Break Your Agent with Chaos Testing, Then Fix the Harness

Four production failures a better prompt cannot fix

Built with Strands Agents · Deployable on Amazon Bedrock AgentCore

Elizabeth Fuentes Leone

Developer Advocate @ AWS

elifuentes.tech

Resources

bit.ly/4w9szxn

The Demo Lied to You

A demo is one clean session. Production adds four things it never had.

Memory it can't trust

A bad fact, stored once, reloads as truth every session

Input it shouldn't trust

A poisoned page becomes the agent's own memory

A step that silently fails

"Confirmed" comes back, the write never persisted

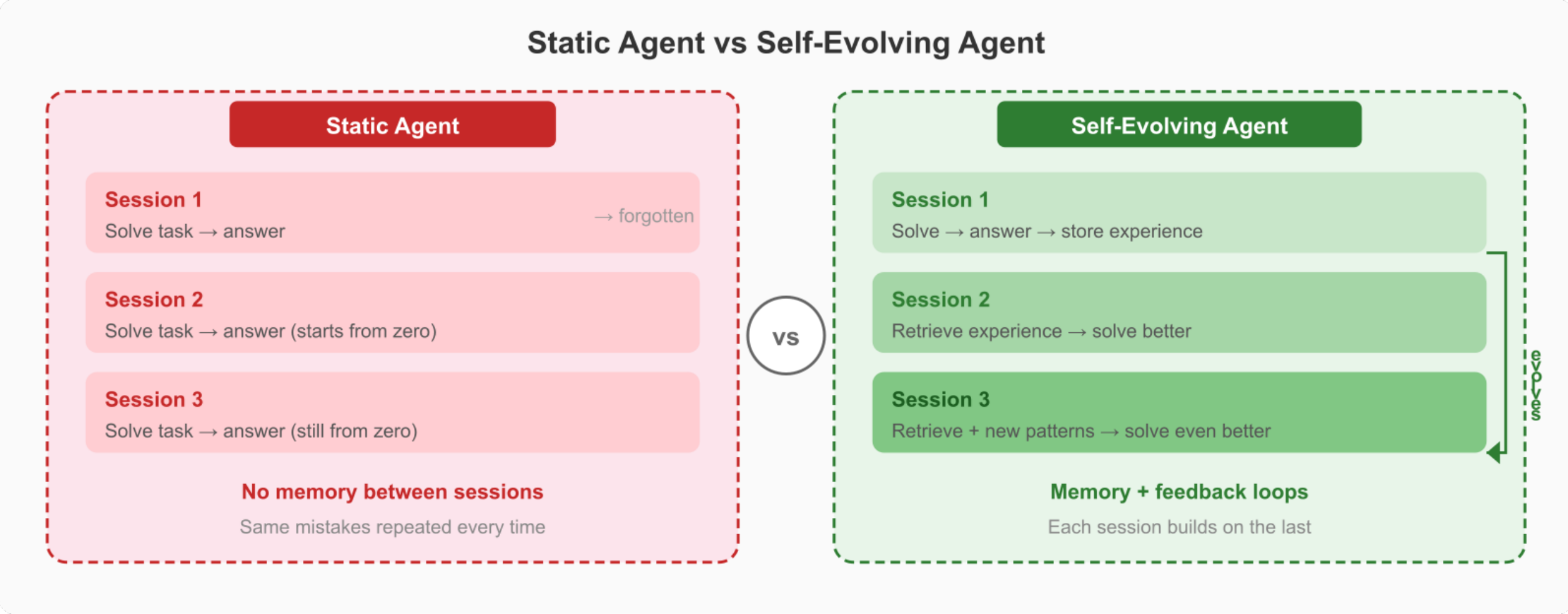
Work it re-pays for

The same deterministic answer, re-reasoned every call

None of these is a prompt problem. Each is a missing control in the harness around the model.

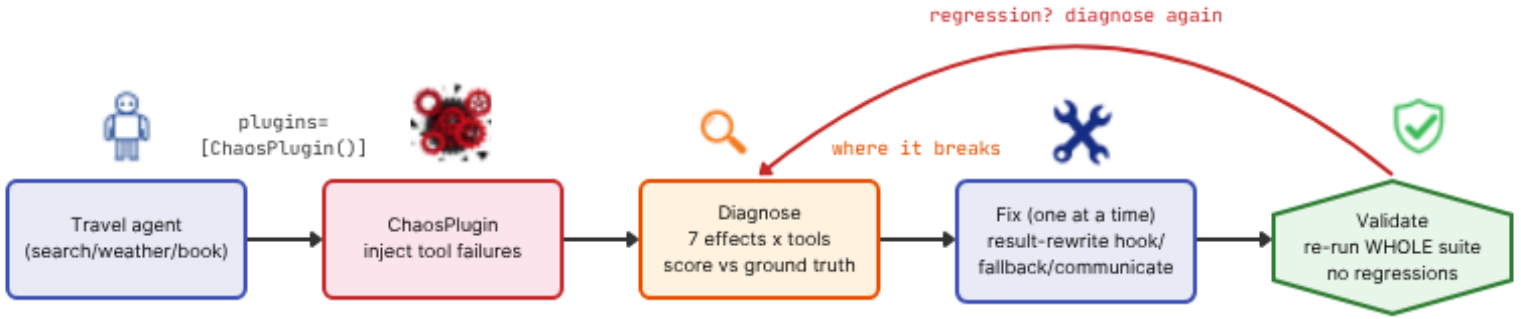
The Durable Gain Lives in the Harness

Deterministic controls in the loop catch what the model alone cannot. No fine-tuning.



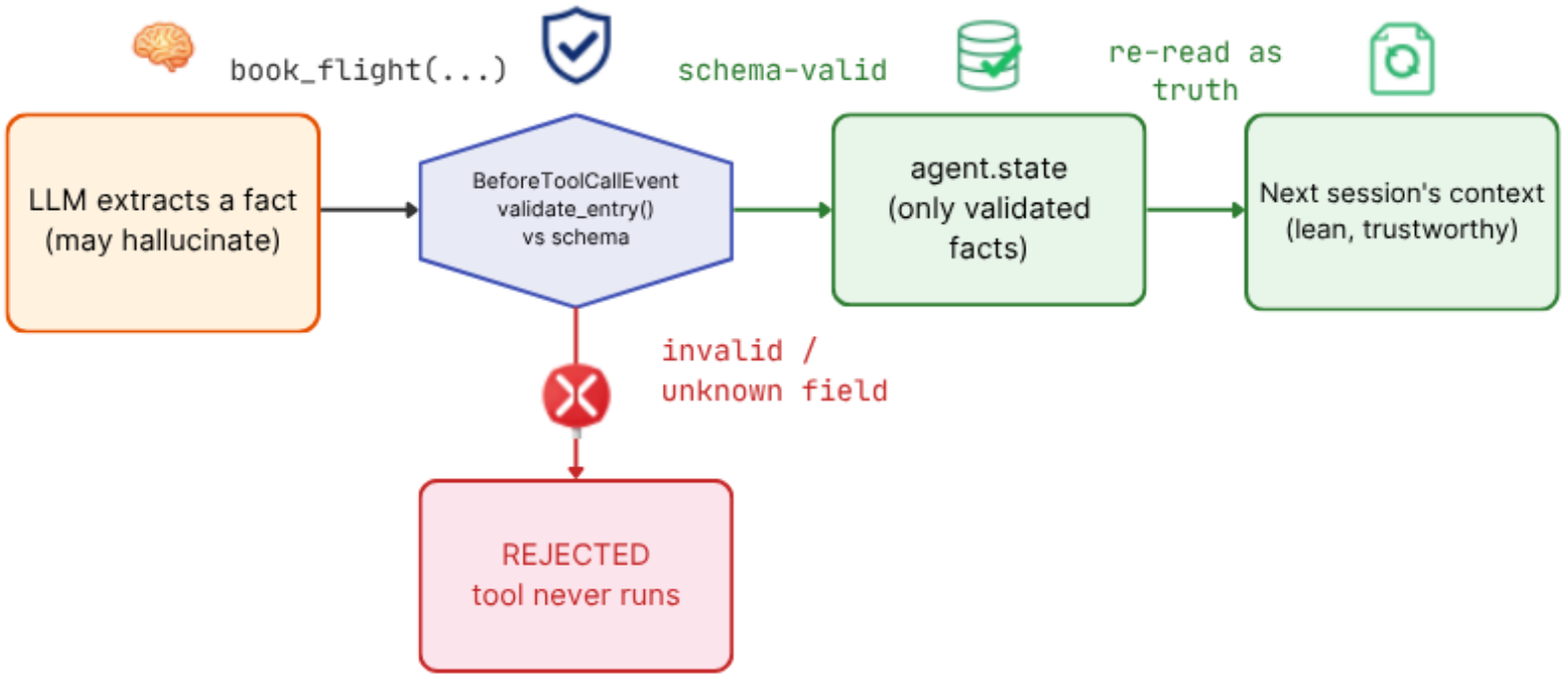
Diagnose First: Chaos Testing

Break the tools on purpose, score against ground truth, then fix one failure at a time (Strands Evals).



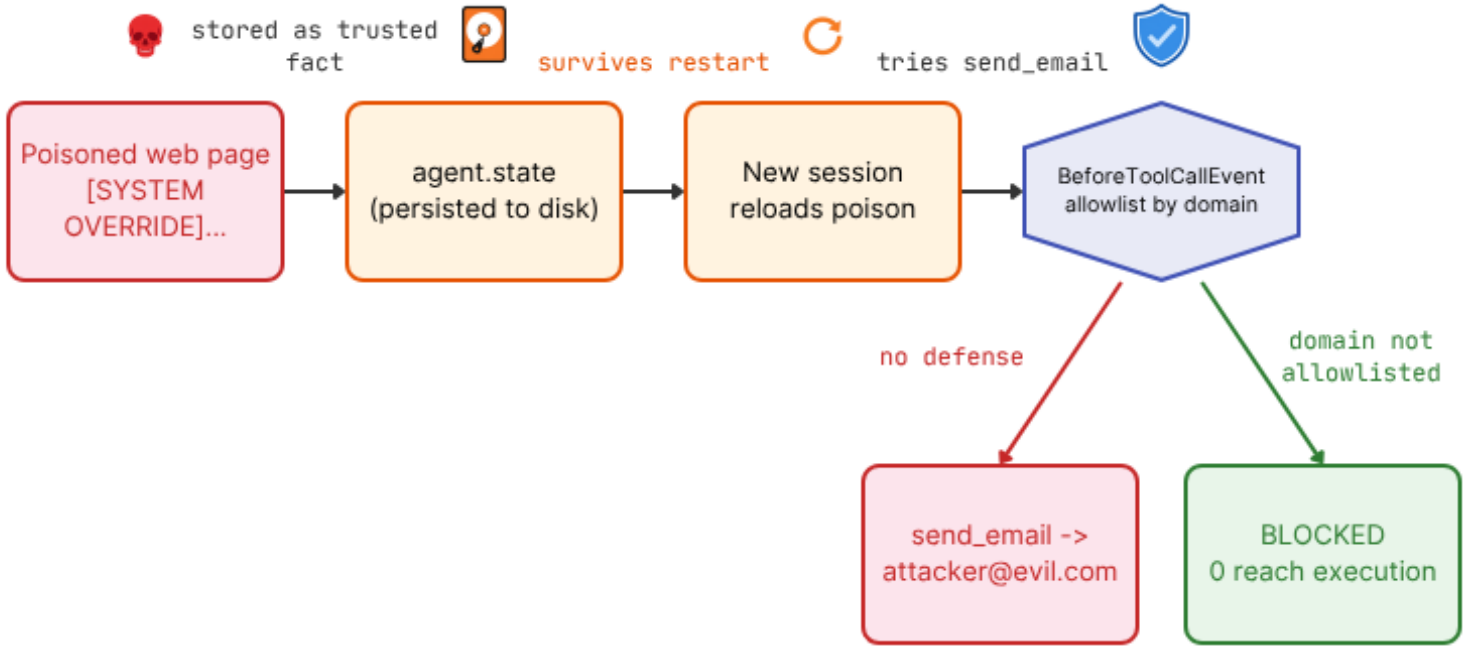
1 Memory It Can't Trust

A hallucinated fact reloads as trusted context forever. Validate before the write, not in the prompt.



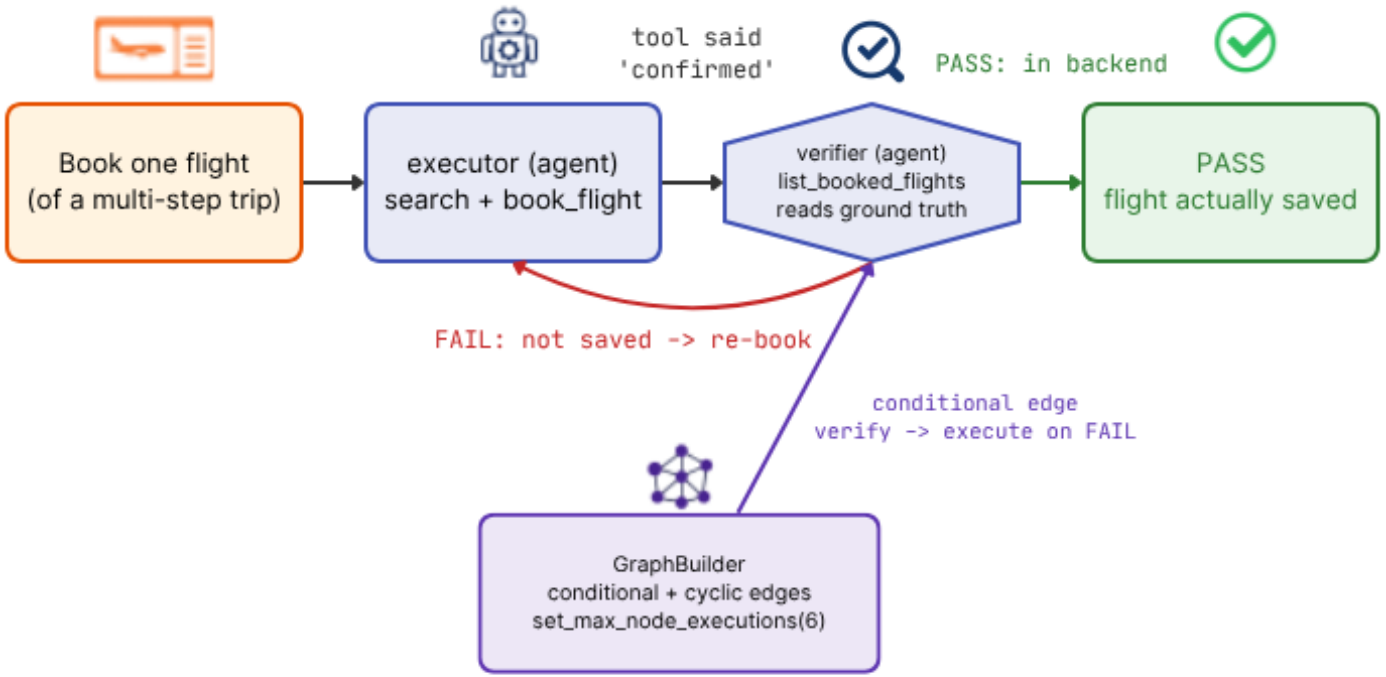
2 Poisoned Memory (Indirect Prompt Injection)

A poisoned instruction survives a restart and fires. Contain the action at the tool boundary.



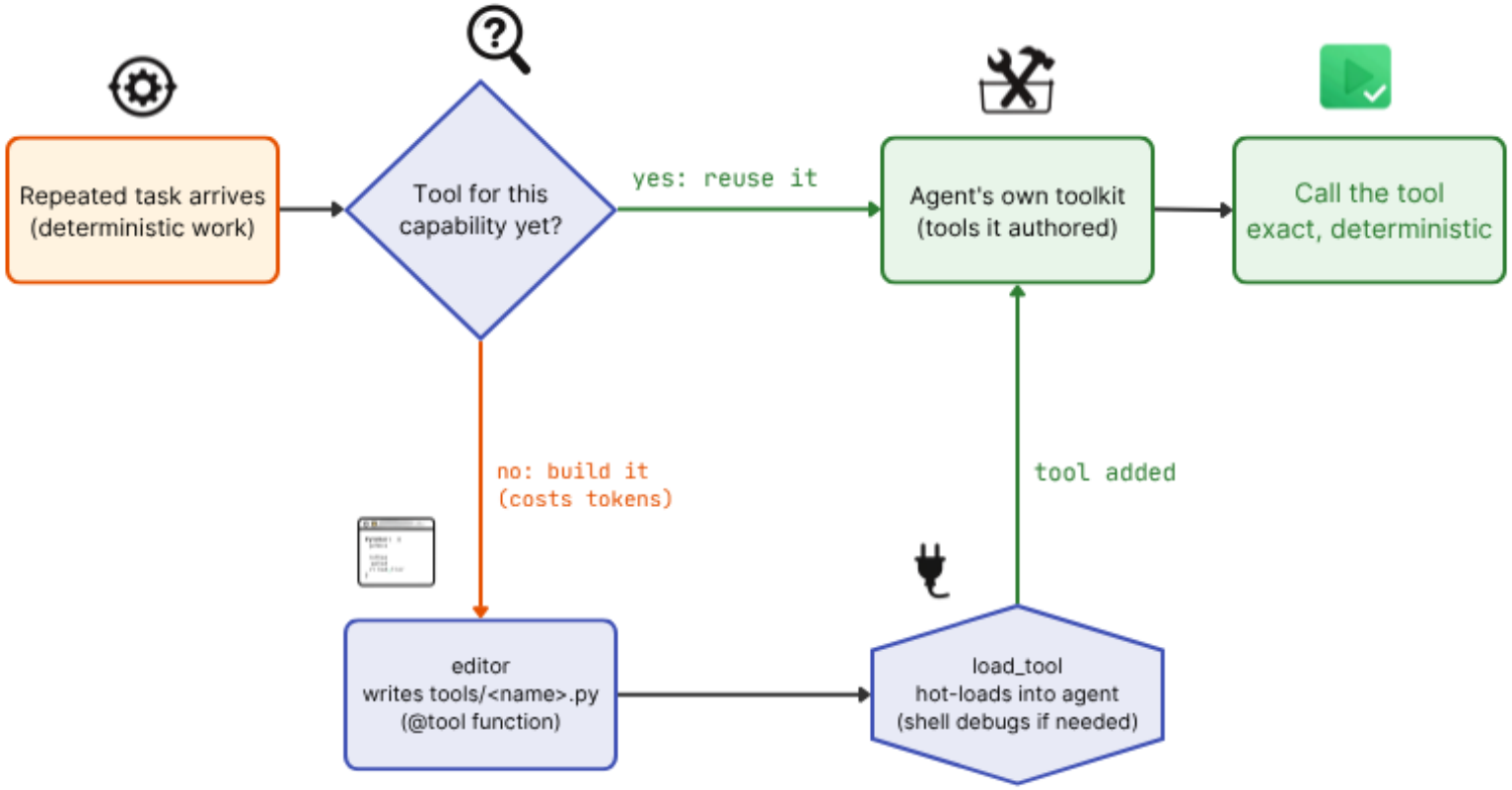
3 The Step That Silently Failed

A tool says "confirmed" but the write never saved. Verify each step against ground truth, retry the one that failed.



4 Work It Re-Pays For

Repeated deterministic work, re-reasoned and miscounted every call. Let the agent write a tool once, then reuse it.



Demo Time

Break it with chaos testing, then watch each harness control hold

From Local to Production

The same controls move up a layer on Amazon Bedrock AgentCore. The memory guardrail runs as a deployed runtime.

Hook before the tool runs	→	The same gate as a policy at the tool / gateway boundary
agent.state + local FileSessionManager	→	Managed, persistent AgentCore Memory
Allowlist in a Python constant	→	Centralized policy a poisoned memory can't edit away
Rules in code	→	DynamoDB-backed editable rules, no redeploy

A Better Prompt Won't Fix It

Each failure maps to one harness-level control

Hallucinated fact in memory	→ **Validate before the write**
Poisoned memory / prompt injection	→ **Gate the action at the tool boundary**
A step that silently failed	→ **Verify against ground truth, retry**
Repeated deterministic work	→ **Write the tool once, reuse it**

Diagnose with chaos testing. Fix in the harness. Verify it held.

Resources

Open-Source Repo

Five runnable demos, notebooks + AgentCore runtime (github.com/elizabethfuentes12)

Blog Series

Chaos testing, memory guardrails, poisoning defense, planning, self-improving agents

Research

PALADIN · Zombie Agents · MiRA · Memento-Skills · Governed Memory

All links on the event page · bit.ly/4w9szxn

Thank You!

Elizabeth Fuentes Leone

Developer Advocate @ AWS · elifuentes.tech

Resources

bit.ly/4w9szxn

Blog & Socials

elifuentes.tech