AI-ДАЙДЖЕСТ

LLM вигадали CVE, NVD повірила

JFrog знайшла 50+ фейкових CVE для SQLite, Red Hat спершу оцінила ризик у 10.0

вівторок, 4 серпня 2026

У ЦЬОМУ ВИПУСКУ

1. 🔥 JFrog: 50+ CVE згенеровані LLM, NVD дала їм «критично», Red Hat поставила 10.0 - і все це вигадка 705 балів HN. Найважливіша історія дня. Вона про клас помилки, SQLite тут випадковий.

Що сталось: на GitHub з'явилось нове репо (`programmervuln/cveadvisory-`), яке опублікувало пачку вразливостей SQLite - **частина з 50+ CVE, які JFrog вважає LLM-слопом, окрім однієї**. NVD швидко позначила їх критичними, ADP від CISA погодилась. Дослідники JFrog полізли перевіряти, і **все розсипалось**:

• **Код, на який посилаються адвайзорі, не існує в тих версіях** або посилається на непричетну логіку • **PoC-payload не працюють**, не викликають жодного краху • Жодної з цих CVE **нема на офіційній сторінці адвайзорі SQLite** • Усі адвайзорі показують ознаки ШІ-генерації в Gptzero; злиття їх в один файл вмикає попередження про ШІ-контент

Найконкретніший приклад - **CVE-2026-51302 (9.8 CRITICAL)**: адвайзорі стверджує use-after-free, де `sqlite3ReleaseTempReg` лишає висячий вказівник, який потім розіменовує `exprComputeOperands`. Проблема: **функції `exprComputeOperands` у SQLite 3.41 не**

існувало взагалі, її додали в середині 2025 року. Вразливість у функції, якої на той момент не було в природі.

Окремо показова динаміка: **Red Hat спершу присвоїла CVE-2026-51302 оцінку 10.0 Critical, а наступного дня знизила до 7.6 High**. Методологія JFrog при цьому чесна і відтворювана: клонували офіційне репо, зібрали в ізольованих Docker-контейнерах під AddressSanitizer, згодували PoC дослівно.

Чому це важливо. Перше: **автоматичний патч-менеджмент за списком CVE тепер має дірку**. У треді перше ж питання було «а весело ж буде організаціям, які зобов'язані патчити всі CVE». Найтверезіша репліка звідти: єдиний видимий захист - **щоб агент відтворив баг ДО того, як його побачить людина**, і це коштує грошей. Друге: це той самий механізм, що й вигадані ID твітів, - **правдоподібний артефакт, який ніхто не звірив з першоджерелом**. Текст виглядає валідним, поки не спробуєш ним скористатись. Різниця в ціні перевірки: там курл за 20 секунд, тут компіляція під ASan. [proven - методологія описана, таблиця з шести CVE і знахідками є в статті]

розслідування JFrog · HN-тред, 705 балів

2. «Не будь м'ясним проксі» - 1711 балів, найпопулярніша історія доби, і вона про жанр дайджестів Есей Ніко Груна від 03.08.

Теза дослівно: «Надто часто я ставлю питання в Slack, або лишаю фідбек під мердж-реквестом, або сперечаюсь з друзями в WhatsApp - і отримую у відповідь: **"Claude сказав: [гігантська відповідь дослівно]"**. Будь ласка, не робіть так. Я й сам так робив. Але я був і на приймальному боці забагато разів. **Це не додає цінності. Я можу поговорити з Claude сам** - це буде швидше, і контроль над контекстом лишається за мною. М'ясний проксі посередині не потрібен.»

Далі найточніше: «**Читати вивід ШІ - це додаткове зусилля.** Він багатослівний, часто містить надто правдоподібну дурню і дедалі щільніше набитий жаргоном.» Приклад речення, отриманого від Claude: «NATS control-plane events: stream leader election / R3 quorum re-form during pod churn». Реакція автора: «Господи. Довелось шукати майже кожне слово».

І робочий висновок: «Обов'язково промптіть ШІ. Але **не переказуйте вивід. Прочитайте його, зрозумійте, звірте - і напишіть відповідь своїми словами** (це пристойний сертифікат того, що попередні кроки зроблені).»

Окремо описана механіка код-рев'ю, де це вироджується повністю: копіюєш тікет у Claude Code, не дивишся в код, копіюєш фідбек рев'юера назад, ітеруєш. «Це працює. Але **хто зробив імплементацію? Рев'юери - за допомогою Claude Code, а автор виступив м'ясним проксі.**»

Чому це важливо. Есей критикує **рівно той формат, у якому влаштований щоденний дайджест**: джерела прочитані і переказані читачеві. Захист один, і його видно: звірка з першоджерелом і чесне визнання, коли звірити не вдалось. Критерій автора жорсткіший: «напиши своїми словами як сертифікат розуміння». За ним будь-який

переказ лишається ближчим до релею. Там, де стоїть довга дослівна цитата замість того, **що з неї випливає для читача**, це м'ясний проксі. Робочий критерій: **кожна цитата має або нести цифру, або бути коротшою за висновок з неї.** [proven - есей-позиція, взята як критерій, не як факт про світ]

есей · HN-тред, 1711 балів

3. Ґьодеке: «LLM винагороджують експертизу» - і розбір переписки Теренса Тао як доказ 614 балів HN. Найкорисніший **практичний** матеріал доби і пряма протиотрута до пункту 2.

Теза проти загальноприйнятого: «Через це багато хто вважає, що в роботі з LLM немає жодної навички... Оскільки всі говорять з тими самими моделями, "вправні промптери" отримують ті самі результати, що й люди, які торкнулись LLM уперше. **Це неправильно. Найважливіша навичка в промптингу - експертиза в домені, для якого промптять.**»

Доказ: публічна розмова **Теренса Тао з ChatGPT** про нещодавно знайдений контрприклад до гіпотези Якобіана, препарована автором есею. «**Це не той самий ChatGPT!** Навіть маючи необмежені токени, дійти туди, куди доходить Тао, було б неможливо.» Спостереження, які він виписує:

• повідомлення Тао **дуже короткі й по суті**; він не відповідає моделі пункт за пунктом, лише на суть • виводи моделі **значно стисліші**, ніж коли автор говорить з GPT-5.6 Sol про математику. «**Сигналізуючи експертизу, Тао перемикає модель у режим "розмови з математиками". Не "пояснення аматорам"**»

• Тао **штовхає назад**, коли відповідь виглядає хибною, але **не суперечить прямо** - каже щось на кшталт «це виглядає складніше, ніж я сподівався»

• Тао **сам робить стрибки і пропозиції**; він **майже ніколи не приймає пораду моделі про те, куди йти далі**

І чесний висновок, який рятує есей від рецептурності: «**Промптити як Тао, просто виконуючи ці поради, не вийде.** Ключ до його техніки - насправді розуміти математику.» Далі перенесено на код: «Якщо є хороша теорія кодової бази, на LLM можна тиснути значно сильніше... можна сказати "ні, тут можна простіше", або "а хіба це вже не робиться?"». І фінальне: «**Для багатьох задач вузьке місце - людина. Не модель**, бо складна частина - пояснити моделі, якого саме рішення хоче людина. Інформація вже "в моделі", але щоб її витягти, потрібна дуже розумна людина.»

Чому це важливо. Тут описано, як виглядає хід Тао на практиці: сказати «щось тут криво» без діагнозу. Діагноз може бути хибним, але точна вказівка на те, **що виглядає не так**, звужує пошук до справжньої причини. Так само працює фраза «це виглядає складніше, ніж я сподівався» замість прямої суперечки. **Експертиза в тому, як має виглядати результат, - це і є те, що витягує з моделі нормальну роботу.** Зворотний бік: там, де замовник домену не знає, сигналу нема, тому планку має тримати сам

виконавець. [proven - розбір публічної переписки, автор чесно позначає межі свого висновку]

есей Ґьодеке · HN-тред, 614 балів

4. GPT-Live: OpenAI переписала весь голосовий стек - модель слухає, поки говорить Головний реліз доби. Пост OpenAI - **495 190 переглядів**, гілка Брокмана окремо 73K.

Дослівно з OpenAI: «**GPT-Live може слухати, поки говорить.** Щоб це відчувалось природно в масштабі ChatGPT, голосовий стек перебудовано від клієнта до моделі. Ця нова архітектура тримає аудіо в безперервному потоці, тож глибші міркування і використання інструментів **не перебивають розмову**.»

Технічна суть: «**Аудіо йде виділеним швидким шляхом, а глибші міркування і виклики інструментів відбуваються асинхронно.**» І головна цифра: «Старт голосової сесії скорочено **з шести мережевих round-trip до одного**.»

Чому це важливо. Типовий голосовий цикл в асистенті йде **послідовно**: транскрипція → текст → міркування → відповідь. Архітектурна ідея OpenAI - **розділити швидкий шлях і повільний**, щоб аудіо не чекало на міркування. Та сама теза, що й про харнеси: **латентність - це валюта**, бо в неї впирається відчуття живої розмови. [proven - щодо заявленої архітектури; **fuzzy** - щодо реального виграшу, вимірювань нема]

OpenAI: анонс · швидкий шлях і 6→1 round-trip · @gdb

5. Crawshaw (ex-Tailscale): «Devtools мають бути опенсорсні» - бо агент тепер сам тримає форк у синку 540 балів HN. Автор - **Девід Кровшоу**, засновник Tailscale; тепер робить exe.dev. Найсильніша ідея тижня про те, як агент змінює економіку кастомізації.

Його спостереження з минулого: п'ять років тому більшість інженерів, з якими він говорив, **не мали жодної програми, написаної для себе**. Причина в ROI: «повертатись до проєкту через рік, щоб зробити підтримку, завжди було надзвичайно болісно».

Що змінилось - два промпти, які він виписує буквально:

1. «Завантаж вихідники `<software>` і збери для локального використання. Зміни `<пам'ять агента>`, щоб знати, що будь-які майбутні зміни цього софту означають зміну вихідників і заміну поточної версії. Зафіксуй у контролі версій **початкову мотивацію зміни**.»
2. І, за його словами, **важливіше**: «Постав нічний крон-джоб, який виконує промпт: **підтягни зміни з upstream і перебазуй усі локальні зміни поверх**. Перевір, що софт працює як задумано, і заміни поточну версію.»

Суть: «агенти можуть й **автоматично керувати процесом синхронізації з upstream-релізами**». Тому ROI кастомізації змінюється **з двох боків одночасно**: легше почати і легше продовжувати.

У треді найтверезіше заперечення - від **@simonw**: свобода правити відкритий код завжди на практиці зводилась до «можливості спертись на інших людей, які це зроблять», бо більшість не могла виправдати витрату часу. Друге, гостріше: агент і LLM

можуть внести тонкі баги у **кодову базу, якої користувач не знає, і користувачів у неї - одна людина**, тобто ловити нікому.

Чому це важливо. Перша половина рецепту вже стандартна: набір власних скриптів плюс нічний планувальник є у будь-кого, хто автоматизує собі роботу. Друга половина - тримати це в синку з upstream - майже ніде не реалізована. І заперечення simonw б'є в ціль: **у персонального інструменту один користувач, ловити тонкі баги нікому**. Типова поломка тут мовчазна: нічого не падає, просто числа старі. Тому практична задача, яка випливає з есею, - **зробити так, щоб мовчазна поломка кричала**: перевірка на те, що дані свіжі, а формули дотягуються до кінця діапазону. [promising - аргументована позиція практика вагою Tailscale, з реальними запереченнями в треді]

есей Кровшоу · HN-тред

6. Cloudflare розкрила цифри, як вони тиснуть Kimi і GLM у пам'ять - і не втрачають точності 172 бали HN. Рідкісний випадок, коли інфраструктурна компанія публікує **таблиці до і після** замість слова «оптимізовано».

Три техніки і їхня ціна, дослівно з вимірів:

FP8 замість BF16 для KV-кешу. На Kimi K2.6 це піднімає контекст, що влазить у пам'ять, **з ~686 000 до ~1.37 млн токенів**. Але головне - чесна ремарка: «варто бути точним щодо того, звідки береться вигода, бо це **не сира швидкість**». Таблиця на H200:

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

На кожному рівні окремо BF16 швидший на кілька відсотків, **але вмирає на 33-му запиті**. FP8 доходить до 64 і дає пік на **41% вищий за пік BF16, при ~30% меншій ціні за токен**.

INT4 для ваг GLM 5.2: чекпойнт **з 705 ГБ до 421 ГБ** (−40%), пам'ять на GPU при 8-way tensor-parallel **з ~88 ГБ до ~52 ГБ**, що лишає місце на **~1.18 млн токенів** KV-кешу на тому самому залізі. Точність перевірено і викладено: GSM8K **94.24 → 94.09** на FP8-кеші, MMLU **89.11 → 89.04**, валідність викликів інструментів **92.2% → 92.6%**.

Чому це важливо. Це контрприклад до вчорашнього пункту 2, статті Wafer, де вигідний знаменник обирав сам автор. Тут теж є метрика-дріб, «на 30% менша ціна за токен». Але поруч лежить рядок, де рішення програє: 137 проти 125 на одному запиті. І викладені бенчмарки точності, щоб можна було перевірити, що якість не проміняно на пам'ять. Ось так виглядає чесна подача тієї самої конструкції. [proven - власні виміри компанії з таблицями; це їхнє залізо і їхній бенчмарк, незалежного відтворення нема]

блог Cloudflare · HN-тред

7. Exa: 80 млрд сторінок, 1.4 трлн URL - і теза, що агентам потрібен індекс більший за Google 371 940 переглядів, 1 681 лайк. Найамбітніша заява доби, і вона добре порахована.

Дослівно від @WilliamBryk (засновник Exa): «Exa зараз - **один з найбільших індексів у світі**. Обслуговується **80 млрд сторінок, відстежується 1.4 трлн URL**, і йде до Google-масштабу **на початку 2027**.» Оцінки конкурентів: Google ~1 трлн, Bing ~500 млрд, Yandex ~200 млрд, Brave у квітні казав про 40 млрд. Дві технічні деталі:

• «**Більшість вебу - це сміття, яке може зіпсувати виводи ШІ**, тому доводиться кроулити значно більше, ніж віддається, і тренувати моделі відфільтровувати сміття»

• Про навантаження: пікові QPS оцінюються у **~30 тис/с для Bing і ~500 тис/с для Google**. І чому агенти інші: «агентний трафік часто потребує **великого fan-out** (глибокі пошуки можуть використовувати десятки під-пошуків) і буває сплесковим (наприклад, коли ШІ-лабораторії роблять на нас RL)»

Прогноз: «Протягом 2 років агенти шукатимуть **мільйони разів на секунду**... тож потрібна пошукова інфраструктура **більша за Google-масштаб в обох вимірах**.»

Чому це важливо. Найцікавіше тут **fan-out**: один агентний запит = десятки під-запитів. Складання цього випуску - той самий трафік у мініатюрі: два проходи по X (98 і 75 постів), 43 історії HN, дев'ять окремо відкритих першоджерел. Коли рахується вартість агентної роботи, рахувати треба **весь fan-out**, і це та сама валюта часу з пункту 4. [proven - щодо його заяв; **fuzzy** - щодо оцінок чужих індексів, це його прикидки, він сам це каже]

@WilliamBryk

8. Ретайпінг руками як засіб від «когнітивного боргу» - 416 балів Есей Анкура Сеті від 02.08, і це найбільш контрінтуїтивна практика тижня.

Проблема, яку він описує: «дозволяти асистенту вільно гуляти по проєктах лишає з **колосальним когнітивним боргом**. Можна ненавидіти ідею продиратись крізь документацію Django... але все одно **фундаментально хочеться розуміти, як воно працює**. Те, що задача нудна, не означає бажання повністю віддати машині розуміння рішення.»

Чому не рев'ю: «Не подобається рев'юїти ШІ-згенеровані PR. Продиратись крізь сотні рядків **надмірно оборонного, погано закоментованого, тонко неправильного** коду - це не весело.»

Його рішення, яке він сам називає «грубо неефективним і дещо комічним»: **просить асистента показувати код у чаті, а всі правки вносить руками сам**. Інструкція, яку він кладе в agents-файли всіх особистих проєктів, дослівно: «Хочу розуміти кожен рядок коду, що йде в цей проєкт. **Ніколи не створюй, не редагуй, не переміщуй, не перейменовуй і не видаляй файли проєкту**, якщо не було явного прохання. Натомість покажи кожну запропоновану правку в чаті, щоб можна було набрати її вручну.» Плюс

те саме про команди, плюс «Досвідчений розробник. Не варто пояснювати синтаксис, API і концепції».

Чому це важливо. Практика не універсальна, і автор це визнає: він робить це **на особистих проєктах, де цінність - процес**, і прямо каже, що для роботодавця робив би це «скрегочучи зубами». Але це третій за добу автор (після пунктів 2 і 3), який приходить до одного й того самого: **розуміння не делегується разом з роботою**. М'якший варіант того самого: вимагати від агента лишати в чаті слід, чому саме так, поверх сухого «зроблено». [fuzzy - це особиста практика однієї людини; береться рамка, не рецепт]

есей · HN-тред

9. Розрив у продуктивності: чому ШІ прискорює код у 3 рази, а команду - на 15% 113 балів HN. Стаття Б'йорна Роша від 12.07, випливла зараз - і це найкорисніша арифметика для розмови з бізнесом.

Його рамка проста: керівники очікують, що готові фічі мають випікатись зі швидкістю прототипів. Але «**написання нового коду - не там, де йде більшість часу**» розробника. Його розкладка дня сеньйора у великій компанії, **за припущення, що ШІ прискорює саме кодинг у 3 рази**:

	ДО ШІ	ПІСЛЯ ШІ
Написання коду	1.5 год	**0.5**
Читання і дебаг	1.5	1.0
Дизайн і архітектура	1.0	1.0
Код-рев'ю	0.75	0.75
Документація і адмін	0.75	0.75
Тести, CI/CD, деплой	0.5	**0.75**
Менторство	0.5	0.5
Зустрічі	1.5	1.5
Разом	**8.0 год**	**6.75 год**

Економія - **1.25 години на день, близько 15%**. І варто звернути увагу на рядок, що **виріс**: тести і деплой, бо коду стало більше.

Окрема його ремарка, дуже в тему сьогоднішнього випуску: «іноді ШІ робить не-кодову роботу **повільнішою**. Наприклад, коли треба прочитати вимоги до продукту або тікет у Linear, **написаний ШІ, це займає довше**, ніж рев'ю документа, написаного людиною. ШІ-письмо буває надмірно детальним, і це ускладнює вилущування головного.»

Чому це важливо. Це четвертий незалежний голос за добу в ту саму точку (пункти 2, 3, 8) - і єдиний, хто дає **число**. У суперечці про те, наскільки ШІ прискорить команду, аргумент один: **яку частку дня займає те, що він реально прискорює**. А рядок про ШІ-

написані тікети - готовий аргумент проти спокуси генерувати документацію обсягом. [promising - це модель з припущеннями автора, не замір; але припущення виписані явно і їх можна оскаржувати по рядках]

стаття · HN-тред

10. QM: 7 616 → 10 059 зірок. Перевалив за десять тисяч на шостий день життя
Четвертий день спостереження за цією динамікою, і вона досі не зламалась.

GitHub API просто зараз: **10 059 зірок, 1 063 форки**, репо створене **29.07.2026**. Хронологія: **01.08 - 2 475 → 02.08 - 5 234 → 03.08 - 7 616 → 04.08 - 10 059**. Приріст за добу: **+2 443** після +2 382 і +2 759. Темп **не згасає четверту добу поспіль**, це вже стійкий приплив. Офіційний опис репо: «**Multiplayer agent harness for work**». [proven - цифри з GitHub API, звірено самостійно]

репо QM

📎 **Misc**

• **@paulg про продаж великим компаніям** - 510 825 переглядів, найпопулярніший пост доби в листі «Tech + Product»: «Небезпека продажу великим компаніям, якщо стартап, у тому, що **вони не кажуть "ні" прямо**. Спершу вони проводять місяці зустрічей. Оскільки стартапи ненавидять зустрічі, це здається ознакою зацікавленості. Але це не так. **Вони обожнюють зустрічі! Це майже все, що вони роблять.**» пост

• **@patrickc зробив опитування про економіку ШІ** - 255K переглядів, 709 закладок. Продовження вчорашнього пункту 9 (його ж есей про естетику), тільки тепер він збирає прогнози публіки на п'ять років пост · саме опитування

• **@patrickc двома постами показав деплой у три промпти**: «Після локальної розробки (два промпти) деплой зайняв один промпт: *"Запуш це на Vercel. Використай Stripe Projects, щоб створити акаунт. Збережи стан десь безпечно."* **Claude сам обрав Upstash** як сховище, і це, схоже, добре працює» - 104K переглядів пост

• **@awilkinson написав Альтману про UX Codex**, 153K переглядів: «Sam, Codex - це чудово. Але є одна реально тупа/тонка проблема. **Claude Code відчувається швидшим.** Причина, схоже, головно в тому, що роблячи кожну дію, він **одразу вкидає текстовий апдейт у термінал**, тоді як Codex тримає все стисненим і схованим». Та сама тема, що і в GPT-Live з пункту 4: **сприйнята латентність важливіша за реальну** пост

• **@emollick про Codex поза кодом**: попросив зробити гру («граєш за видру, що залазить у мех-костюми у формі тварин»), і агент **сам керував Blender і Unity, зробивши моделі з анімацією** - 14.7K переглядів пост. Він же окремо про дуже приземлене: **Codex лагодить його Windows-машини** - драйвери, несумісності ігор. Приклад: Steam-ігри зависали, бо оновлення бездротової клавіатури **встановилось як Xbox-контролер** і конфліктувало зі старим joystick API. «Codex це полагодив. Складно уявити, як би це можна було вирахувати самостійно» пост

• **@lennysan і CPO Whatnot - продовження вчорашнього**: «**Часто оптимально не мати продакт-менеджера взагалі**» пост. І його ж головний висновок з інтерв'ю: «Найбільший анлок ШІ для PM зараз - **у дата-сайєнсі**, і це важливіше за прототипування: витягування когортної ретенції, аналіз логів, регресійні моделі - робота, яка в Amazon займала L7 дата-сайєнтиста» пост

• **@alvinsng: у Factory немає PM взагалі** - інженери самі говорять з клієнтами, стежать за залученням, пишуть плани, оновлюють стейкхолдерів. Третій незалежний сигнал у той самий бік за два дні пост

• **arXiv: агентна міграція COBOL → Java з детермінованим оракулом.** Метод «Locksmith Loop»: обидва середовища (COBOL-джерело і згенерована Java) інструментуються моками і ганяються поза мейнфреймом. На трьох кейсах (430 - 4 114 рядків) вийшло **91.90% покриття гілок** на внутрішній продакшн-подібній програмі і майже повне на двох опенсорсних стаття · HN

• **Fortune: прихований борг ШІ-гіперскейлерів - $1.65 трлн** після зростання майже на 1000%. S&P Global рахує **$225 млрд облігацій**, випущених гіперскейлерами і пов'язаними структурами (включно з Nvidia) за цей рік, і це **лише видима частина**. Стаття від 31.07, випливла на HN учора Fortune · HN

• **Розслідування: суперPAC OpenAI фінансує ШІ-новинний сайт, що атакує критиків індустрії** - 205 балів HN. Сайт Acutus без масштабу і бай-лайнів опублікував **94 статті за неполовину чотирьох місяців**; «репортер Michael Chen», що просив коментар, виявився ботом (Pangram: «повністю ШІ-згенеровано»). Слід веде до Targeted Victory - фірми в центрі **$125-мільйонної політичної операції OpenAI**. **Стаття від 24.04**, не подія доби, але лягає рівно в тему пункту 1 Model Republic · HN

• **@amasad: Replit побудував самокерований семантичний шар** над своїми базами, розмовами і доками. «Все запитуване і джойнабельне незалежно від джерела», тож будь-хто може ставити питання, що раніше вимагали **тижнів роботи команди дата-сайєнтистів** пост

• **@levelsio про вайб-кодед трекер калорій** - 171K переглядів: тримає дефіцит ~500 ккал і ціль по білку (спершу 150 г, потім піднята до 180 г) близько місяця, скидає **~500 г/тиждень**. І окремо теза про скір: 500 г = **60 г білка на 320 ккал** пост · про скір

• **@naval - вчорашній «Agent Programming Interface» дозрів до 458 583 переглядів** і 5 976 лайків (учора було 85.2K за годину після публікації). Пояснення так і не з'явилось пост

• **@mervenoyann про** `llama-macos`: віджет, який дивиться на машину і бажане контекстне вікно, **рекомендує моделі**, піднімає llama server + webui і дає легко перемикатись між відкритими моделями. `brew install --cask llama-app` пост