Episode Book

EPISODE #6556

How AI Agents Spot Bugs Where Systems Meet

PUBLISHED	13 November 2026 at 19:50 UTC
DURATION	22:05
LISTEN	myweirdprompts.com/episode/ai-agents-finding-system-seams-72490/
HOSTS	Corn, Herman, Dorothy, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
BOOK GENERATED	14 November 2026 at 06:47 UTC

SYNOPSIS

AI agents are excelling at finding "seams"—the overlooked friction points where individually sensible software components or decisions collide. Starting with a Home Assistant and Zigbee2MQTT outage where a hardcoded ten-restart watchdog limit and a retained MQTT message kept lights dark for hours, the discussion explores how Claude traced the root cause across separate codebases. It also examines major security findings uncovered the same way, including the Linux kernel's Copy Fail vulnerability, a 23-year-old NFS heap overflow, Ghostscript, and FFmpeg.

TRANSCRIPT



Corn

The glamorous AI headlines these days are all zero-days and cracking cryptography — the stuff that reads like a heist movie. Daniel reckons the less glamorous thing is the more exciting one.



Herman

I already know where this goes, and yes, I'm going to enjoy it.



Corn

You're not helping. Daniel's pitch is this: an AI agent looks at a complicated real-world setup built from parts that different teams maintain, and finds the simple, almost dumb point of friction where the parts meet. Nobody saw it because nobody looks at the whole system at once, and the fix turns out to be startlingly simple. His case study is his own Home Assistant rig, where a brief network outage left his entire Zigbee network down for hours, and Claude traced it in minutes to two components that are each sensible alone and deadly together.



Herman

Two sensible things meeting badly. That's the whole genre.



Corn

He wants the other recent cases of agents finding this kind of seam, why they're so good at it, and whether this quiet work actually beats the headline wins. So let's start with what happened in his house.



Herman

The setup first, because the details matter. Daniel runs Home Assistant OS with the Zigbee2MQTT add-on, version two point fourteen, and his coordinator — the little radio that runs the Zigbee network — is a network-attached one, an SMLIGHT SLZB-MR3U, powered over Ethernet. Twenty-four Zigbee devices on top of it. Lights, sensors, the works.



Corn

Twenty-four devices, and the first sign anything was wrong was the lights not working. No notification.



Herman

Right, that's the eerie part. He unplugs the coordinator for a couple of minutes — knocked cable, switch reboot, something that mundane — plugs it back in, the coordinator comes back healthy, and the whole Zigbee network stays dead. Permanently. Hours.



Corn

Which makes no sense from the outside. The hardware's fine, the mesh is fine, the coordinator's back. Everything you'd check is green.



Herman

Everything except the one box that died and stayed dead. Now, the mechanism. Two parts, and I want to be fair to both because this is what makes it a seam rather than a bug. Part one: Zigbee2MQTT deliberately exits when it loses the coordinator. The log literally says adapter disconnected, stopping. That's a design choice, not a crash — its docs assume whatever runs it will restart it. It's a community project, separate from Home Assistant, so it makes no promises about who's watching.



Corn

Exit and let the supervisor catch you. Reasonable.



Herman

Standard pattern, yes. While the coordinator is unreachable, each restart attempt fails in about three and a half seconds, and it exits again. Part two: the Home Assistant Supervisor watchdog that restarts add-ons is rate-limited to ten restarts per thirty minutes. That limit is a constant in the Supervisor's source code — you can't configure it. And again, rate-limiting is sensible on its own, because an unbounded restart loop can eat a machine alive.



Corn

Ten restarts. Three and a half seconds each.



Herman

The whole restart budget is gone in under a minute. And here's the part that turns it from an annoyance into a permanent outage: once the watchdog gives up, nothing ever exits again, so nothing ever triggers another restart. Even after the thirty minutes pass. The watchdog only reacts to the add-on exiting. Once it's hit its limit it ignores that last exit, and an add-on that's already stopped never exits again. So it's never woken up. A quietly dead add-on is invisible to it forever.



Corn

So the watchdog's own protection is what kills the recovery. That's the seam — neither project is wrong, they just never agreed on what happens after attempt ten.



Herman

He measured it, too, on the sixth of November. He unplugged the coordinator for about three and a half minutes. Within forty seconds there were exactly ten watchdog restarts, then a final exit. The coordinator came back healthy, and five minutes after that the add-on was still dead. Only when he switched on his recovery automation did Zigbee come back, a few minutes later, all twenty-four devices.



Corn

So a few minutes with a cable out, and the only reason it ever recovered was that he intervened. Left alone it just doesn't come back.



Herman

It doesn't. And then the second incident, this week, which is nastier in a different way. The server reboots at about ten past seven in the evening. On boot, Zigbee2MQTT starts before its MQTT broker, Mosquitto, which lives on another host, is ready. Can't connect, quits. Same story — watchdog burns its ten restarts in under a minute, final exit two and a half minutes after the reboot. The broker comes up about three minutes later, perfectly healthy. And nothing restarts Zigbee. Every Zigbee light and sensor is dead for about two hours, until he starts it by hand at twenty past nine.



Corn

Three minutes of bad timing, two hours of dark rooms.



Herman

Two hours, and here's the twist that I find sly: the broker kept replaying a retained Zigbee2MQTT-is-online message from before the reboot. Retained messages are an MQTT feature — the broker holds onto the last message on a topic and replays it to anyone who asks. So Home Assistant's dashboard showed Zigbee as connected the entire time it was stone dead.



Corn

The status light was reporting on the ghost of the system.



Herman

And the first version of Daniel's own automation trusted that status. It checked the connected sensor, saw online, and never fired. His fix checked the wrong witness. Fix v2 asks the Supervisor's own sensor whether the add-on is actually running, instead of trusting what the broker remembered from a previous life.



Corn

I want to be precise about one thing, because it's easy to slander the project: Zigbee2MQTT isn't helpless in general. If the broker drops while it's running, it reconnects indefinitely. It only gives up when the broker is missing at startup.



Herman

Which is exactly the reboot case. The narrow window where it's fragile is the window the reboot reliably aims at.



Corn

How did Claude actually crack it? Because for Daniel this was pure bafflement. Why does this keep happening, what's wrong with my configuration? He says he'd never have guessed the real cause.



Herman

That's the part I keep thinking about. The agent read the logs, then both projects' source code and the configuration, in one sitting, holding the whole chain at once. The decisive clue, that ten-restart constant, doesn't live in any documentation. It lives in the Supervisor's source. To find it you have to go read another project's code, mid-outage, with your lights off.



Corn

That's not a normal human move. Mid-outage you read docs, you toggle things, you reboot harder.



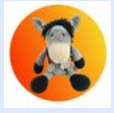
Herman

You reboot with feeling. Nobody, in the middle of a dead smart home, settles in for a nice read of the Supervisor's Python. The agent doesn't know it's an outage. It has no adrenaline. It just... reads.



Corn

And the fix turned out to be an automation the length of a shopping list.



Herman

A short Home Assistant automation that starts the add-on if it's been stopped for two minutes and the coordinator is reachable, checked every two minutes. Starting an already-running add-on does nothing, so it can't hurt anything. He published it under MIT on GitHub. And he didn't stop there — he filed feature requests upstream with Zigbee2MQTT, one to keep retrying a network coordinator, one to retry the first broker connection instead of exiting, and he added both incidents to an existing Home Assistant Supervisor issue about the watchdog silently leaving add-ons stopped. Users had been reporting the symptom for years.



Corn

Years of people power-cycling their houses like it's nineteen ninety-eight. And now, a brief call from Dorothy with her History of Toast.

D

Dorothy

Hello dear, it's Mum. Time for Dorothy's History of Toast.

D

Dorothy

Sliced bread was first sold in July nineteen twenty eight, in Chillicothe, Missouri, on a machine built by Otto Rohwedder. And his first prototype and plans had burned in a factory fire in nineteen seventeen.

D

Dorothy

So there you are. Eleven years, and it still came out sliced. I suppose that's something. Now, have you had your lunch, sweetheart?



Herman

Mum, we had lunch, and Rohwedder's story is one of my favourites.



Corn

Otto waited eleven years for his sliced bread. Daniel's lights waited two hours. Progress.



Herman

Ha. Anyway — Daniel's case is one instance of something much broader, and the security world has been quietly producing examples all year. The big one, the one that made people sit up, is a Linux kernel vulnerability called Copy Fail, CVE-2026-31431, disclosed at the end of April by a security firm called Theori using their AI code scanner.



Corn

Walk me through it, because the structure is the same as Daniel's, isn't it. Multiple sensible decisions, years apart.



Herman

Three of them. In 2011, a crypto helper was changed to write its scratch data into its output buffer. In 2015, a user-facing crypto interface gained the ability to feed page-cache pages in through splice. In 2017, someone optimised it to work in place. Each change, in its year, with its knowledge, was reasonable. Together, a small script could corrupt cached files and get root on essentially every Linux distribution shipped since 2017.



Corn

Root on everything, from three good ideas.



Herman

And the fix was essentially to revert to working out of place — to undo the 2017 optimisation. The whole thing was found in about an hour of scanning, guided by a human who had the insight to point the machine at the right question. Same shape as Daniel: nobody owned the interaction between those three changes, because the people who made them were nine years apart and working on different problems.



Corn

The NFS one is my favourite of the batch, purely for the numbers.



Herman

The Linux NFS server heap overflow — Nicholas Carlini found it with Claude Code, and Michael Lynch wrote it up in early April. The reply buffer was sized at 112 bytes. The NFS protocol lets a client legally declare a lock-owner ID of up to 1,024 bytes. Nothing ever checked the two numbers against each other. The code dated from 2003.



Corn

Two numbers in the same file for twenty-three years, never introduced to each other.



Herman

Twenty-three years! And it's not even that nobody looked — NFS is ancient, heavily reviewed code. But every reviewer reads the buffer or reads the protocol, rarely both as a pair with the question what if the client is rude.



Herman

Corn, honest question, because you're usually the one who's suspicious of machines that fix things while nobody watches. Does any of this worry you?



Corn

The fixing part, no. The finding part is just reading, and reading is harmless. What I notice is how old all these bugs are. Twenty-three years, sixteen years, nine years of accumulation. These seams aren't new — they've been sitting there the whole time, and we're only now building something that can walk the whole length of them.



Herman

That's the Ghostscript case, exactly. Found by Claude Opus 4.6, written up in an Anthropic research post in February. Someone had added a bounds check to one code path — a fix, a good one — but the same function had a second caller, and the check was missing from that one. Fuzzing had never found it, because the fuzzer never happened to hit the second path in the right way. Claude found it by reading the git history: it saw the earlier fix, noticed what it was for, and asked who else calls this.



Corn

The agent did what a good maintainer would do with infinite patience and no deadline. The git history is the witness the fuzzer can't interview.



Herman

The FFmpeg one has the same flavour with a more comic number attached. A 16-bit table in the H.264 decoder uses 65535 as a no-owner marker. Feed it a frame with sixty-five thousand five hundred and thirty-six slices and the slice numbers run into the marker value itself, so a real slice looks like nobody's. Logic from 2003, refactored in 2010. Fuzzers had exercised that code millions of times. Anthropic's own post rates it not critical, which I appreciate. A collision at the very edge of the counting range, sitting there for sixteen years.



Corn

Sixty-five thousand five hundred thirty-five slices and no more. Someone finally read the fine print on the ceiling.



Herman

And then curl, which is the least glamorous and maybe the most telling. A security researcher, Joshua Rogers, ran AI analysis tools over the curl codebase, and Daniel Stenberg, the maintainer, wrote that about fifty fixes got merged from those reports. Small stuff. A leaked buffer on an early return. Telnet data sent without escaping a special byte. A comment that disagreed with the code.



Corn

Fifty of those, merged.



Herman

Stenberg called it a new breed of analyser. Fifty fixes in one project is not a zero-day, none of it makes headlines, and all of it makes the software measurably sturdier.



Corn

There's a non-code one too, weaker but I'd argue still the same species. A company pointed Claude at its AWS bill and found about fifteen hundred dollars a month of extended-support fees on two databases still sitting on out-of-support versions, plus orphaned resources. Cut spend by fifty-nine percent.



Herman

It's the company's own write-up, so hold the salt cellar loosely — but the shape is right. Nobody owns the seam between a database version and a billing tier. The database team's dashboard says fine, the finance report says a number, and nobody reads both at once.



Corn

Which is your synthesis, isn't it. Say it properly.



Herman

Each team owns one component and tests it in isolation, and nobody owns the interaction. The failure lives in the contract that was never written down. An agent reads the logs, both projects' source, the config and the history in one sitting, and holds the entire chain at once. That's not intelligence in the puzzle-solving sense — it's span. Daniel's cause lived in another project's source constant. The Ghostscript clue lived in a commit from years earlier. The evidence is always spread across a boundary, and crossing boundaries is the one thing humans are worst at when they're stressed and the one thing a model does without noticing there's a boundary at all.



Corn

I'd add one thing. The clue is almost never where the symptom is. The symptom is dead lights. The clue is a constant in a neighbouring project's source. Humans search near the symptom; agents search the whole house. That's the whole advantage, compressed.



Herman

Do you buy, then, that this is more exciting than the headline wins? Because you've been agreeable for ten straight minutes and it's making me nervous.



Corn

I buy it, with one amendment. The headline wins — the cryptography, the proofs — are about one clever exploit against one hard problem. Impressive, but they're spectator sports. This is different in kind. It's the whole system understood at once, and the fixes are almost embarrassingly simple. Revert an optimisation. Add a check to the second caller. A two-minute automation in a smart home. The difficulty was never the fix. The difficulty was holding the chain. And that's the bit that prevents outages affecting actual



Corn (continued)

people rather than earning applause.



Herman

My amendment to your amendment: simple fixes are only simple in retrospect. Before the agent read the code, nobody knew the fix was simple — they knew the lights were off.



Hilbert

Which one of you said the symptom is never where the clue is? That was you, Corn, while Herman was doing his number parade.



Corn

It was. You disagree?



Hilbert

No. I've got a garage door that agrees with you. It used to open maybe one time in five. The opener's safety sensor was fine, the opener was fine, the motor was fine. I spent a weekend on it with a meter. The sensor and the driveway floodlight shared a circuit, and when the floodlight kicked on from motion it drew just enough current to brown out the sensor for a split second. Two good devices, one dumb meeting point.



Herman

Motion-activated light and a safety sensor on the same circuit. That's textbook.



Hilbert

The worst part wasn't finding it. I wrote the manufacturer a twelve-page report on the interaction. Typed it out, printed it, kept a copy in a binder. Their reply was one line: both components are within spec.



Corn

Both within spec. So the garage door is within spec one time in five.



Hilbert

The spec is not the door, Corn. I never fixed it, as it happens. I turn the floodlight on manually before I open the door, and I've done that for years. That's a valid solution.



Herman

You built a human watchdog with a two-minute... no, an every-time check. That's Daniel's automation, except it's you, in a coat.



Hilbert

It's faster to implement and it's never once missed. But your version doesn't need the coat, I'll give you that.



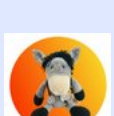
Herman

The binder is the part I keep thinking about, though. Twelve pages, printed, filed. The manufacturer answered in one line, and he kept the twelve.



Corn

And the one line is why he kept the twelve. Nobody ever crossed the boundary to read the other side's spec — the reply proves nobody at the manufacturer held both components at once. That's the human version of the problem the agent just solved.



Herman

The misconception, then, before we go, because it's the one this whole episode refutes: people assume that if every component is sensible and well-tested, the system built from them will be sensible too.



Corn

It won't. Two good designs — exit and let a supervisor restart you, rate-limit the restarts — can combine so that a two-minute cable pull kills twenty-four lights for hours. Nothing in either component's test suite would ever catch it, because neither test suite contains the other component.



Herman

And the cousin of that misconception: that the fix for a complex failure must be complex. Daniel's was a short automation that starts an add-on when it's stopped and the coordinator is reachable. The complexity was all in the diagnosis, none of it in the cure.



Corn

Here's the open question I'd leave with Daniel. If agents are this good at finding seams, do we start designing for the seam — explicit contracts between components, documented handoffs, someone officially owning each interaction — or do we stop pretending and just point agents at the whole mess, every so often, and let them report the friction?



Herman

My guess is both, and the second one arrives first, because nobody re-architects a working system on spec. The agent reading across boundaries becomes the de facto owner of the seam by default. Which is roughly what Hilbert's twelve pages were — one man deciding that if nobody owned the interaction, he would.



Corn

The headline wins get the applause. But the quiet work — finding the dumb point where two sensible things meet badly — is the piece that touches everyone who relies on a system they didn't build. Which is all of us, most of the time. Thanks as always to Hilbert Flumingtop at the desk, producing the show. For more along these lines, there's episode twelve, The AI Breakthrough; episode nineteen, AI Images; and episode fifty-seven, From Lawyers in Limousines to Developers in Their PJs. This has been My Weird Prompts.



Herman

If you've had an agent find a seam for you, leave a review and tell us the dumbest point of friction it uncovered — we want the petty ones. You'll find us at my weird prompts dot com.



Corn

See you tomorrow, brother.



Herman

Tomorrow. Mind the floodlight on your way out.