AI DIGEST

🔥 JFrog: 50+ CVEs generated by an LLM, NVD marked them critical, Red Hat gave one a 10.0 - and all of it is invented

A day where the two most popular Hacker News stories are both about a person who stopped reading what the model produced. Top of the day at 1711 points is the essay "Don't be a meat proxy". The third story at 705 points is JFrog's investigation into fake critical SQLite CVEs generated by an LLM, which NVD and CISA then spread as real. One theme from two sides. In the first case the sender does not read. In the second, the reading is skipped by the same ecosystem that hands out severity scores of 9.8.

Tuesday, 4 August 2026

IN THIS ISSUE

1. 🔥 JFrog: 50+ CVEs generated by an LLM, NVD marked them critical, Red Hat gave one a 10.0 - and all of it is invented 705 points on HN. The most important story of the day. It is about a class of error; SQLite is incidental.

What happened: a new repo appeared on GitHub (`programmervuln/cveadvisory-`) publishing a batch of SQLite vulnerabilities - **part of the 50+ CVEs that JFrog considers LLM slop, all but one**. NVD quickly marked them critical, CISA's ADP agreed. JFrog researchers went to check, and **everything fell apart**:

• **The code the advisories point to does not exist in those versions**, or points at unrelated logic • **The PoC payloads do not work**, they trigger no crash at all • None of these CVEs **appear on the official SQLite advisory page** • Every advisory shows signs of AI generation in Gptzero; merging them into one file trips the AI-content warning

The most concrete example is **CVE-2026-51302 (9.8 CRITICAL)**: the advisory claims a use-after-free where `sqlite3ReleaseTempReg` leaves a dangling pointer that `exprComputeOperands` later dereferences. The problem: **the function `exprComputeOperands` did not exist at all in SQLite 3.41**, it was added in mid-2025. A vulnerability in a function that was not in existence at the time.

The scoring history is telling on its own: **Red Hat first gave CVE-2026-51302 a 10.0 Critical, then dropped it to 7.6 High the next day**. JFrog's methodology is honest and reproducible: they cloned the official repo, built it in isolated Docker containers under AddressSanitizer, and fed the PoC verbatim.

Why it matters. First, **automated patch management driven by a CVE list now has a hole in it**. The first question in the thread was "this is going to be fun for organisations that are required to patch every CVE". The soberest reply there: the only visible defence is **having an agent reproduce the bug BEFORE a human ever sees it**, and that costs money. Second, this is the same mechanism as invented tweet IDs: **a plausible-looking artifact that nobody checked against the primary source**. The text looks valid until you try to use it. The difference is the price of the check: there a curl takes 20 seconds, here it is a compile under ASan. [proven - the methodology is described, the article has a table of six CVEs with the findings]

JFrog investigation · HN thread, 705 points

2. "Don't be a meat proxy" - 1711 points, the most popular story of the day, and it is about the digest genre Essay by Niko Gruhn, 03.08.

The thesis, verbatim: "Too often I ask a question in Slack, or leave feedback on a merge request, or argue with friends in WhatsApp - and I get back: **'Claude said: [giant answer, verbatim]**'. Please don't do this. I've done it myself. But I've also been on the receiving end too many times. **This adds no value. I can talk to Claude myself** - it will be faster, and I keep control of the context. The meat proxy in the middle is not needed."

The sharpest bit follows: "**Reading AI output is extra effort.** It's verbose, often contains plausible-sounding nonsense, and is increasingly dense with jargon." His example of a sentence he got from Claude: "NATS control-plane events: stream leader election / R3 quorum re-form during pod churn". His reaction: "Good lord. I had to look up almost every word."

And the working conclusion: "By all means prompt the AI. But **don't relay the output. Read it, understand it, verify it - and then write the reply in your own words** (which is a decent certificate that the earlier steps happened)."

He also describes the code review version, where this degenerates completely: paste the ticket into Claude Code, never look at the code, paste the reviewer's feedback back, iterate. "It works. But **who did the implementation? The reviewers did, with Claude Code, and the author acted as a meat proxy.**"

Why it matters. The essay criticises **exactly the format a daily digest runs on**: sources read and retold to the reader. There is one defence and it is visible: checking against the primary source and admitting honestly when the check failed. The author's bar is higher: "write it in your own words as a certificate of understanding". By that bar any retelling stays closer to relaying. Wherever a long verbatim quote sits in place of **what follows from it for the reader**, that is a meat proxy. A usable rule: **every quote should either carry a number or be shorter than the conclusion drawn from it.** [proven - a position essay, taken as a criterion, not as a fact about the world]

essay · HN thread, 1711 points

3. Goedecke: "LLMs reward expertise" - with a breakdown of the Terence Tao conversation as evidence 614 points on HN. The most useful **practical** material of the day, and a direct antidote to item 2.

The thesis, against the consensus: "Because of this, many people think there's no skill at all in working with LLMs… Since everyone is talking to the same models, 'skilled prompters' get the same results as people touching an LLM for the first time. **This is wrong. The most important skill in prompting is expertise in the domain you're prompting about.**"

The evidence: the public conversation between **Terence Tao and ChatGPT** about the recently found counterexample to the Jacobian conjecture, dissected by the essay's author. "**This is not the same ChatGPT!** Even with unlimited tokens, getting to where Tao gets would be impossible." The observations he writes out:

• Tao's messages are **very short and to the point**; he does not answer the model point by point, only on substance • the model's outputs are **much more compressed** than when the author talks to GPT-5.6 Sol about mathematics. "**By signalling expertise, Tao flips the model into 'talking to mathematicians' mode. Not 'explaining to amateurs'**"

• Tao **pushes back** when an answer looks wrong, but **does not contradict directly** - he says something like "this seems harder than I was hoping for"

• Tao **makes the leaps and proposals himself**; he **almost never takes the model's advice about where to go next**

And the honest conclusion that saves the essay from being a recipe: "**You can't prompt like Tao just by following this advice.** The key to his technique is actually understanding the mathematics." Then he carries it over to code: "If you have a good theory of the codebase, you can push much harder on the LLM… you can say 'no, this can be simpler', or 'isn't that already being done?'". And finally: "**For many tasks the bottleneck is the human. Not the model**, because the hard part is explaining to the model which solution the human wants. The information is already 'in the model', but getting it out takes a very smart human."

Why it matters. This describes what the Tao move looks like in practice: saying "something here is off" without a diagnosis. The diagnosis may be wrong, but pointing precisely at **what looks wrong** narrows the search down to the real cause. The phrase "this seems harder than I was hoping for" works the same way, in place of flat contradiction. **Expertise in what the result should look like is what pulls decent work out of a model.** The flip side: where the requester does not know the domain there is no signal, so the person doing the work has to hold the bar themselves. [proven - a breakdown of a public conversation, the author marks the limits of his own conclusion]

Goedecke's essay · HN thread, 614 points

4. GPT-Live: OpenAI rewrote the whole voice stack - the model listens while it speaks The main release of the day. OpenAI's post got **495,190 views**, Brockman's thread another 73K.

Verbatim from OpenAI: "**GPT-Live can listen while it speaks.** To make this feel natural at ChatGPT scale, the voice stack was rebuilt from client to model. This new architecture keeps audio in a continuous stream, so deeper reasoning and tool use **do not interrupt the conversation**."

The technical core: "**Audio travels on a dedicated fast path, while deeper reasoning and tool calls happen asynchronously.**" And the number that matters: "Starting a voice session went **from six network round trips down to one**."

Why it matters. A typical voice loop in an assistant runs **sequentially**: transcription → text → reasoning → reply. OpenAI's architectural idea is to **split the fast path from the slow one**, so audio does not wait on reasoning. Same thesis as with harnesses: **latency is the currency**, because the feeling of a live conversation runs straight into it. [proven - on the claimed architecture; **fuzzy** - on the actual gain, no measurements]

OpenAI: announcement · fast path and 6→1 round trip · @gdb

5. Crawshaw (ex-Tailscale): "Devtools must be open source" - because the agent now keeps your fork in sync 540 points on HN. The author is **David Crawshaw**, founder of Tailscale, now building exe.dev. The strongest idea of the week about how an agent changes the economics of customisation.

His observation from the past: five years ago most engineers he talked to **had no program written for themselves at all**. The reason was ROI: "coming back to a project a year later to do maintenance was always extraordinarily painful".

What changed - two prompts he writes out literally:

1. "Download the sources of `<software>` and build for local use. Edit `<agent memory>` to know that any future changes to this software mean changing the sources and replacing the current version. Record in version control **the original motivation for the change**."
2. And, in his words, **the more important one**: "Set up a nightly cron job that runs the prompt: **pull changes from upstream and rebase all local changes on top**. Check that the software works as intended, and replace the current version."

The point: "agents can also **automatically manage the process of syncing with upstream releases**". So the ROI of customisation shifts **from both sides at once**: easier to start and easier to keep going.

The soberest objection in the thread came from **@simonw**: the freedom to modify open source always came down in practice to "the ability to lean on other people who will do it", because most could not justify the time. The second, sharper one: an agent and an LLM can introduce subtle bugs into **a codebase the user does not know, with exactly one user**, so there is nobody to catch them.

Why it matters. The first half of the recipe is already standard: a set of personal scripts plus a nightly scheduler exists for anyone who automates their own work. The second half, keeping it in sync with upstream, is implemented almost nowhere. And simonw's objection lands: **a personal tool has one user, and there is nobody to catch subtle bugs**. The typical failure here is silent: nothing crashes, the numbers are just stale. So the practical task from the essay is **making a silent failure loud**: check that the data is fresh and that formulas reach the end of the range. [promising - an argued position from a practitioner with Tailscale's weight, with real objections in the thread]

Crawshaw's essay · HN thread

6. Cloudflare published the numbers for squeezing Kimi and GLM into memory without losing accuracy 172 points on HN. A rare case of an infrastructure company publishing **before and after tables** instead of the word "optimised".

Three techniques and what each costs, straight from the measurements:

FP8 instead of BF16 for the KV cache. On Kimi K2.6 this raises the context that fits in memory **from ~686,000 to ~1.37M tokens**. But the honest remark is the main part: "it's worth being precise about where the benefit comes from, because it's **not raw speed**". The table on an H200:

CONCURRENT REQUESTS	BF16 (TOK/S)	FP8 (TOK/S)
1	**137**	125
32	**1,558**	1,489
64	**out of memory**	**2,192**

At each level taken alone BF16 is a few percent faster, **but it dies at the 33rd request**. FP8 gets to 64 and peaks **41% above BF16's peak, at ~30% lower cost per token**.

INT4 for GLM 5.2 weights: the checkpoint goes **from 705 GB to 421 GB** (−40%), GPU memory at 8-way tensor parallel **from ~88 GB to ~52 GB**, which leaves room for **~1.18M tokens** of KV cache on the same hardware. Accuracy was checked and published: GSM8K **94.24 → 94.09** on the FP8 cache, MMLU **89.11 → 89.04**, tool call validity **92.2% → 92.6%**.

Why it matters. This is a counterexample to yesterday's item 2, the Wafer article where the author picked the flattering denominator himself. There is a ratio metric here too, "~30%

lower cost per token". But next to it sits the row where the decision loses, 137 against 125 at one request. And the accuracy benchmarks are laid out, so you can check that quality was not traded for memory. That is what an honest presentation of the same construction looks like. [proven - the company's own measurements with tables; it is their hardware and their benchmark, no independent reproduction]

Cloudflare blog · HN thread

7. Exa: 80B pages, 1.4T URLs - and the claim that agents need an index bigger than Google's 371,940 views, 1,681 likes. The most ambitious claim of the day, and a well-counted one.

Verbatim from @WilliamBryk (founder of Exa): "Exa is now **one of the largest indexes in the world**. Serving **80B pages, tracking 1.4T URLs**, and heading to Google scale **in early 2027**." His estimates of the competition: Google ~1T, Bing ~500B, Yandex ~200B, Brave said 40B back in April. Two technical details:

• "**Most of the web is garbage that can poison AI outputs**, so you have to crawl far more than you serve and train models to filter the garbage out"

• On load: peak QPS is estimated at **~30K/s for Bing and ~500K/s for Google**. And why agents are different: "agentic traffic often needs **large fan-out** (deep searches can use dozens of sub-searches) and is bursty (for example when AI labs run RL on us)"

The forecast: "Within 2 years agents will be searching **millions of times per second**... so you need search infrastructure **bigger than Google scale on both dimensions**."

Why it matters. The interesting part is **fan-out**: one agent request equals dozens of sub-requests. Assembling this issue is the same traffic in miniature: two passes over X (98 and 75 posts), 43 HN stories, nine primary sources opened individually. When the cost of agent work is being counted, **the whole fan-out** has to be counted, and it is the same currency of time as in item 4. [proven - on his claims; **fuzzy** - on his estimates of other indexes, those are his guesses and he says so himself]

@WilliamBryk

8. Retyping by hand as a cure for "cognitive debt" - 416 points Essay by Ankur Sethi, 02.08, and the most counterintuitive practice of the week.

The problem he describes: "letting an assistant roam freely across projects leaves you with **enormous cognitive debt**. You can hate the idea of wading through Django documentation... but you still **fundamentally want to understand how it works**. A task being boring does not mean you want to hand the machine your understanding of the solution."

Why not review: "I don't like reviewing AI-generated PRs. Wading through hundreds of lines of **overly defensive, poorly commented, subtly wrong** code is not fun."

His solution, which he himself calls "grossly inefficient and somewhat comical": **he asks the assistant to show the code in chat and types every edit in by hand**. The instruction he puts in the agents file of all his personal projects, verbatim: "I want to understand every line of code that goes into this project. **Never create, edit, move, rename or delete project files**

unless explicitly asked. Instead, show every proposed edit in chat so I can type it in manually." Plus the same for commands, plus "I am an experienced developer. Don't explain syntax, APIs or concepts."

Why it matters. The practice is not universal and the author admits it. He does this **on personal projects where the process is the value**, and says plainly that for an employer he would do it "through gritted teeth". But this is the third author of the day (after items 2 and 3) arriving at the same place: **understanding does not get delegated along with the work**. A softer version of the same: require the agent to leave a trail in chat of why it did what it did, on top of a bare "done". [fuzzy - one person's personal practice; the frame is worth taking, the recipe is not]

essay · HN thread

9. The productivity gap: why AI speeds up code 3x and the team 15% 113 points on HN. An article by Bjorn Roche from 12.07 that resurfaced now, and the most useful arithmetic for a conversation with the business.

His frame is simple: executives expect finished features to come out at prototype speed. But "**writing new code is not where most of the time goes**" for a developer. His breakdown of a senior's day at a large company, **assuming AI speeds up coding itself by 3x**:

	BEFORE AI	AFTER AI
Writing code	1.5 h	**0.5**
Reading and debugging	1.5	1.0
Design and architecture	1.0	1.0
Code review	0.75	0.75
Documentation and admin	0.75	0.75
Tests, CI/CD, deploy	0.5	**0.75**
Mentoring	0.5	0.5
Meetings	1.5	1.5
Total	**8.0 h**	**6.75 h**

The saving is **1.25 hours a day, around 15%**. Worth noting the row that **grew**: tests and deploy, because there is more code.

A separate remark of his, very much on theme for this issue: "sometimes AI makes non-code work **slower**. For instance, when you have to read a product requirement or a Linear ticket **written by AI, it takes longer** than reviewing a document written by a human. AI writing tends to be over-detailed, which makes it harder to extract the main point."

Why it matters. This is the fourth independent voice of the day hitting the same spot (items 2, 3, 8), and the only one giving **a number**. In an argument about how much AI will speed up a team, there is one argument: **what share of the day is taken by the thing it actually**

speeds up. And the line about AI-written tickets is a ready-made argument against the temptation to generate documentation by volume. [promising - a model with the author's assumptions, not a measurement; but the assumptions are written out explicitly and can be contested line by line]

article · HN thread

10. QM: 7,616 → 10,059 stars. Past ten thousand on day six of its life Fourth day of watching this curve, and it still has not broken.

GitHub API right now: **10,059 stars, 1,063 forks**, repo created **29.07.2026**. The timeline: **01.08 - 2,475 → 02.08 - 5,234 → 03.08 - 7,616 → 04.08 - 10,059**. Growth in a day: **+2,443** after +2,382 and +2,759. The pace **has not faded for four days running**, which makes it a steady inflow. The official repo description: "**Multiplayer agent harness for work**". [proven - numbers from the GitHub API, checked directly]

QM repo

📎 **Misc**

• **@paulg on selling to big companies** - 510,825 views, the most popular post of the day in the "Tech + Product" list: "The danger of selling to big companies, if you're a startup, is that **they don't say 'no' directly**. First they'll have months of meetings. Since startups hate meetings, this seems like a sign of interest. But it isn't. **They love meetings! It's practically all they do.**" post

• **@patrickc ran a poll on the economics of AI** - 255K views, 709 bookmarks. A continuation of yesterday's item 9 (his own essay on aesthetics), except now he is collecting the public's five-year forecasts post · the poll itself

• **@patrickc showed a deploy in three prompts across two posts**: "After local development (two prompts), the deploy took one prompt: *'Push this to Vercel. Use Stripe Projects to create an account. Store the state somewhere safe.'* **Claude picked Upstash on its own** for storage, and it seems to work well" - 104K views post

• **@awilkinson wrote to Altman about the Codex UX**, 153K views: "Sam, Codex is great. But there's one really dumb/subtle problem. **Claude Code feels faster.** The reason seems mostly to be that with each action it **immediately dumps a text update into the terminal**, while Codex keeps everything collapsed and hidden." Same theme as GPT-Live in item 4: **perceived latency matters more than real latency** post

• **@emollick on Codex beyond code**: he asked for a game ("you play an otter climbing into animal-shaped mech suits") and the agent **drove Blender and Unity itself, producing animated models** - 14.7K views post. Separately, something very mundane: **Codex fixes his Windows machines** - drivers, game incompatibilities. Example: Steam games were hanging because a wireless keyboard update **installed itself as an Xbox controller** and conflicted with the old joystick API. "Codex fixed it. Hard to imagine how I would have worked that out on my own" post

• **@lennysan and Whatnot's CPO, a continuation of yesterday**: "**Often it's optimal to have no product manager at all**" post. And his main takeaway from the interview: "The biggest AI unlock for PMs right now is **in data science**, and it beats prototyping: pulling cohort retention, log analysis, regression models - work that at Amazon took an L7 data scientist" post

• **@alvinsng: Factory has no PMs at all** - engineers talk to customers themselves, watch engagement, write plans, update stakeholders. Third independent signal in the same direction in two days post

• **arXiv: agentic COBOL → Java migration with a deterministic oracle.** The "Locksmith Loop" method: both environments (the COBOL source and the generated Java) are instrumented with mocks and run off the mainframe. Across three cases (430 - 4,114 lines) they reached **91.90% branch coverage** on an internal production-like program and near-complete coverage on two open source ones paper · HN

• **Fortune: the hidden debt of AI hyperscalers is $1.65T** after growing almost 1000%. S&P Global counts **$225B of bonds** issued by hyperscalers and related entities (including Nvidia) this year, and that is **only the visible part**. Article from 31.07, surfaced on HN yesterday Fortune · HN

• **Investigation: OpenAI's super PAC funds an AI news site attacking critics of the industry** - 205 points on HN. The site Acutus, with no scale and no bylines, published **94 articles in under four months**; "reporter Michael Chen", who was asking for comment, turned out to be a bot (Pangram: "fully AI-generated"). The trail leads to Targeted Victory, the firm at the centre of OpenAI's **$125M political operation**. **The article is from 24.04**, so not an event of the day, but it lands squarely on the theme of item 1 Model Republic · HN

• **@amasad: Replit built a self-driving semantic layer** over its databases, conversations and docs. "Everything queryable and joinable regardless of source", so anyone can ask questions that used to take **weeks of work from a data science team** post

• **@levelsio on his vibe-coded calorie tracker** - 171K views. He has held a ~500 kcal deficit and a protein target (150 g at first, later raised to 180 g) for about a month, losing **~500 g a week**. And separately on skyr: 500 g gives **60 g of protein for 320 kcal** post · on skyr

• **@naval - yesterday's "Agent Programming Interface" has grown to 458,583 views** and 5,976 likes (yesterday it was 85.2K an hour after publication). An explanation still has not appeared post

• **@mervenoyann on `llama-macos`**: a widget that looks at the machine and the context window you want, then **recommends models**. It spins up llama server plus webui and makes switching between open models easy. `brew install --cask llama-app` post