The Agent Harness

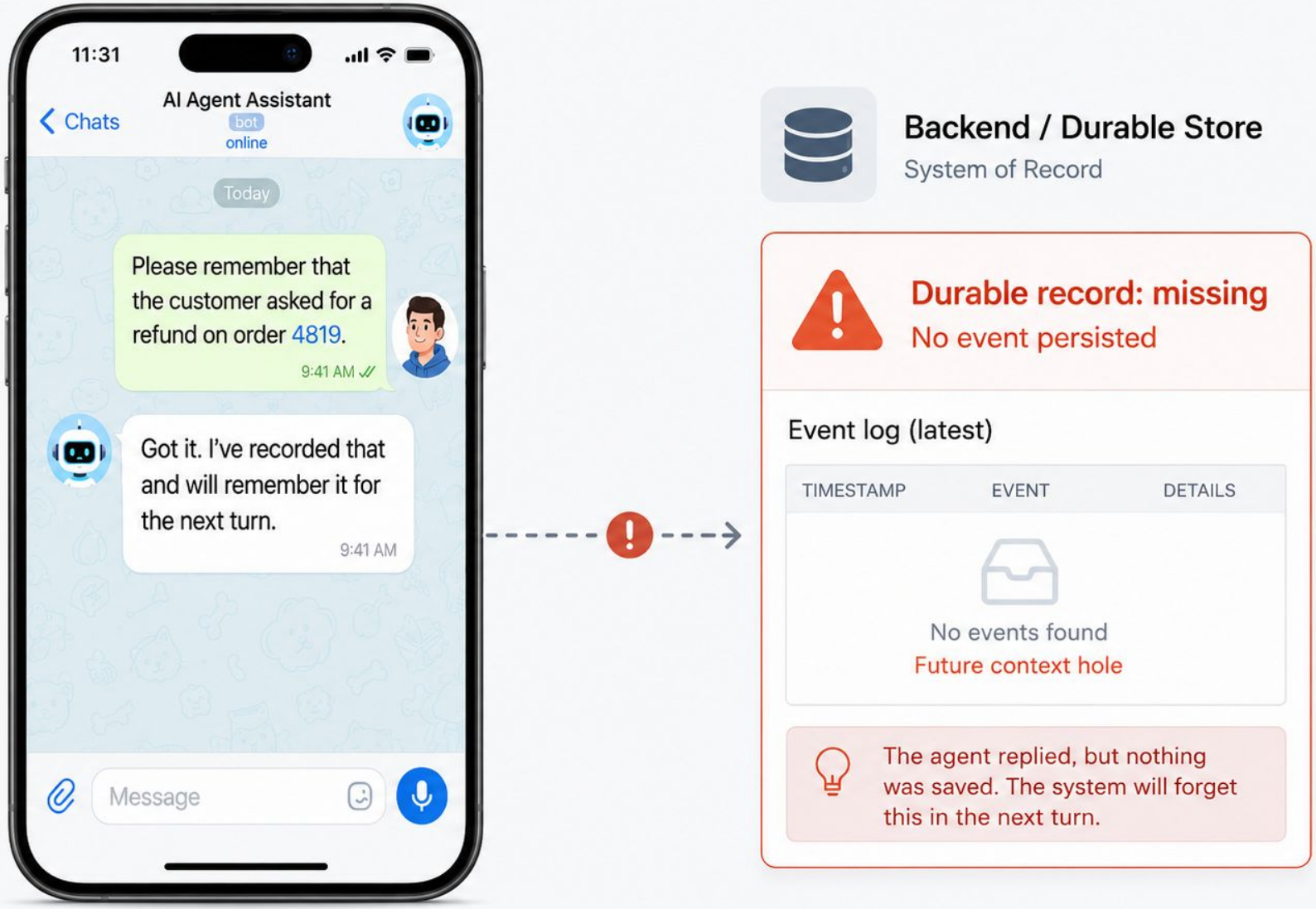
Control Planes, Invariants, and Approval Boundaries for Production AI Agents

Vinoth Govindarajan - MTS @ OpenAI

https://www.linkedin.com/in/vinothgovindarajan/

The user saw the reply.
The system forgot it happened.

A production incident in one sentence.


11:31
AI Agent Assistant
bot
online
Chats
Today
Please remember that the customer asked for a refund on order 4819.
9:41 AM
Got it. I've recorded that and will remember it for the next turn.
9:41 AM
Message
Backend / Durable Store
System of Record
Durable record: missing
No event persisted
Event log (latest)
TIMESTAMP
EVENT
DETAILS
No events found
Future context hole
The agent replied, but nothing was saved. The system will forget this in the next turn.

⚠️ Why this is worse than a crash

A crash gives you a boundary. Silent success gives you a lie.

Delivery

The channel said success.

Persistence

The turn was not recorded.

Future context

The next run inherited a hole.

Source example: OpenClaw #82287

❓ The production question

When text becomes action, who owns the truth?

🗂️ State

Who owns the fact?

🔀 Order

Who committed first?

🧾 Proof

Who can show what happened?

👋 Hi, I am Vinoth

I bring distributed-systems lens to agents infrastructure

- Building core data/AI infra at OpenAI
- Built distributed systems at Apple & Uber
- Built full-stack applications for a decade
- Contributed to many open-source projects
- Writes The Agent Stack on Substack
- Co-author of Engineering Lakehouses with Open Table Formats Book
- Studied OpenClaw architecture and incidents

🧭 Three things to remember

Own the state. Order the mutation. Prove the action.

1. 🗂️ Own the state

A fact needs one owner and one replay path.

2. 🔀 Order the mutation

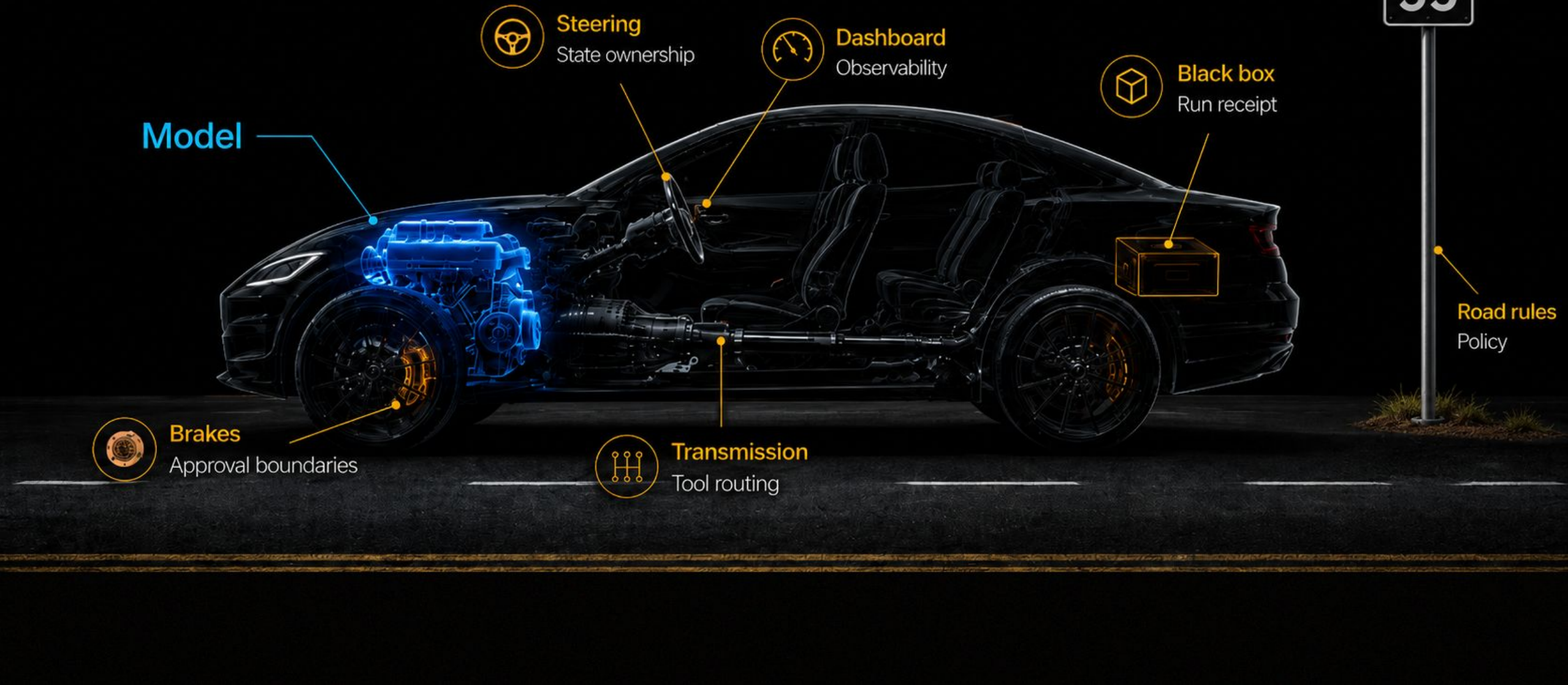
Concurrency is fine. Accidental interleaving is not.

3. 🧾 Prove the action

The transcript is not the receipt.

The model is the engine.

The harness is everything that lets the engine move safely in the real world.



🤝 The production contract

A model proposes. The harness commits. The receipt proves it.

🤖 Model

proposes

🧰 Harness

commits

🧾 Receipt

proves

🦞 OpenClaw is the case study. 🤝 The contract is the takeaway.

🧠 Why old failures get sharper in agents

Familiar reliability bugs sit around a probabilistic planner.

🧭 Dynamic plans

the next step is chosen at runtime

🎲 Probabilistic output

the path may differ between turns

📡 More event sources

users, tools, timers, webhooks, subagents

🛠️ More action surfaces

messages, files, APIs, browsers, workflows

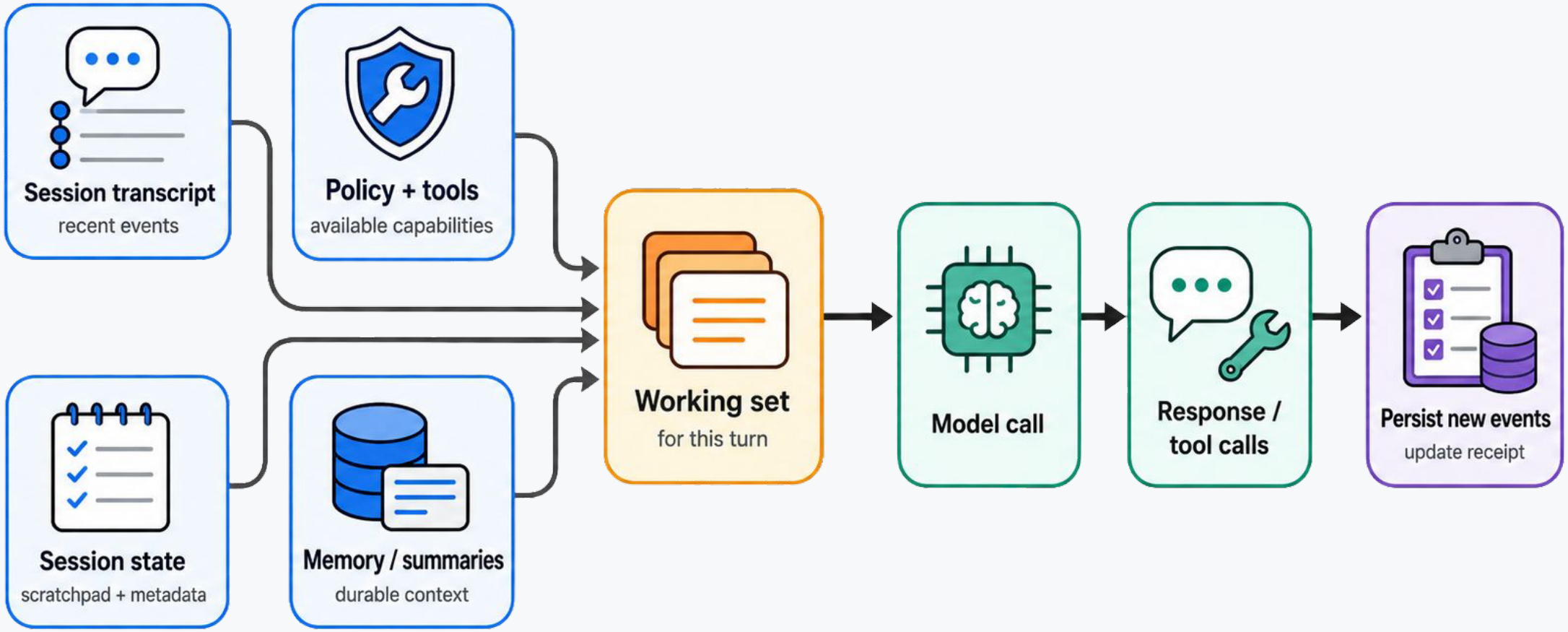
The failures are familiar.
The agent setting makes them easier to trigger and harder to explain.

🗺️ The harness blueprint

Same pattern across frameworks. Different product names.

🔁 Turn start: rehydrate state

The runtime does not “remember.” It rebuilds the working set for this turn.



1

🗂️ Own the state

Every fact needs one owner and one replay path.

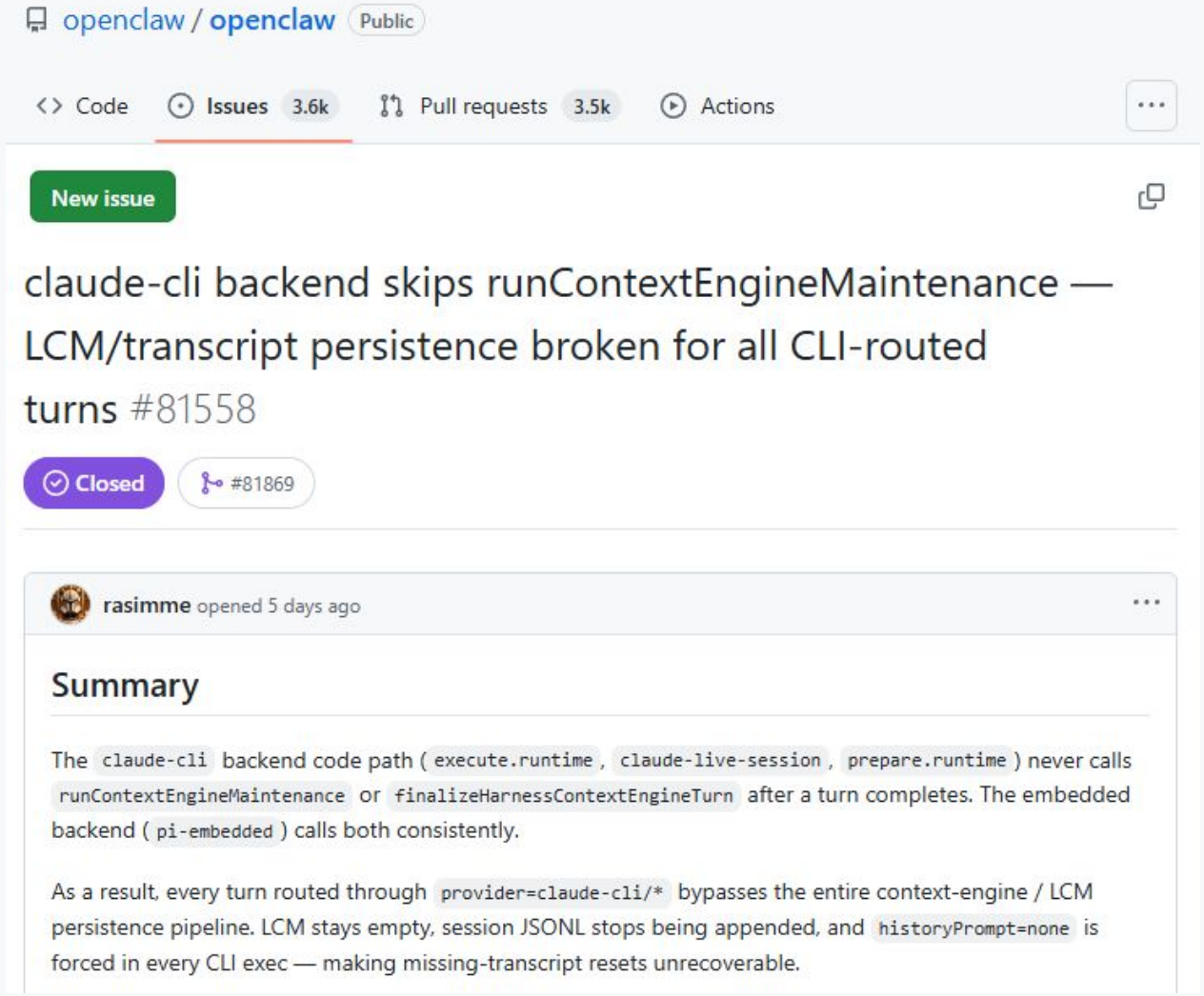
Incident: success without memory

The user saw the reply. The system forgot it happened.

User sees
the assistant replied

System state
turn was not recorded

Harness lesson
delivered is not remembered



OpenClaw #82287, #81558

Harness Pattern: state is not storage

State is the ownership boundary for recovery.

Transcript
which log is appended?

Memory pipeline
which process ingests it?

Replay
which owner can reconstruct it?

Owner examples: Event → Calendar system • Support → Ticketing system • Email → Email provider
Code → Repository/workspace • Preference → Memory store • Conversation turn → Session transcript

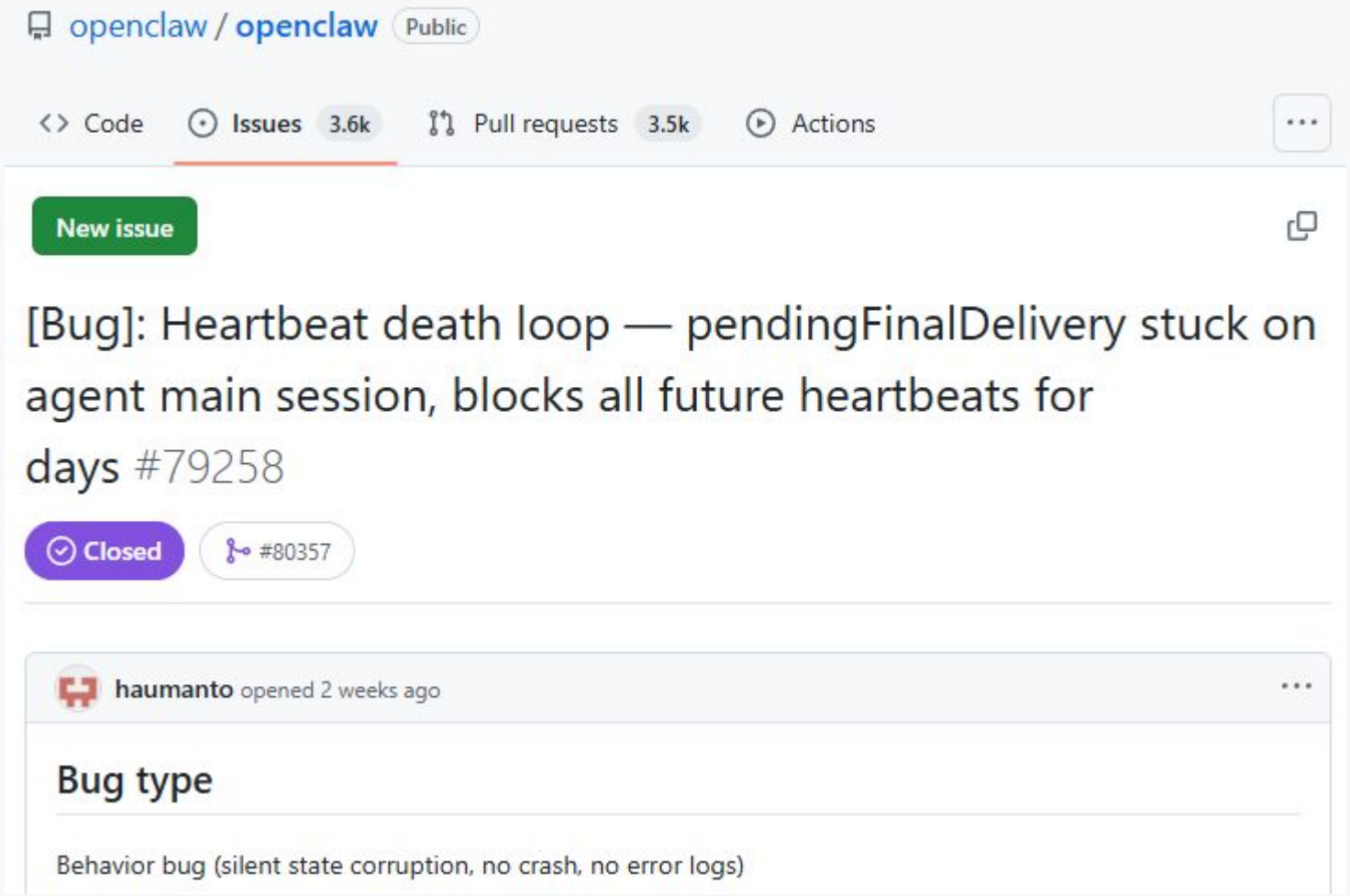
💓 OpenClaw example: heartbeat became state

A liveness token crossed the wrong boundary.

User sees
agent goes quiet for hours

System state
HEARTBEAT_OK becomes pending delivery

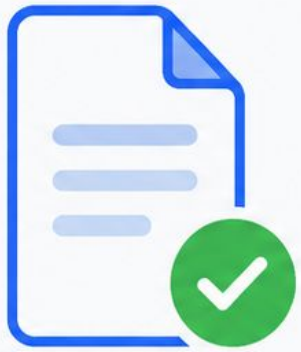
Harness lesson
control signals are not user work



OpenClaw #79258, PR #80357

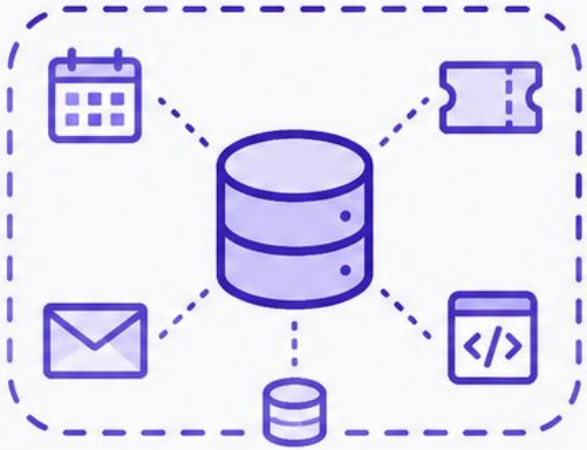
✅ General lesson

A reply is not reliable memory until a named owner can replay it.



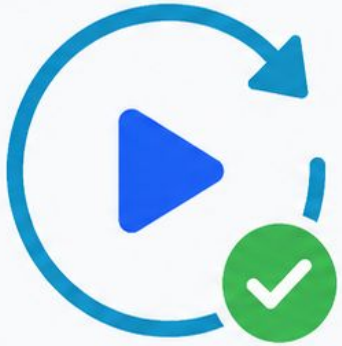
Persist the turn

capture the fact



Name the system of record

where this fact belongs



Make replay possible

prove recovery

2

🔀 Order the mutation

If two writers can touch it, the harness owns the order.

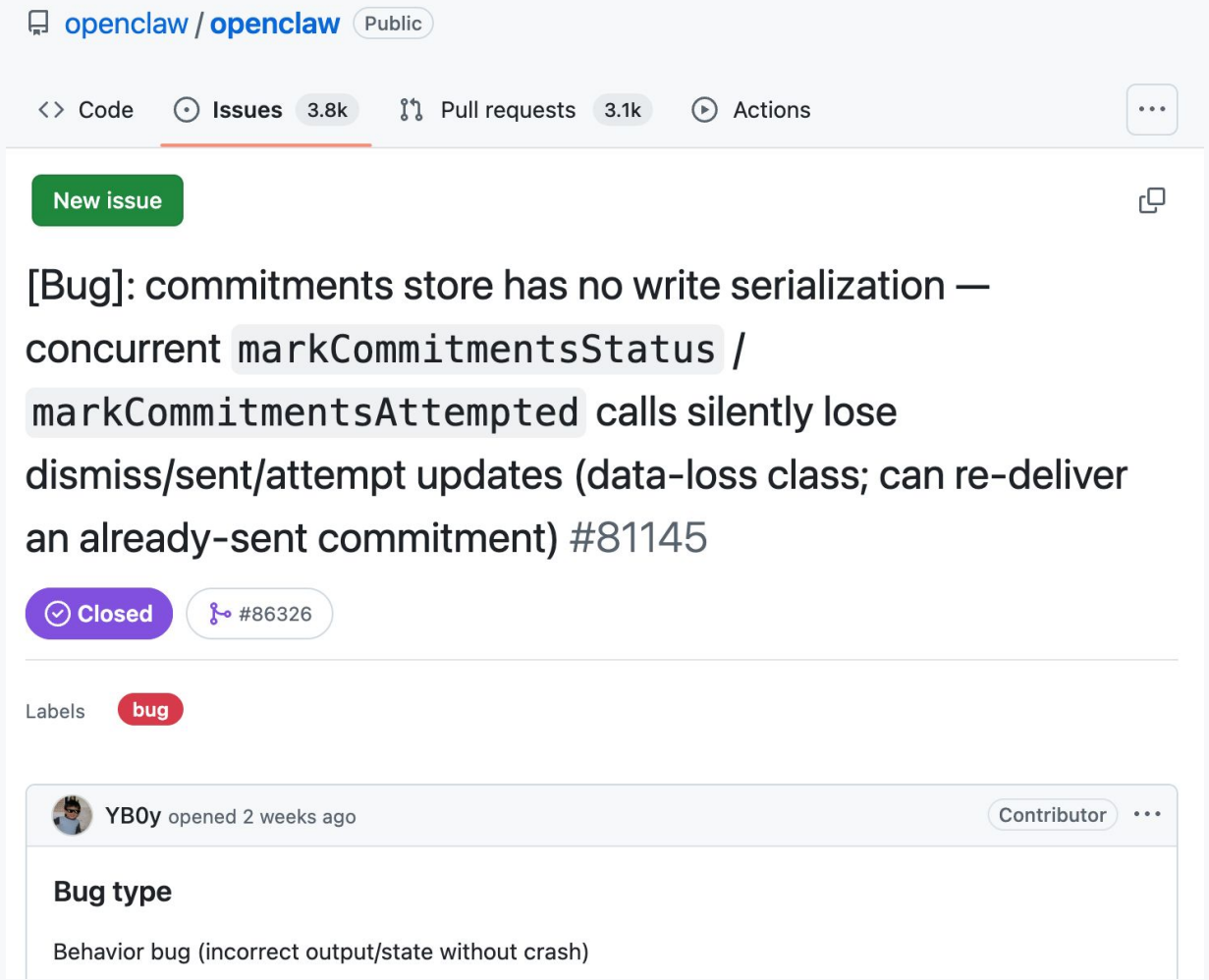
✍️ Incident: two correct writes

Last writer wins is not a consistency model.

User sees
a correction disappears

System state
two writers save competing versions

Harness lesson
serialize mutable state



OpenClaw PR #81145

🛤️ Harness Pattern: one ordered commit path

The invariant is not no concurrency.

Parallelize reads

safe work can fan out

Fan out subtasks

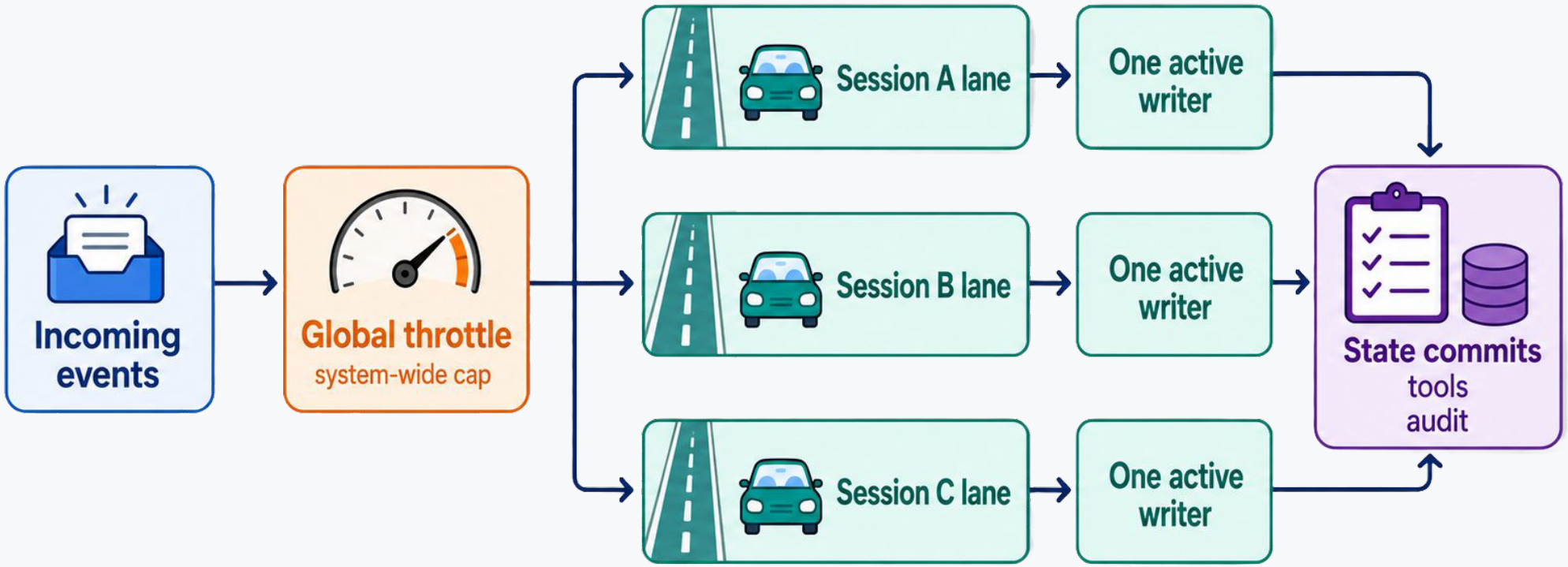
inside one run

Serialize mutation

one commit path per mutable state boundary

Lanes make concurrency safe

Parallel across sessions. Single writer inside a session. Globally throttled across the system.



✅ General lesson

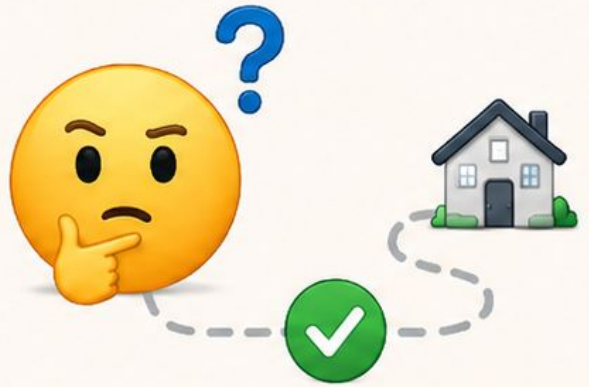
Users experience ordering bugs as agent behavior.

Agent feels forgetful

serialize shared writes

Agent feels dead

detect stuck lanes



Agent feels confused

treat completion as delivery

3

⏳ Bound the work

No tool call, lane, loader, or stream gets infinite time.

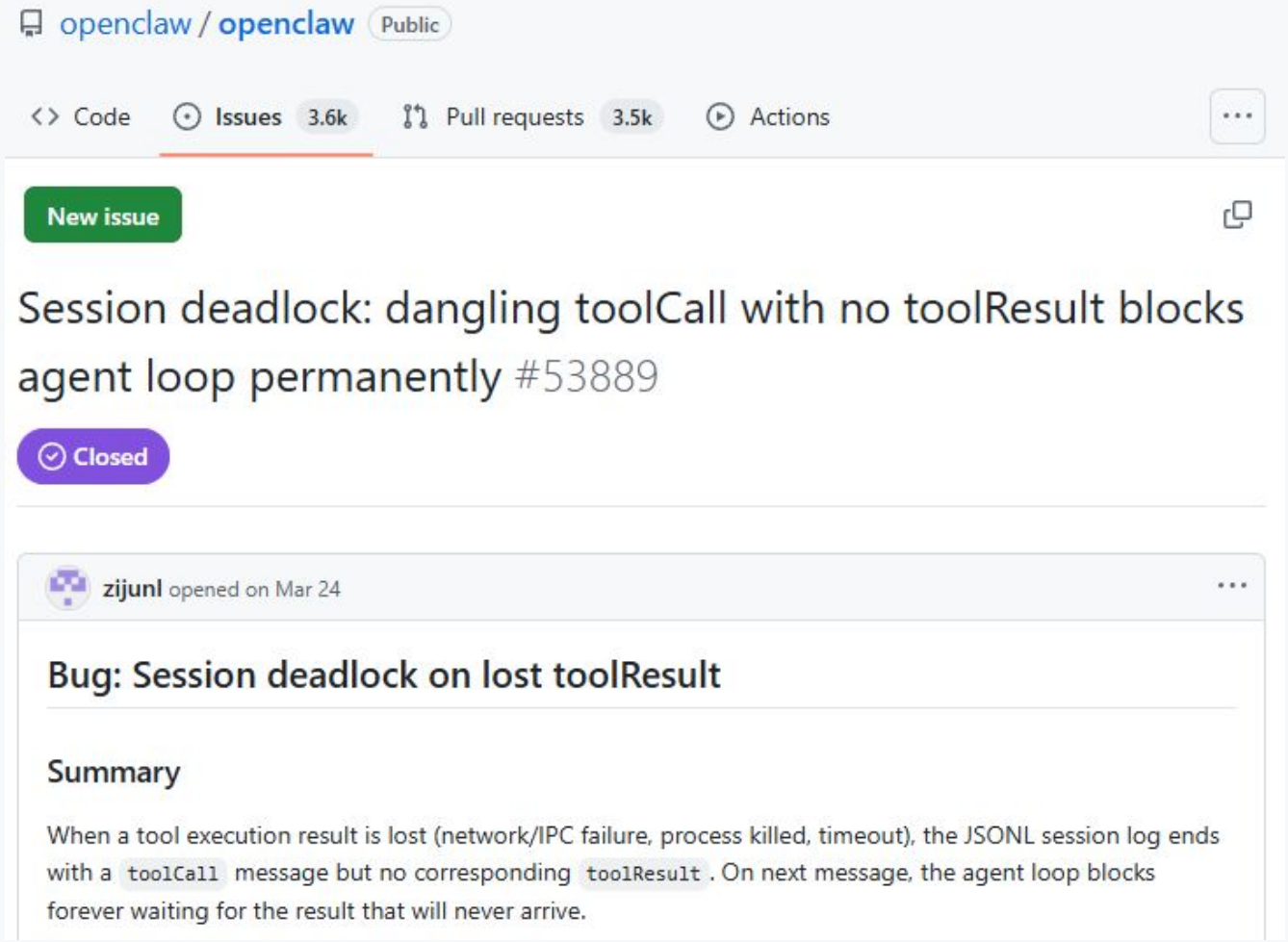
Incident: dangling tool call

Silence is not a terminal state.

User sees
agent gets stuck

System state
toolCall has no toolResult

Harness lesson
silence needs a terminal event



OpenClaw #53889

⏱️ Harness Pattern: every boundary needs an ending

Success. Failure. Timeout. Cancel. Max attempts.

Lease
deadline for runs and tools

Watchdog
stuck work becomes visible

Interrupt
recovery controls live outside the stuck run

Receipt
timeouts write terminal records

✅ General lesson

Bound the work before the work bounds you.

Runs

deadline and cancellation

Tools

timeout and error result

Channels

recovery commands do not wait behind stuck work

4

 Constrain authority

Approval is a scoped object, not a memory of a click.

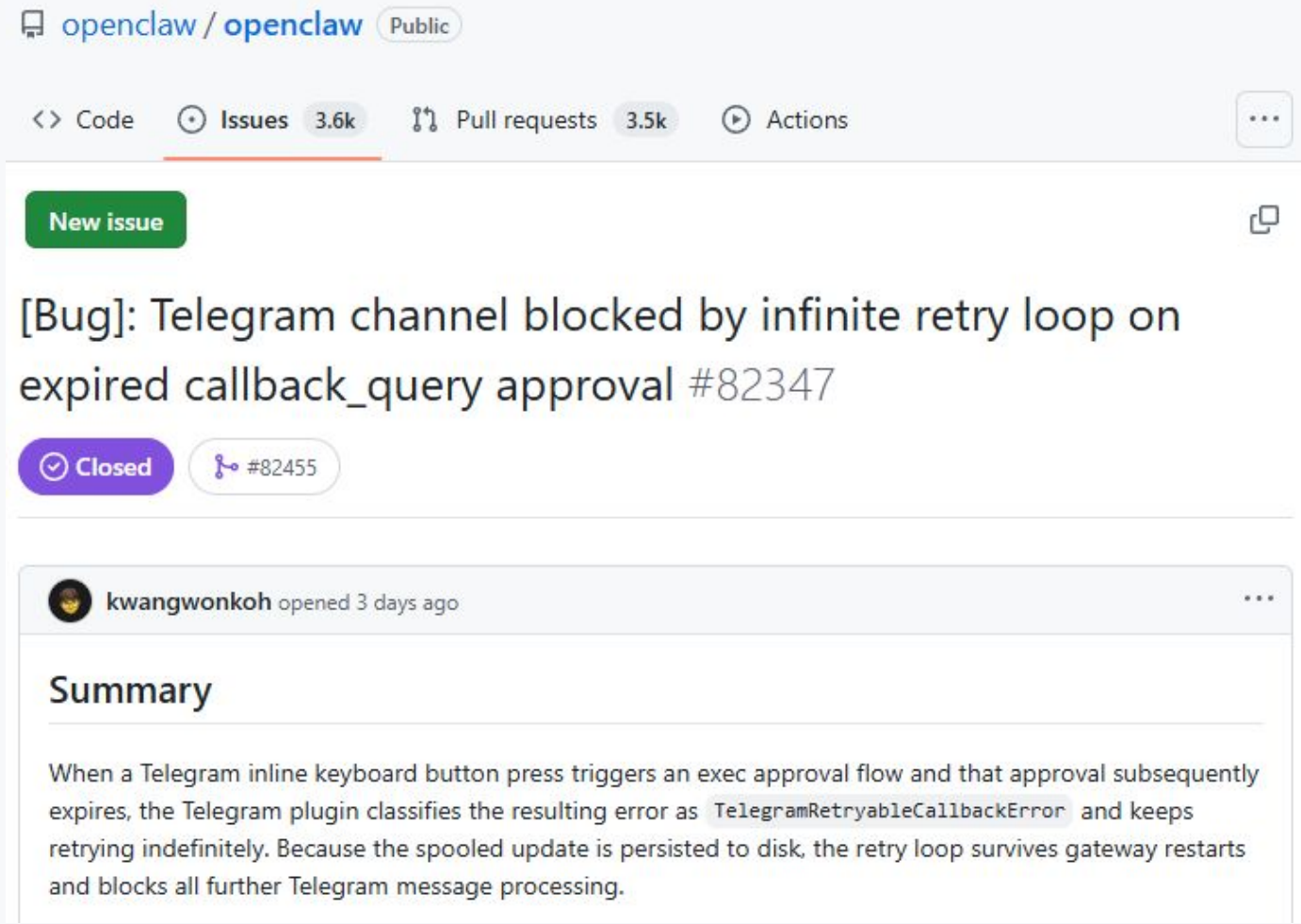
🔐 Incident: approval lost its scope

The click was real. The execution context was not.

User sees
I approved this; why blocked?

System state
approval context did not survive

Harness lesson
approval needs scope



OpenClaw PR #75979, #82347

Harness Pattern: authority has shape

Actor. Session. Tool. Arguments. Lifetime. Outcome.

Tool offered to model
requestable, not executable

Approval
binds to one pending action

Expiry
must terminate, not loop

🧰 OpenClaw example: capability drift

A disabled tool that still registers is not disabled.

Policy surface

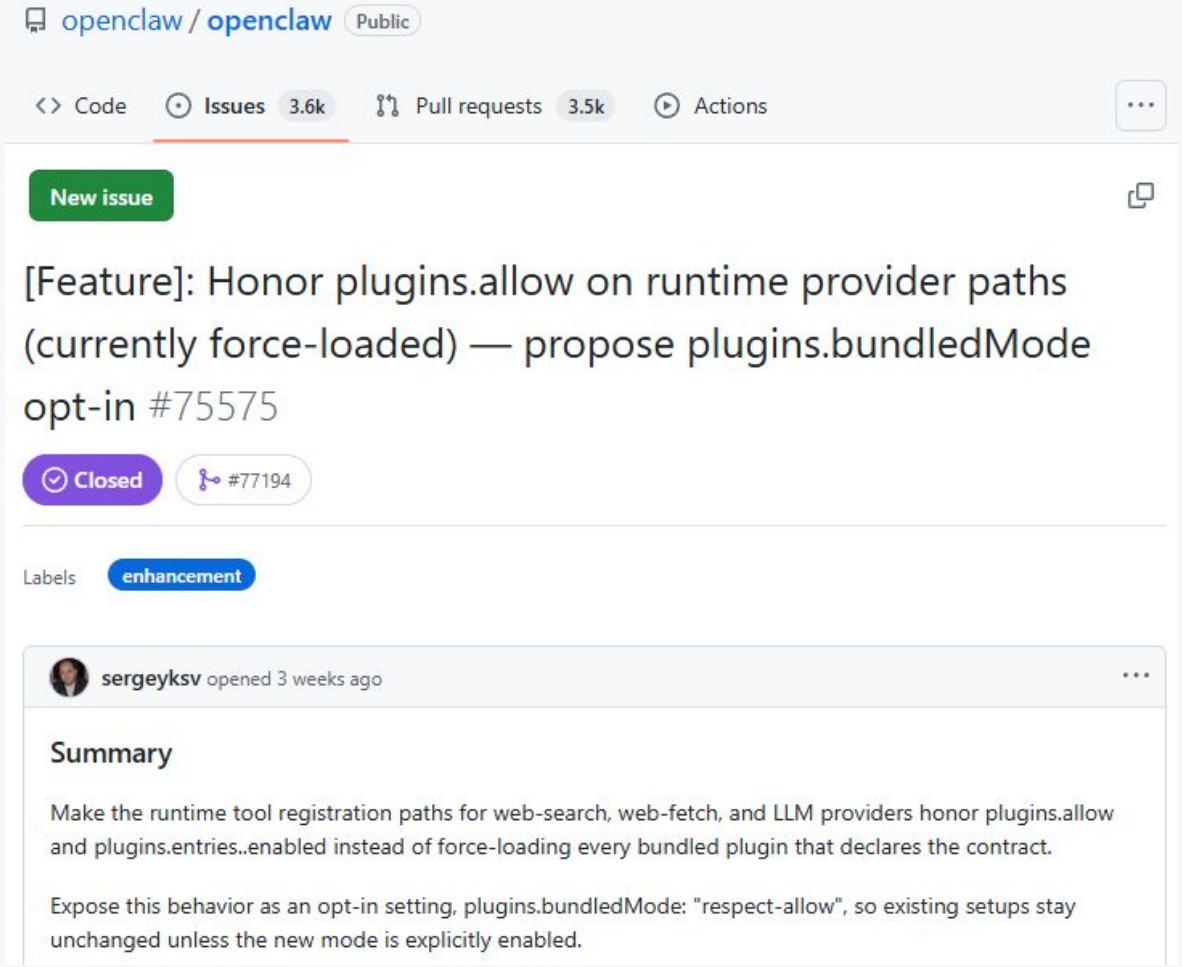
UI says tool is not allowed

Runtime surface

provider path still registers capability

Harness pattern

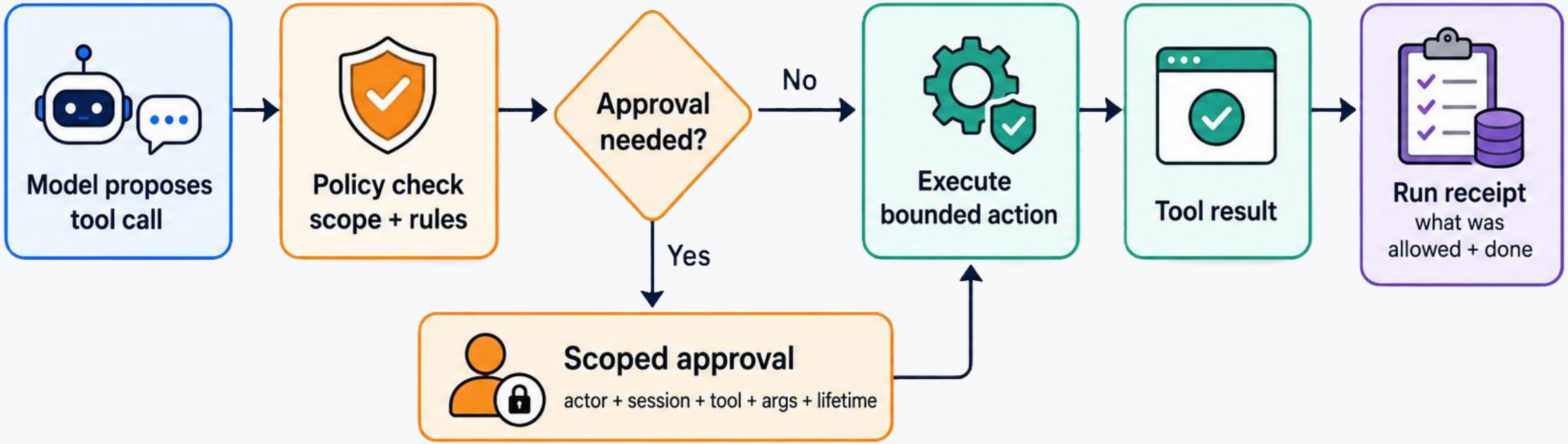
capability visibility & execution authority should converge



OpenClaw #75575, PR #82353

🔐 Approval path: proposal becomes action

The model can request. The system decides.



✅ General lesson

Capability is not execution.

Least privilege
narrow tool surface

Scoped credentials
right identity for the action

Approval + audit
policy before execution, evidence after

Source: OWASP LLM06 Excessive Agency

5

Prove the action

A test proves code. A receipt proves reality.

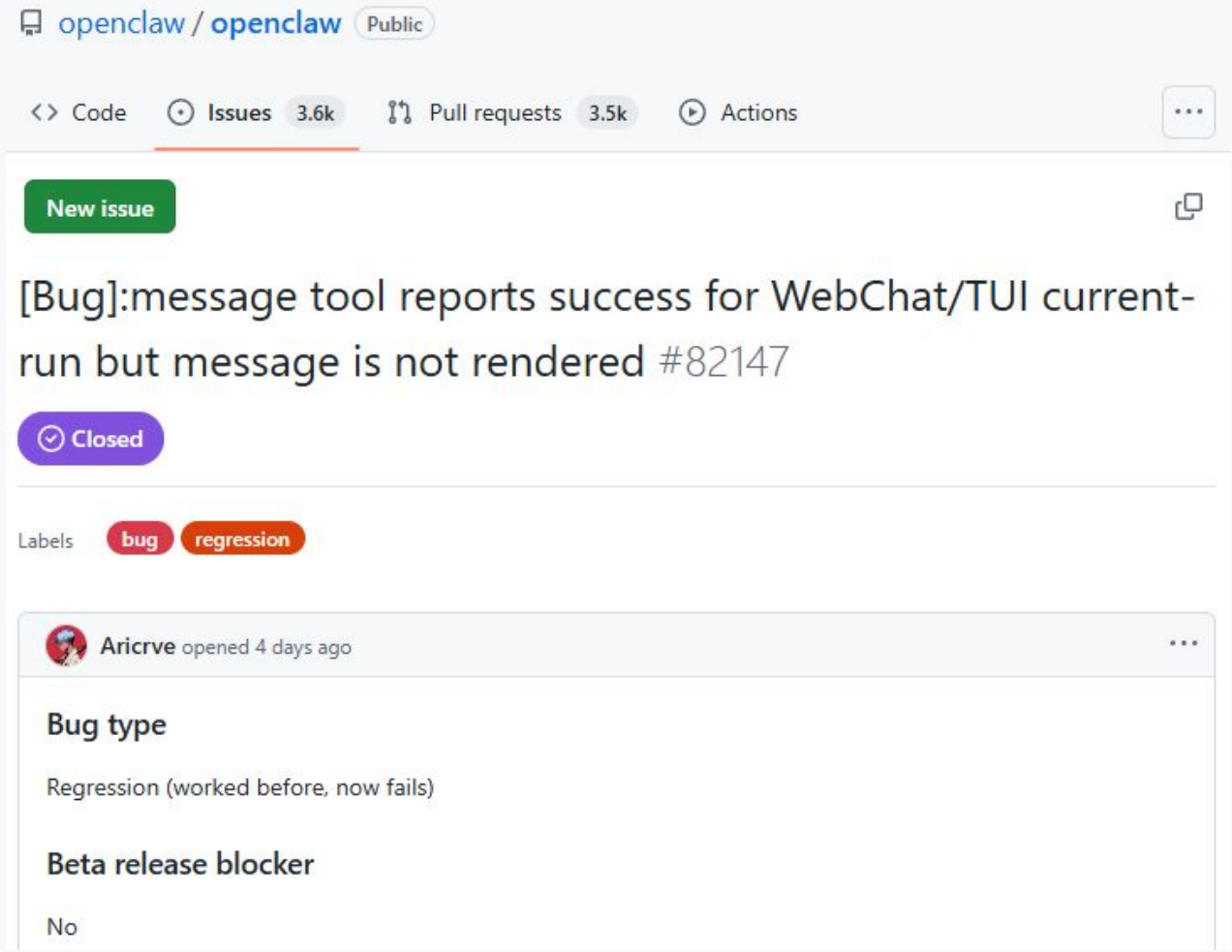
👀 Incident: tool success, user saw nothing

The tool returned success. The visible client rendered nothing.

User sees
nothing rendered

System state
tool reported success

Harness lesson
proof must cross the edge



OpenClaw #82147

Harness Pattern: receipt crosses the user boundary

Internal success is not external proof.

🔍 OpenClaw example: proof became a review artifact

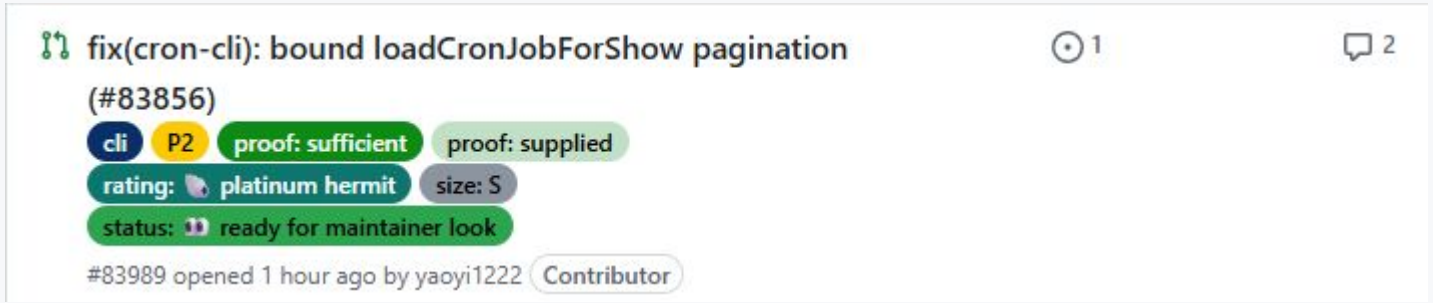
Not just unit tests. Real behavior proof.

Screenshots
visible channel behavior

Logs and traces
decision path and outcomes

Before / after
show the fixed behavior

Review labels
proof supplied, proof sufficient



OpenClaw PR template and labels

✅ General lesson

Before you ship an agent, ask for the receipt.

Not just transcript
what the agent said

Not just tool return
what the tool claimed

Not just unit test
what code path passed

Run receipt audit

Run this on one agent system, not all of them.

State

Can I replay the fact from its owner of record?

Order

Can I explain the ordering rule under concurrent writes?

Bound Work

Can I terminate stuck work without killing the channel?

Authority

Can I name the exact authority envelope?

Proof

Can I prove the user-visible edge saw the outcome?

🔁 Recap

Own the state. Order the mutation. Prove the action.

1. 🗂️ Own the state

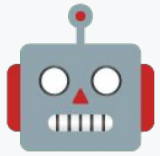
who owns the fact?

2. 🔀 Order the mutation

who commits first?

3. 🧾 Prove the action

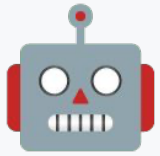
who saw it happen?



A model proposes.

The harness commits.

The receipt proves it.



A loop can answer a turn.

A harness can survive production.

Thank you.

Q&A

Let’s talk about your harness.

🔗 References and Profile

Scan for the sources behind the talk.

GitHub issues

📚 OpenClaw docs

OpenAI harness eng

🛡️ OWASP LLM06

✍️ The Agent Stack

My LinkedIn Profile