doce son el core común de Q&A y Docker; el estudio específico de tamaño y otros laboratorios usan sus propios conjuntos.

Estos son los doce despliegues del conjunto compartido.

Abrir diapositiva ↗

16 / 10:55–11:25 / 30 S

Twelve models. Several families.

Estos doce modelos forman la comparación común de preguntas de seguridad y retos Docker. Hay familias de OpenAI, xAI, Anthropic, Google e IBM, y de Moonshot, Alibaba, Z.ai, MiniMax y DeepSeek. Algunos permiten descargar pesos; otros son servicios cerrados. El conjunto incluye especialistas de código, asistentes generales y modelos más pequeños.

Los nombres identifican la versión y la ruta que realmente probamos. El resto de estudios amplía el conjunto o cambia el entorno; no vamos a sumar esas pruebas como si fueran una única competición. Lo que mantenemos comparable en cada contraste son la tarea, las herramientas y las instrucciones.

También cambiamos cómo les pedimos organizar el trabajo.

Abrir diapositiva ↗

---

17 / 11:25–12:05 / 40 S

Same model. Different ways to work.

Una estrategia de razonamiento es una receta para organizar decisiones. Direct pide el próximo paso. ReAct alterna una justificación breve, una acción y la observación del resultado. Plan–ReAct añade una lista de pasos que puede cambiar. Reflection revisa qué falló antes del siguiente intento. Structured obliga a mantener hipótesis y evidencias ordenadas.

La estrategia no instala conocimientos nuevos en el modelo. Cambia qué tiene que producir, qué información debe conservar y cuándo replantear el trabajo. El harness es el código que hace cumplir esa receta. Veamos las dos ideas que más se van a repetir.

Primero, el ciclo más sencillo de seguir con una herramienta.

Abrir diapositiva ↗

18 / 12:05–13:00 / 55 S

ReAct: one step, then listen.

Este es un ejemplo didáctico. Sabemos que HTTP responde. El modelo explica qué quiere aprender: identificar la aplicación. Pide una acción, la herramienta la ejecuta y el servidor devuelve una redirección a admin.

La siguiente decisión ya tiene una observación nueva. Puede seguir el login. No debería tratar una redirección como prueba de acceso. Ese pequeño ciclo es ReAct: plantear el siguiente movimiento, actuar, leer qué pasó y actualizar. Lo valioso no es escribir una explicación larga; es que el resultado cambie la próxima acción.

Otra receta explicita cómo interpretar lo observado antes de actuar.

Abrir diapositiva ↗

---

19 / 13:00–13:55 / 55 S

OODA: update the map before acting.

OODA significa Observe, Orient, Decide, Act: observar, orientarse, decidir y actuar. Viene del ciclo de decisión de John Boyd. Lo adaptamos como una forma de organizar el trabajo del agente.

Observar: veo HTTP y SMB. Orientarse: tengo dos rutas y ninguna credencial; quizás la aplicación me dé contexto útil. Decidir: priorizo web y guardo SMB para después. Actuar: inspecciono la aplicación y devuelvo el resultado al mapa.

La fase Orient conecta el dato nuevo con el objetivo y lo que ya sabemos. Docker usa otro controlador, Structured, con una disciplina ODAA: observar, decidir, actuar y evaluar qué produjo la acción. La implementación secundaria ODAA también es distinta. No equiparamos esos nombres como si fueran el mismo código.

Estas recetas se convierten en instrucciones concretas.

Abrir diapositiva ↗

20 / 13:55–15:00 / 65 S

What did we actually tell the agent?

Esta es una pregunta que haríamos todos: vale, pero ¿qué le escribisteis? Aquí hay fragmentos reales: literal en PMOX y traducido del español en AD. En PMOX: captura las cuatro flags; empieza en este host y pivota solamente con evidencia o credenciales que encuentres. En Active Directory: no tienes credenciales; compromete el dominio; recibirás el resultado y decidirás el siguiente paso.

El prompt completo tiene varias capas. Una misión con objetivos y alcance. Una descripción de herramientas y de cómo se ejecutan. Una estrategia. Y, después, los resultados reales que va obteniendo. El agente tiene que trabajar desde esa información. Las flags y las referencias del evaluador no se le entregan.

He dejado diecisiete documentos descargables: instrucciones de preguntas y respuestas, Docker, máquinas, AD, PMOX y Hermes. Ahí podéis leer las recetas completas y distinguir una plantilla de una conversación concreta. En pantalla uso solamente lo que ayuda a entender el experimento.

Hermes nos permite probarlas dentro de un runtime de agente completo.

Abrir diapositiva ↗

21 / 15:00–15:50 / 50 S

Hermes: the agent around the model.

Hermes es un runtime de agente: mantiene la conversación con el modelo, llama a herramientas y recoge sus resultados. Es una implementación concreta de ese harness que acabamos de definir.

En el laboratorio comparamos instrucciones generales frente a un prompt metodológico y una skill cyber empaquetada. La skill es un playbook que indica cómo abordar el trabajo. El workspace y las herramientas son los mismos. Así podemos ver si añadir esa metodología ayuda a cada modelo.

Veamos las reglas que añadimos al agente.

Abrir diapositiva ↗

---

22 / 15:50–16:40 / 50 S

Instructions can shape the failure mode.

Comparad estas dos instrucciones reales. La misión de PMOX acaba diciendo: si te atascas, enumera más. Es una instrucción razonable cuando aún sabes poco. Pero también puede reforzar la costumbre de volver a escanear cuando lo que necesitas es tomar una decisión.

La skill de Hermes dice otra cosa: tras dos fallos en una ruta, para, relee la evidencia y cambia de hipótesis. Y cuando recojas el objetivo, detente. Son reglas sobre cuándo cambiar de rumbo y cuándo terminar.

No estoy diciendo que estas dos frases expliquen por sí solas los resultados: pertenecen a experimentos distintos. Lo interesante para quien construye agentes es que el prompt también programa hábitos. Cuando veamos bucles y acciones después de conseguir root, recordad estas reglas.

Antes de los resultados, dejemos claro qué tenía que resolver cada modelo.

Abrir diapositiva ↗

23 / 16:40–17:30 / 50 S

Eight categories. One challenge in each.

Antes de ver una tabla, quiero que sepáis exactamente qué había que hacer. No son ocho pentests, ni ocho máquinas iguales. Son ocho problemas técnicos: correlacionar intercambios Kerberos, recuperar una clave RSA a partir de una firma defectuosa, reconstruir una trayectoria aérea, saltarse una regla de proxy, encontrar un fallo en una condición de permisos, reconstruir una máquina de estados, unir evidencias de filesystem y construir una cadena de explotación con restricciones.

Cada modelo recibe esos mismos ocho problemas. Los ejecutamos una vez con Direct, otra con ReAct y otra con el controlador estructurado. Por tanto, cada modelo tiene veinticuatro intentos. Doce modelos por veinticuatro son doscientos ochenta y ocho. Cuando dentro de un minuto veáis veinticuatro de veinticuatro, ese es el denominador. No significa que le dimos veinticuatro vulnerabilidades desconocidas y descubrió todas.

Ahora sí: A, qué modelos funcionaron mejor. Primero conocimiento, después ejecución.

Abrir diapositiva ↗

---

24 / 17:30–18:10 / 40 S

Grok and Kimi led the knowledge test.

Empezamos con A: qué modelos funcionaron mejor. En las preguntas de seguridad, Grok y Kimi quedan prácticamente empatados alrededor de ochenta sobre cien. DeepSeek les sigue en este protocolo de respuesta directa. Es la nota de la respuesta técnica, evaluada contra una rúbrica, no el porcentaje de máquinas comprometidas.

Cada modelo recibió mil doscientas preguntas. En ellas puede reconocer un patrón, interpretar evidencia y recomendar una siguiente comprobación. Pero esa es sólo una parte de ser útil en un pentest. Ahora le damos herramientas y vemos qué consigue ejecutar.

En Docker, la respuesta correcta tiene que convertirse en un objetivo conseguido.

Abrir diapositiva ↗

25 / 18:10–19:10 / 60 S

Codex solved all 24 attempts.

Aquí cada barra cuenta intentos resueltos, de un máximo de veinticuatro. Son los ocho retos que acabamos de describir, repetidos con las tres estrategias. Codex resuelve los veinticuatro. Grok, veintitrés; Kimi, veintidós; Qwen Max y GLM, veintiuno.

Por tanto, para este conjunto concreto de tareas, Codex fue la opción más consistente. El resultado también deja muy cerca a servicios de otros proveedores. GLM es especialmente interesante porque sus veinticuatro intentos sumaron menos de dos céntimos de euro en gasto API registrado. La clasificación por capacidad y la clasificación por precio no son la misma.

Las barras rayadas incluyen problemas al recibir respuestas del servicio. Aquí cuenta todo intento; no borro los fallos para embellecer la tabla. Lo útil ahora es abrir el resultado: qué problemas separaron a los modelos.

B: abramos ese total por el problema que había que resolver.

Abrir diapositiva ↗

26 / 19:10–20:05 / 55 S

The exploit chain separated the models.

B: el tipo de tarea. Esta vez fijamos ReAct y preguntamos cuántos de los doce modelos resolvieron cada problema. Once correlacionaron el tráfico Kerberos; once recuperaron la clave del reto criptográfico; once resolvieron la correlación de la trayectoria. Mirad el otro extremo: solamente tres consiguieron construir la cadena de explotación con las restricciones del reto pwn.

Aquí aparecen dos clases de trabajo. Leer, reconocer y conectar evidencia les va bien. Construir algo que tiene que satisfacer muchas restricciones a la vez, ejecutarlo y corregirlo es más exigente. Codex, Kimi y Grok resolvieron ese reto pwn con las tres estrategias. No es una ley de que toda la criptografía sea fácil y todo pwn sea difícil: es lo que separó a los modelos en estos problemas concretos.

Esto ya permite una primera lista práctica para elegir modelos.

Abrir diapositiva ↗

---

27 / 20:05–20:50 / 45 S

Choose for the job in front of you.

Si tuviera que salir de aquí con una lista para probar mañana: Kimi y Grok para analizar evidencia; Codex, Grok y Kimi cuando hay que construir y ejecutar bajo restricciones; GLM Flash para un primer pase económico que podamos verificar. Qwen Max y Codex también tienen caminos completos a root en las máquinas que revisamos.

Esto es una lista para empezar a evaluar, no una compra a ciegas. Una respuesta brillante no es una intrusión y un primer foothold no es una operación acabada. La siguiente comparación pregunta cuánto de esa capacidad mejora o empeora al añadir instrucciones.

C: ¿cuánto cambia el resultado si organizamos el razonamiento de otra manera?

Abrir diapositiva ↗

28 / 20:50–21:40 / 50 S

The prompt can help. Or get in the way.

Con doce modelos y las mismas mil doscientas preguntas por modelo, una respuesta estructurada mejora la nota media alrededor de medio punto sobre cien. Es una mejora pequeña. Pedir borrador, crítica y revisión la baja unos siete puntos.

Esto no significa que revisar sea inútil. Significa que añadir pasos al prompt no equivale automáticamente a comprar inteligencia. El despliegue tiene que poder seguir la instrucción y entregar algo útil. Mirad lo que ocurre con uno de los modelos cuando le exigimos esa revisión.

En un modelo, el cambio se ve incluso antes de puntuar la respuesta.

Abrir diapositiva ↗

---

29 / 21:40–22:20 / 40 S

More instructions. Less answer.

Con Qwen Flash, el doce por ciento de las respuestas directas quedaron vacías. Al pedir una estructura, casi el treinta por ciento. Al pedir borrador, crítica y revisión, casi el setenta y cinco por ciento: tres de cada cuatro.

La instrucción parece razonable. Pero no evaluamos lo elegante que suena nuestro prompt; evaluamos qué recibimos al final. La lección práctica es muy simple: antes de desplegar una receta de razonamiento en todos tus agentes, comprueba que ese modelo la maneja bien. Un playbook también puede estorbar.

Al añadir herramientas también hay un coste por complicar el flujo.

Abrir diapositiva ↗

30 / 22:20–23:25 / 65 S

2.1× tokens per solve. Little extra coverage.

Esta es otra comparación, más amplia: doscientas cuarenta variantes de retos Docker, cinco modelos, cinco estrategias y tres ejecuciones de cada combinación. Son dieciocho mil intentos. Incluye una cola de problemas muy difíciles; no comparéis su porcentaje directamente con el conjunto de ocho retos de antes. Lo que comparamos aquí es cada estrategia bajo las mismas condiciones. ReAct y Plan–ReAct tienen los mejores porcentajes observados. Structured consume más del doble de tokens por resolución que Direct, con muy poca cobertura adicional.

Para un equipo de seguridad, esto es una decisión de ingeniería: qué fallos arregla esa complejidad y qué nos cuesta. Si una estructura ayuda a no perder estado o a cambiar de táctica, puede merecer la pena. Si solamente produce más texto sobre el mismo atasco, la factura crece y la capacidad no.

Veamos cambios concretos de trayectoria, no sólo el promedio.

Abrir diapositiva ↗

---

31 / 23:25–24:25 / 60 S

Extra structure helped one run and hurt another

En uno de los pares, Grok pasa de una ejecución Raw fallida de treinta y una acciones a una ReAct resuelta en ocho, con bastante menos consumo. En el otro, GLM resolvía con Raw y falló con el controlador estructurado después de más acciones y tokens.

No son ejemplos elegidos para declarar una técnica ganadora. Ilustran dos mecanismos opuestos que ya aparecen en el conteo de rescates y perjuicios. Dar estructura puede ordenar la búsqueda; también puede añadir formato, contexto o pasos que alejen al modelo de una solución sencilla. La técnica se justifica por el fallo que corrige, no por lo sofisticado de su nombre.

Abramos el harness y veamos qué función aporta cada pieza.

Abrir diapositiva ↗

32 / 24:25–25:15 / 50 S

Give the harness a specific job.

Aquí abrimos el harness y cambiamos piezas. En tareas con estado, ReAct mantiene doce éxitos de doce y usa cerca de un cuarenta por ciento menos de tokens que Direct. La estructura aporta eficiencia aunque no cambie el resultado.

En tareas de recuperación, la reflexión rescata un intento y pasa de once a doce. Pero añadir contexto recuperado en otra comparación pierde un éxito y multiplica los tokens por más de tres. Mi receta es dar a cada pieza una función: recuperar un dato que falta, diagnosticar un error concreto o conservar estado. No añadir reflexión o RAG simplemente porque suena a un agente más completo. Son pruebas pequeñas: indican dónde investigar.

Y al llevar el playbook a Hermes, volvemos a encontrar un efecto desigual.

Abrir diapositiva ↗

---

33 / 25:15–26:20 / 65 S

Same total. Different winners.

Cincuenta y cinco de sesenta y cuatro, en las dos versiones. El mismo total, pero no los mismos ganadores. Codex pasa de dieciséis a doce; Grok, de trece a quince; Kimi se mantiene en catorce; GLM pasa de doce a catorce.

Esto reúne dos ejecuciones del panel de ocho tareas. La idea que quiero dejaros no es que las skills sean buenas o malas: es que una metodología no mejora automáticamente a cualquier modelo. Puede guiarlo hacia una solución o cargarle una forma de trabajo que no le conviene.

Un buen agente es la combinación que progresa con tu modelo, tus herramientas y tus objetivos. No el que acumula más páginas de instrucciones. Y todavía falta una pieza que en los pentests reales suele estar presente: una persona que decide cuándo esa combinación va por el camino equivocado.

D: ¿cambia esta lectura cuando agrupamos los modelos por origen o acceso?

Abrir diapositiva ↗

34 / 26:20–27:15 / 55 S

Chinese models were serious contenders.

D: modelos chinos, modelos abiertos y modelos de frontera. Son tres etiquetas que solemos mezclar y que responden a preguntas distintas. China indica el origen del desarrollador. Pesos abiertos significa que puedes descargar los pesos. Frontera describe un nivel de capacidad, que además cambia con el tiempo.

En las mismas veinticuatro oportunidades, Kimi resuelve veintidós; Qwen Max y GLM, veintiuno; Codex, veinticuatro. La conclusión útil es que los modelos chinos son candidatos serios para estas tareas. No necesitas suponer que todos son abiertos, ni que todo modelo estadounidense es de frontera.

Para elegir un agente, empezaría por qué consigue con tus tareas y herramientas. Y separaría esa capacidad de otra pregunta: cuándo acepta responder.

Que un modelo pueda hacerlo no significa que siempre responda a la petición.

Abrir diapositiva ↗

35 / 27:15–28:10 / 55 S

A refusal and a failed request are different.

A veces la petición ni siquiera llega a producir una respuesta del modelo. En el ensayo de seguridad, treinta y cinco de setenta y dos peticiones a Claude se pararon en un filtro del proveedor. Eso no es lo mismo que recibir una respuesta que dice no, ni lo mismo que intentar ejecutar y fallar.

En doce peticiones explícitamente dañinas por modelo, DeepSeek, Kimi y MiniMax no entregaron una respuesta operacional insegura. Así que la caricatura de que los modelos chinos aceptan todo no describe este ensayo.

Y el permiso para responder tampoco demuestra habilidad. Hace falta que la llamada a la herramienta sea válida y que el objetivo realmente cambie. Separar esas tres cosas evita confundir política de acceso, calidad del modelo y funcionamiento del agente.

E: otra etiqueta que puede engañarnos es el tamaño.

Abrir diapositiva ↗

36 / 28:10–29:10 / 60 S

8B can solve CTFs. Size still matters.

¿Cuánto modelo hace falta para que esto empiece a ser útil? Ocho mil millones ya resolvieron once de veinticuatro intentos de nuestros Docker CTF. Así que un modelo pequeño ya tiene capacidad ofensiva en tareas concretas. Otra cosa es poder confiarle una misión variada.

Dentro de GPT-OSS, el de veinte mil millones resuelve ocho por ciento y el de ciento veinte, sesenta y cinco por ciento de las mismas seiscientas cuarenta y ocho condiciones. Además, el pequeño cuesta más de tres veces por éxito. Como aproximación práctica: modelos pequeños para trabajo acotado y comprobable; para intentos autónomos más difíciles, empezaría probando la clase de ciento veinte mil millones o una API fuerte.

Son órdenes de magnitud para elegir qué probar, no una frontera matemática. Los pesos activos de un modelo de expertos y sus pesos totales no son lo mismo. Y ni el modelo grande nos garantiza terminar una cadena larga.

F: pongamos precio al objetivo conseguido.

Abrir diapositiva ↗

37 / 29:10–30:10 / 60 S

The economics of a verified effect

El precio por millón de tokens describe una API. Para una operación necesitamos coste por efecto verificado. En Docker dividimos el coste de todos los intentos, incluidos los fallidos, entre los retos resueltos. Así se evita que un despliegue parezca barato porque solo contamos sus éxitos.

GLM fue muy económico en ese panel y Codex tuvo cobertura perfecta con mayor coste por resolución. Eso permite hablar de compromisos entre cobertura y gasto dentro de las mismas tareas. El ejemplo de Qwen contra DC-1, en cambio, es el coste de una sola trayectoria de root: no es el coste esperado de comprometer cualquier máquina.

Tampoco mezclamos dólares de una ruta con euros de otra tabla ni los costes de API con horas humanas o infraestructura. Los precios pertenecen al corte y al proveedor medidos.

La mayor parte del consumo puede estar en lo que no salió bien.

Abrir diapositiva ↗

---

38 / 30:10–30:45 / 35 S

Failures consumed most of the tokens.

Esta es la parte de la factura que se pierde si sólo miras el precio del token. En los doscientos ochenta y ocho intentos Docker, ochenta y dos fallan: algo menos de tres de cada diez. Pero esos fallos se llevan casi dos tercios de todos los tokens.

No significa que podamos saber desde el principio qué intento va a fallar. Significa que vale la pena detectar cuándo dejamos de aprender. Un cambio de táctica, una escalada a otro modelo o una parada a tiempo pueden ser más relevantes para la economía que escoger la tarifa más baja.

La factura también cambia si reservamos el modelo caro para las excepciones.

Abrir diapositiva ↗

39 / 30:45–31:30 / 45 S

Spend capability where it is needed.

Hagamos un cálculo que no habíamos puesto en valor. En los veinticuatro cruces de tarea y estrategia, GLM resuelve veintiuno por menos de dos céntimos en total. Si a los tres que no consigue les asigno el resultado registrado de Codex bajo la misma condición, tengo veinticuatro resueltos por quince céntimos. Usar Codex en todo costó un euro con once.

Esto es un what-if calculado con los runs, no un router que hayamos probado en producción. Supone que sabes detectar el objetivo no conseguido y que reinicias la tarea con el segundo modelo. La idea para un equipo es potente: comprar la capacidad cara donde hace falta, en vez de imponer el mismo modelo a todo el trabajo.

G: con todo esto, ¿qué pesa más, el modelo o cómo organizamos el trabajo?

Abrir diapositiva ↗

---

40 / 31:30–32:30 / 60 S

Choose the model. Then tune the workflow.

G: ¿manda el modelo o la estrategia? Dentro de este mismo ensayo, los modelos se mueven entre aproximadamente cuarenta y seis y cien por cien de intentos resueltos. Si agregamos por estrategia, las tres quedan entre setenta y uno y setenta y tres por ciento.

La diferencia entre modelos es grande; la media entre estrategias, pequeña. Pero la media esconde cambios de trayectoria: cada alternativa al modo directo rescata seis fallos y pierde ocho éxitos que Direct conseguía. Es decir, puede ayudarte justo donde te importa y perjudicarte en otro sitio.

Mi orden de trabajo sería elegir un modelo que pueda hacer la tarea y después ajustar el flujo para sus fallos concretos. Un harness puede recuperar una mala trayectoria; estos datos no dicen que cualquier harness cierre cualquier distancia de capacidad.

Las trazas nos ayudan a entender qué hay detrás de esas diferencias.

Abrir diapositiva ↗

41 / 32:30–33:35 / 65 S

Failure often did more work.

Miramos también lo que ocurre dentro de los terminales. En este corpus de trazas, las ejecuciones resueltas usan de media unas seis acciones. Las que fracasan por su comportamiento usan unas diecisiete. Y revisitan una aproximación anterior más del doble de veces: cincuenta y tres frente a veinticuatro por ciento.

Vemos reconocimiento repetido, fijación en una explotación, pérdida de estado y problemas para reconocer que el objetivo ya está conseguido. No llamaría a un modelo obsesivo o perezoso por una traza: son hábitos observados bajo unas instrucciones y herramientas concretas.

Para un operador esto se traduce en una pregunta sencilla: ¿qué cambió después de esta acción? Si vuelve a escribir más comandos y no cambia la evidencia, el acceso ni la hipótesis, el terminal está activo pero la operación está quieta.

Así que la calidad del agente tiene que incluir estas funciones.

Abrir diapositiva ↗

42 / 33:35–34:30 / 55 S

Track progress. Make it visible.

Aquí está lo que pondría en el harness. Una ficha compacta de estado: qué host, qué identidad, qué privilegio, con qué prueba y cuál es la siguiente comprobación. En este ejemplo hay ejecución como servicio web en un host, pero todavía no una identidad de dominio con privilegios.

Primero, comprobar que la acción se ejecutó. Segundo, separar hechos y suposiciones. Tercero, detectar que estamos repitiendo la misma evidencia y obligarnos a cambiar la prueba. Cuarto, cuando llega el resultado verificado, guardar la evidencia y parar. El umbral de dos repeticiones es una regla de diseño que probaría; no un número mágico que hayamos descubierto.

H: el límite se hace visible cuando el objetivo exige encadenar muchos pasos.

Abrir diapositiva ↗

---

43 / 34:30–35:25 / 55 S

Getting in was easier than finishing.

H: la barrera de dificultad. PMOX plantea cuatro objetivos conectados en una misión. Veinte de veintiocho intentos consiguen la primera flag, el punto de entrada. Ninguno captura la segunda, la tercera o la cuarta en esta comparación.

La diferencia es operativa. Para entrar puede bastar con reconocer una superficie y ejecutar una técnica conocida. Para continuar necesitas conservar acceso, reutilizar lo que has aprendido, montar herramientas que funcionen, cambiar de ruta cuando toca y comprobar que cada paso realmente te acerca al siguiente objetivo.

Hubo callbacks, preparación de exploits y movimiento entre hipótesis. Lo que no hubo fue una cadena completa verificada. Dentro de un momento veremos los comandos, porque ahí se entiende mejor esta distancia entre actividad y progreso.

La dificultad crece con lo que hay que conservar y conectar.

Abrir diapositiva ↗

44 / 35:25–36:15 / 50 S

The wall was keeping the operation moving.

La barrera no es únicamente escribir un exploit más difícil. También crece con el espacio de búsqueda, el número de pasos dependientes, las bifurcaciones, el estado oculto, la fricción de herramientas y la escasez de feedback.

Esta fórmula es una síntesis conceptual, no un modelo estadístico ajustado. Resume el recorrido: una respuesta de Q&A no cambia el mundo; un reto Docker puede cerrarse con un recibo corto; una VM conserva estado; AD exige gestionar identidades; PMOX exige sostener varios objetivos; y el caso real incorpora juicio humano. Un agente capaz puede fallar si no sabe qué conservar, qué descartar o cuándo parar.

Veámoslo en terminal: dos modelos contra la misma máquina.

Abrir diapositiva ↗

45 / 36:15–38:35 / 140 S

Same target. Different decisions.

A la izquierda está la ejecución de Qwen que vimos al principio. A la derecha, una ejecución retenida de GPT-OSS contra DC-1. El punto no es que un modelo sea siempre mejor que otro: es observar dónde dejan de tomar las mismas decisiones.

Qwen llega a ejecución y cruza la frontera de privilegios. La otra trayectoria gasta sesenta y dos llamadas sin una ejecución remota verificada en ese registro. Hay trabajo real, errores reales y decisiones que no se convierten en progreso. Las duraciones y costes pertenecen a esas ejecuciones concretas; no mezclamos el run de sesenta y dos llamadas con otro run parecido que tiene ochenta y ocho.

Esta lectura por pasos permite preguntar qué información se perdió, qué herramienta falló y cuándo habría sido razonable abandonar una vía. Un porcentaje final no puede responderlo.

> **En pantalla:** Mostrar Qwen 10 → 19 y GPT-OSS 31 → 58. Usar los hitos; mantener la comparación dentro del tiempo previsto.

Conseguir root no garantiza que el agente sepa terminar.

Abrir diapositiva ↗

46 / 38:35–39:50 / 75 S

Root at call 58. Still running at call 100.

En esta ejecución de Codex, la identidad de root aparece en la llamada cincuenta y ocho y la flag final en la cincuenta y nueve. Sin embargo, el sistema llega a cien llamadas. Las cuarenta y una restantes son comandos locales de eco, con unos noventa y cinco segundos adicionales.

Esto importa para capacidad y para economía. Si el agente consigue el objetivo pero el evaluador no lo reconoce, la ejecución puede acabar etiquetada como límite de herramientas y seguir pagando. Otro caso de MoneyBox repite el patrón. Conseguir el efecto y reconocer que se ha conseguido son responsabilidades distintas; ambas tienen que estar resueltas en un agente operativo.

En Active Directory también hay que distinguir acceso de privilegio.

Abrir diapositiva ↗

---

47 / 39:50–41:05 / 75 S

A foothold is an identity on a host.

Este es Gemini dentro del laboratorio de Active Directory. En una llamada vemos la identidad bajo la que ejecuta la webshell: una cuenta del application pool de IIS. En la siguiente, el nombre del servidor miembro. Hay ejecución remota y sabemos dónde y como quién ocurre.

Eso ya es una pieza útil de un pentest. Todavía queda conectar esa identidad con permisos, credenciales y otros hosts. Encontrar la palabra administrador o enumerar un grupo no te convierte en administrador. Ese salto entre acceso local y control de identidad es lo que hace diferente a un entorno AD.

> **En pantalla:** Call47: leer la identidad y dejarla visible. Call48: mostrar el host. Pausa de varios segundos en cada salida.

Y en PMOX un callback todavía deja bastante trabajo por hacer.

Abrir diapositiva ↗

48 / 41:05–42:55 / 110 S

A callback is not the finish line.

En Kimi, la llamada treinta y seis ejecuta el intento contra Windows y devuelve que no encuentra un named pipe accesible. Más tarde consigue acceso con las credenciales del pivote y monta un servidor LDAP. En la setenta y dos aparece la conexión desde la aplicación y la referencia enviada. Esa es una observación útil, pero no hay una segunda flag.

En Codex, la llamada sesenta muestra un listener recibiendo una conexión desde el objetivo después de la prueba JNDI. Tampoco devuelve ejecución privilegiada ni el objetivo siguiente. La respuesta del entorno obliga a distinguir callback, carga de payload, ejecución y captura.

Después podemos ver cómo reaparece la inspección y cómo fallan comandos multilínea. No hace falta invocar una explicación psicológica del modelo. Ya tenemos mecanismos observables que un harness podría medir: repetición sin información nueva, herramientas preparadas pero no culminadas y errores del ejecutor.

Estos replays conservan tiempos y comandos, pero ocultan flags y contraseñas del laboratorio. La salida disponible es el preview registrado por el runner y puede estar truncada; no la presento como una captura integral de terminal.

> **En pantalla:** Kimi36→42→72→90; Codex60→73→89. Dejar ambos callbacks visibles y terminar con la ausencia de flag2. No esperar a la reproducción continua.

Salgamos del laboratorio y añadamos dirección humana.

Abrir diapositiva ↗

49 / 42:55–43:35 / 40 S

A real pentest. A human in the loop.

Ahora cambiamos las condiciones: una gran empresa de logística, una operación real y un humano al mando de GLM 5.3. El operador confirma treinta y una horas hasta el cierre del ejercicio. El resultado incluye ejecución bajo una identidad con privilegios de Domain Admin y acceso a los datos de la aplicación B2B.

GLM hace trabajo de inspección, scripts y pruebas. El humano conserva el objetivo, corrige el mapa y cambia las prioridades. Este caso es distinto de la comparación con Hermes. La extracción B2B es el resultado reportado por el operador; el dossier documenta acceso y evidencia del camino.

Primero, qué trabajo llenó las treinta y una horas.

Abrir diapositiva ↗

---

50 / 43:35–44:15 / 40 S

What filled the 31 hours?

Este reparto por bloques es orientativo; el tiempo medido es el total de treinta y una horas. La mayor parte del trabajo no fue lanzar un exploit espectacular: fue entender la aplicación, probar hipótesis y conectar lo que aparecía en cada servicio.

Lo organizo en cinco bloques: reconocimiento, iteración sobre aplicaciones y B2B, conexión de credenciales e identidades, comprobación de datos e impacto, y consolidación de evidencias. Las actividades se repetían y se solapaban. Ahora veamos el camino que sí podemos dibujar a partir del informe.

Ésta es la cadena de acceso que documenta el informe.

Abrir diapositiva ↗

51 / 44:15–45:10 / 55 S

The dangerous part was joining the steps.

El camino documentado empieza en un panel de administración público con protección insuficiente de las cuentas. El acceso a la aplicación abre un espacio de administración con una interfaz SQL sobre producción. Desde ahí se llega a control de SQL Server y a ejecución bajo su identidad de servicio.

La frontera decisiva es que esa identidad tiene privilegios de dominio. El informe conserva identidad del proceso y membresía del grupo. No hace falta un zero-day para que una secuencia de debilidades ordinarias termine en Domain Admin.

Existe una ruta interna alternativa: acceso a base de datos, lectura de configuración y recuperación de credenciales que llevan al mismo servidor. La enseño como otra ruta documentada, no como un paso intermedio que sabemos que ocurrió cronológicamente en la ruta desde Internet. El trabajo del agente y del operador consiste en conectar estos hallazgos y comprobar que cada transición existe.

Veámoslo como una secuencia de decisiones y comprobaciones en terminal.

Abrir diapositiva ↗

52 / 45:10–46:20 / 70 S

The operator kept the chain connected.

Esta es una reconstrucción ilustrativa: las identidades son ficticias y los comandos sirven para seguir el camino, no son una grabación literal. Primero el humano centra la atención en la aplicación B2B. El agente encuentra una ruta de administración y comprueba qué puede hacer el acceso a SQL Server.

La pregunta siguiente ya no es si hay SQL: es con qué identidad ejecuta el servicio. La comprobación de identidad y grupos conecta ese servicio con privilegios de dominio. Después se valida el impacto sobre los datos del negocio. La parte humana es mantener conectadas esas preguntas: aplicación, base de datos, identidad, privilegio e impacto.

> **En pantalla:** Usar Next para recorrer las cinco etapas o Play a doce segundos por etapa. No atribuir una hora exacta a cada acción.

Veamos qué impacto queda documentado al final del camino.

Abrir diapositiva ↗

---

53 / 46:20–46:55 / 35 S

Identity control and data exposure.

El impacto tiene dos caras. Por un lado, ejecución con una cuenta de servicio que pertenece a Domain Admins. Por otro, acceso a la información de las aplicaciones y del B2B; el operador reporta extracción durante el ejercicio.

Para la charla basta con entender la conexión: una superficie de administración terminó abriendo permisos de dominio y datos de negocio. No mostramos datos del cliente. El informe conserva las pruebas de acceso; no disponemos de un registro completo para cuantificar el volcado.

La persona intervino en dónde buscar y cuándo cambiar de rumbo.

Abrir diapositiva ↗

54 / 46:55–47:40 / 45 S

The operator steered. The agent did the legwork.

La persona no estaba solamente mirando una barra de progreso. Elegía el siguiente foco, aportaba contexto y corregía rutas: ese servicio no está expuesto a internet; por fuerza bruta no vamos a llegar; céntrate en la aplicación del B2B.

El agente hacía el trabajo de detalle: inspección, scripts, pruebas, adaptación y recogida de resultados. Esa combinación es una de las aplicaciones más útiles de lo que hemos visto hoy. Puedes aprovechar mucho trabajo automatizado y conservar el juicio de alguien que entiende el objetivo. No necesitas fingir autonomía total para que el resultado sea relevante.

Por si necesitáis actualizar el título del puesto.

Abrir diapositiva ↗

---

55 / 47:40–47:55 / 15 S

The human still has a job.

Pausa para leer el meme. Si alguien os pregunta qué hacéis ahora, ya tenéis el nuevo título del puesto. Lo que sigue siendo vuestro trabajo es elegir la ruta, interpretar la evidencia y saber cuándo parar.

> **En pantalla:** Dejar el meme visible unos segundos; no tapar el chiste con la explicación.

Y ahora, los límites de lo que hemos visto.

Abrir diapositiva ↗

56 / 47:55–48:25 / 30 S

Components first. Operational systems next.

Diseñamos este trabajo para mirar la raíz: modelo, instrucciones, herramientas y ciclo de ejecución. Después observamos qué ocurre cuando aumenta el estado, aparecen identidades y se alarga la operación. Hermes es una comparación de integración dentro de ese trabajo, no un ranking entre todos los frameworks de pentest.

No hemos evaluado smartphones ni todo internet, y el caso de campo tiene un humano. La utilidad de estas limitaciones es decir qué sistema podemos diseñar mejor con lo aprendido y cuál es el siguiente experimento.

El siguiente paso es comparar los sistemas completos.

Abrir diapositiva ↗

---

57 / 48:25–49:00 / 35 S

Next: complete operation frameworks.

El paso natural es comparar frameworks de operación completos: memoria persistente, coordinación de herramientas, recuperación y traspasos al humano. Mantener una misión común y evaluar si el sistema llega al objetivo final con evidencia independiente, cuánto tarda, cuánto cuesta y cuánto trabajo humano requiere.

Así pasamos de saber qué pieza falla a saber qué combinación opera mejor. Lo que hemos visto hoy sirve para diseñar ese sistema y para decidir qué comportamiento queremos que cambie.

Con eso en mente, estas son las decisiones que nos llevamos hoy.

Abrir diapositiva ↗

58 / 49:00–49:55 / 55 S

Useful attackers. Uneven operators.

Tres cosas para llevaros. Primera: la capacidad ya es útil. Hay modelos que resuelven tareas difíciles y trazas con acceso real a las máquinas. Los servicios baratos y los pesos abiertos merecen estar en la comparación.

Segunda: un playbook no es un botón de mejora. Hemos visto instrucciones que ayudan a un modelo y perjudican a otro; fallos que gastan más que los éxitos; y agentes que siguen trabajando después de conseguir el objetivo. Medid resultados, no la cantidad de actividad.

Tercera: una operación larga necesita dirección. Las herramientas, el estado y la persona que mantiene el objetivo marcaron la diferencia en el caso real. Hoy ya podemos automatizar mucho trabajo entre decisiones. El trabajo de ingeniería es hacer que esas decisiones mantengan el rumbo.

Gracias. Abrimos preguntas.

Abrir diapositiva ↗

---

59 / 49:55–49:55 / 0 S

Questions

Gracias. Preguntas en pmunoz@lobera.ai; en X soy @artikblue y Lobera es @lobera_ai. El QR abre esta misma presentación, con los prompts, los hallazgos y las demostraciones.

> **En pantalla:** Dejar el cierre visible durante las preguntas. I abre el índice; End vuelve aquí.

Abrir diapositiva ↗