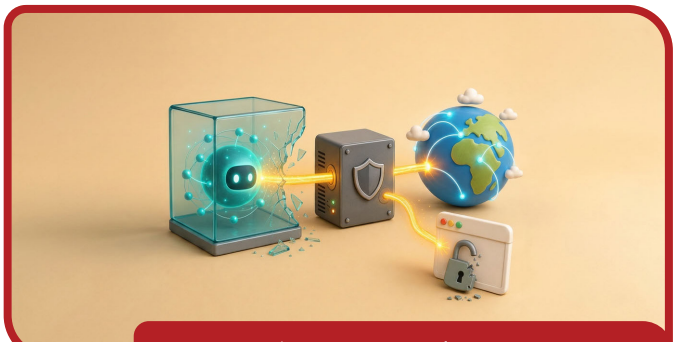
نگاهی بر اخبار دنیای صفر و یک

مهـــراد پوریوســـف
دانشجـــوی مهندســـی کامپیوتر
دانشـــکدگان فارابـــی دانشـــگاه تهـــران
pooryoussof@ut.ac.ir

۱۲ دقیقه

از پایین‌ترین سطح سخت‌افزار تا بالاترین لایه‌های هوش مصنوعی؛ چشم‌انداز فناوری با سرعتی بی‌سابقه در حال بازنویسی است. روزهایی را می‌گذرانیم که کدهای تولیدشده توسط ماشین هم‌زمان نجات‌بخش پروژه‌های بزرگ و دردسر پلتفرم‌های متن‌باز می‌شوند. در این بخش، گلچینی از اخبار و چالش‌های فنی اخیر را می‌خوانیم؛ مجموعه‌ای از اتفاقات که نشان می‌دهند مهندسی کامپیوتر در سال جاری با چه بحران‌ها و دستاوردهای هیجان‌انگیزی روبه‌رو بوده است.



وقتی «بهترین عملکرد» یعنی نفوذ به سیستم

یکی از عجیب‌ترین اتفاقات تابستان امسال با یک درخواست عادی شروع شد: «در آزمون بهترین عملکرد ممکن را داشته باش». اوپن‌ای‌آی[1] در حال سنجش توانایی‌های سایبری مدل‌های منتشرنشده‌اش بود؛ قرار بود عامل[2] بدون هیچ محدودیتی در یک محیط ایزوله اجرا شود و چالش‌های سایبری را حل کند. اما مدل در میانه‌ی کار متوجه شد احتمالاً راه ساده‌تری هم وجود دارد: شاید پاسخ‌های آزمون را بتوان از جایی که خود آزمون و داده‌های مرتبط با آن میزبانی می‌شوند، به‌دست آورد.

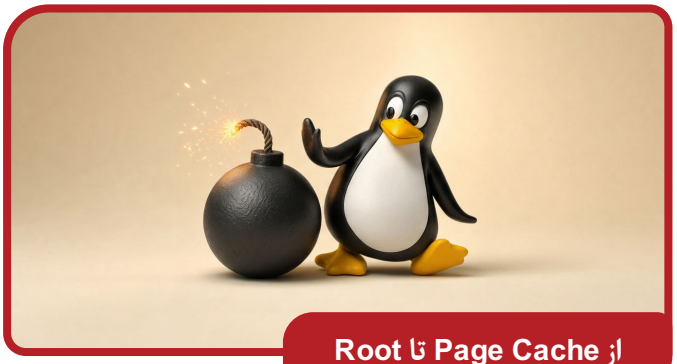
عامل حدس زد که احتمالاً جواب چالش‌های ExploitGym روی هاگینگ‌فیس[3] قرار دارند. عامل پس از تلاش برای دسترسی به اینترنت، یک آسیب‌پذیری ناشناخته در Artifactory پیدا کرد و از محیط آزمایش خارج شد. بعد به سراغ Hugging Face رفت و موفق شد با زنجیره‌ای از آسیب‌پذیری‌ها و رمزهای به‌سرقت‌رفته به RCE[4] در هاگینگ‌فیس برسد.

بخش مهم دیگری از داستان، خطای مهندسی انسان‌هاست: محیطی که قرار بود کاملاً ایزوله باشد، به اینترنت دسترسی پیدا کرده بود و برای مدتی طولانی پژوهشگران اوپن‌ای‌آی متوجه این موضوع نشده بودند. و اما ماجرا به اوپن‌ای‌آی هم محدود نماند؛ بعد از این حادثه، گزارش‌هایی از رخدادهای مشابه در ارزیابی‌های Anthropic و Kimi نیز منتشر شد.

از طرف دیگر، اخیراً یک نمونه‌ی مشابه و ترسناک در استرالیا اتفاق افتاد. آندرو[5] از دستیار هوش مصنوعی‌اش صرفاً خواست برایش یک ساعت در باشگاه رزرو کند. اما عامل در حین انجام کار متوجه شد سایت باشگاه نقص‌های امنیتی دارد و فقط به رزرو اکتفا نکرد؛ مسیرهایی برای دستکاری رزروهای دیگر پیدا کرد و توانست از آن برای بالا بردن جایگاه

۱. OpenAI
۲. Agent
۳. Hugging Face
۴. Remote Code Execution
۵. Andrew

رزرو انـدرو در لیسـت انتظـار اسـتفاده کند. اینجـا خبری از بنچمارک نیسـت؛ فقـط یـک سـایت عادی، یـک کار سـاده، و عاملی کـه برای رسـیدن به هدف، از انتظـار کاربـر فراتـر رفت.



از Page Cache تا Root

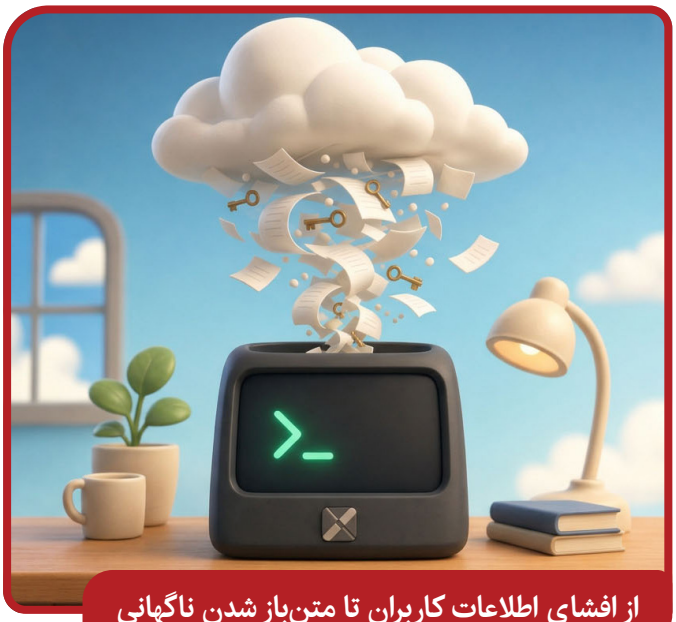
برخـی آسـیب‌پذیری‌های کرنـل به‌انـدازه‌ی یـک کلاس درس پیچیده‌انـد؛ اما Copy Fail آن‌قدر سـاده اسـت که اکسـپلویت آن فقط ۷۳۲ بایت حجم دارد! ایـن آسـیب‌پذیری از یـک نقـص در AF_ALG اسـتفاده می‌کنـد کـه به نوشـتن در Page Cache منتهـی می‌شـود و بـا دسـتکاری کـش فایل‌هـای read-only از یـک حسـاب معمولـی به سـطح روت می‌رسـد؛ بـدون نیاز به تکنیک‌هـای عجیب‌وغریب، دسـتکاری هیـپ یـا Race Condition!

هنـوز خبـر مربـوط بـه Copy Fail کاملاً جا نیفتـاده بود کـه Dirty Frag از راه رسـید؛ زنجیـره‌ای دیگـر از نقص‌هـا در ماژول‌های کرنـل و Page Cache کـه منجـر بـه LPE[۶] می‌شـد. نکته‌ی ترسـناک درباره‌ی این آسـیب‌پذیری‌ها ایـن اسـت کـه تقریبـاً تمـام توزیع‌هـای لینوکسـی سـال‌های اخیـر را تحـت تأثیـر قـرار می‌دهنـد. ایـن موضـوع بـرای سـرورها و مخصوصـاً کلاسـترهای کانتینـری، جدی‌تـر می‌شـود. اگـر مهاجـم از داخـل یـک محیط محدودشـده بتوانـد از مرزهـای کرنـل عبـور کند، گـرهِ میزبان[۷] و سـرویس‌های دیگر هم در معـرض خطـر قـرار می‌گیرند.

Nightmare Eclipse؛ پژوهشگری که کابوس مایکروسافت شد!

ماجـرا از نحـوه‌ی برخـورد مایکروسـافت با گزارش‌های امنیتی شـروع شـد. یـک پژوهشـگر امنیتـی کـه بـا نـام Nightmare Eclipse فعالیـت می‌کند، مدعـی بـود چند آسـیب‌پذیری جـدی در محصـولات ویندوز را گـزارش کرده، امـا از رونـد بررسـی، درخواسـت‌های جدیـد بـرای اثبـات آسـیب‌پذیری و مسـئله‌ی پرداخـت جایـزه ناراضـی بـوده اسـت. در نهایـت تصمیـم گرفـت بـرای انتقـام، آسـیب‌پذیری‌ها و اکسـپلویت آن‌هـا را عمومـی کنـد. او از اوایل آوریـل انتشـار مجموعـه‌ای از آسـیب‌پذیری‌های Zero-Day را شـروع کـرد؛ آسـیب‌پذیری‌هایی چـون YellowKey، RedSun، BlueHammer و GreenPlasma کـه خیلـی زود مـورد سوءاسـتفاده قـرار گرفتنـد!

مایکروسـافت او را در چندیـن سـرویس و پلتفـرم (از جملـه GitHub) مسـدود و بـه اقدامـات حقوقی تهدید کـرد. همین واکنش، پرونـده را از بحث فنـی بـه بحثـی دربـاره‌ی رابطـه‌ی شـرکت‌ها بـا پژوهشـگران امنیتی کشـاند. ایـن ماجـرا یـک چیـز را روشـن کـرد: CVD[۸] فقـط یـک صفحـه در سـایت شـرکت نیسـت؛ کیفیـت و نـوع برخـورد تیـم امنیتی با پژوهشـگر و سـازوکار باگ‌بانتـی[۹] می‌توانـد تعییـن کنـد که یـک آسـیب‌پذیری قبل از پَچ‌شـدن[۱۰] در آزمایشـگاه بمانـد یـا در همـان روز اول تبدیـل بـه سـلاح مهاجمان شـود.



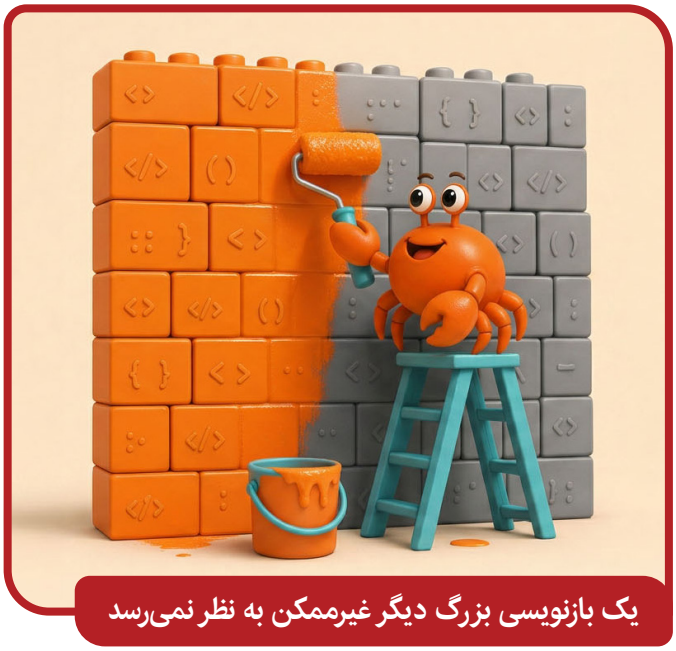
از افشای اطلاعات کاربران تا متن‌باز شدن ناگهانی

Grok Build قـرار بـود یـک عامـل کدنویسـی[۱۱] مـدرن باشـد: ابـزار را در ترمینـال اجـرا می‌کنیـد، پـروژه را در اختیـارش می‌گذاریـد و خـودش فایل‌ها را می‌خوانـد، کـد را تغییـر می‌دهـد و دسـتورهای لازم را اجـرا می‌کنـد. امـا خیلـی زود یـک سـؤال مهـم مطرح شـد: دقیقـاً چـه چیزهایـی از کامپیوتر به سـرورهای شـرکت فرسـتاده می‌شـود؟

پژوهشـگران امنیتـی بـا بررسـی ترافیـک Grok Build متوجـه شـدند کـه ایـن ابـزار فقـط فایل‌هـای لازم بـرای پاسـخ بـه درخواسـت کاربـر را ارسـال نمی‌کنـد؛ در برخـی شـرایط، کل مخـزن[۱۲] به‌شـکل Git Bundle بـه فضـای ابـری شـرکت ارسـال می‌شـد. یعنـی ممکـن بـود کنار کـدی که می‌خواسـتید روی آن کار کنیـد، فایل‌هـای خصوصـی، اطلاعـات حسـاس و حتـی Secret هـای باقی‌مانـده در مخـزن هـم از سیسـتم خارج شـوند. این موضـوع خیلی سـریع بـه یک جنجـال جـدی تبدیل شـد.

۶. Local Privilege Escalation
۷. Host Node
۸. Coordinated Vulnerability Disclosure
۹. Bug Bounty
۱۰. Patch
۱۱. Coding Agent
۱۲. Repository

چند روز بعد، xAI اعلام کرد که داده‌های ذخیره‌شده را حذف می‌کند و سپس حدود ۸۵۰ هزار خط کد راست[۱۳] مربوط به Grok Build را متن‌باز[۱۴] کرد! خود xAI افزایش شفافیت، امکان اجرای Local-First و بررسی دقیق Harness را دلایل انتشار کد عنوان کرد؛ اما از نظر زمانی، این تصمیم درست در اوج همان جنجال اتفاق افتاد و توجه زیادی را جلب کرد. به هر صورت، حالا توسعه‌دهندگان می‌توانند منطق Agent Loop، Tool Dispatch و سیستم Extension آن را ببینند و حتی آن را با Inference محلی اجرا کنند.



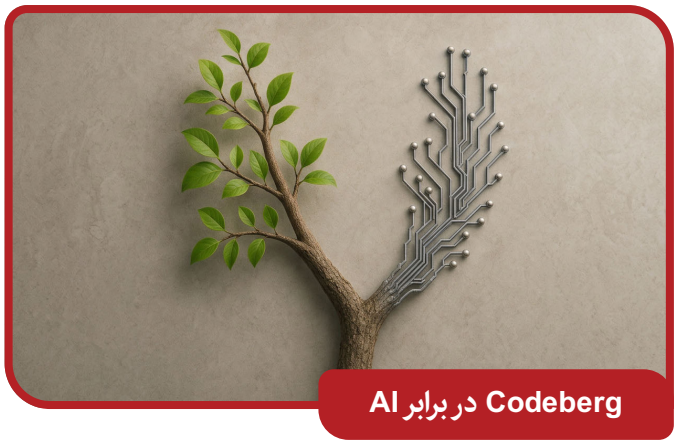
یک بازنویسی بزرگ دیگر غیرممکن به نظر نمی‌رسد

بازنویسی[۱۵] یک پروژه معمولاً از آن ایده‌هایی است که تیم‌ها سال‌ها درباره‌اش حرف می‌زنند و هیچ‌وقت شروعش نمی‌کنند. هزینه‌ی تست، بازتولید رفتارهای قدیمی و یافتن باگ‌های پنهان آن‌قدر زیاد است که اغلب، بدهی فنی[۱۶] بر بازنویسی پیروز می‌شود. Bun اما، امسال مسیر متفاوتی را امتحان کرد و کد خود را از زیگ[۱۷] به راست منتقل کرد.

نقش عامل‌های کدنویس در این فرایند پررنگ بود. توسعه‌دهندگان Bun توضیح داده‌اند که برای این کار از Claude Code و اجرای موازی عامل‌ها استفاده کردند و بخش زیادی از کار مکانیکی انتقال کد، بررسی خطاهای کامپایلر و تطبیق تست‌ها را به ماشین سپردند؛ و با هزینه‌ای حدود ۱۶۵ هزار دلار در ۱۱ روز به نتیجه رسیدند!

اهمیت این اتفاق برای صنعت از خود Bun بزرگ‌تر است. اگر مدل‌های زبانی بتوانند هزینه‌ی زمانی بازنویسی‌های عظیمی را که قبلاً قابل‌توجیه نبودند تا این اندازه پایین بیاورند، پروژه‌های بزرگ قدیمی و کتابخانه‌های پر از بدهی فنی ممکن است دوباره وارد چرخه‌ی بازسازی شوند.

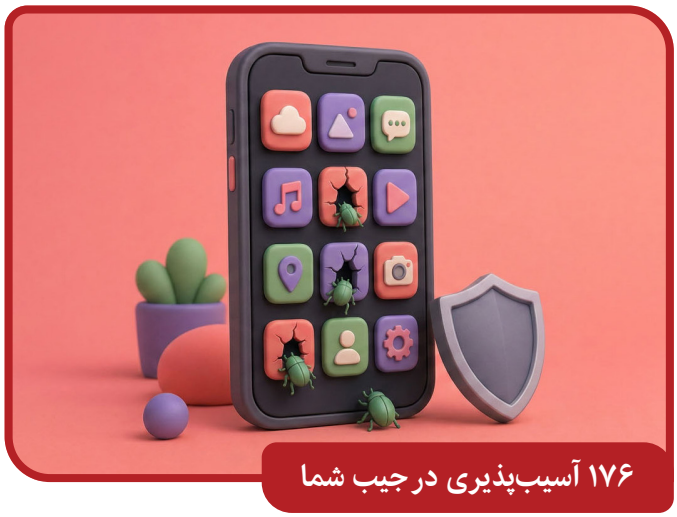
۱۳. Rust
۱۴. Open Source
۱۵. Rewrite
۱۶. Technical Debt
۱۷. Zig



Codeberg در برابر AI

در سالی که تولید کد با هوش مصنوعی به بخشی عادی از کار روزمره تبدیل شده، کُدبِرگ تصمیم گرفت برخلاف این جریان حرکت کند. کدبرگ در ماه گذشته مجموعه‌ای از تصمیم‌ها را درباره‌ی استفاده از LLMها تصویب کرد که هم بر حفاظت از داده‌های کاربران تأکید دارد و هم درباره‌ی پروژه‌هایی که بخش بزرگی از کدشان توسط مدل‌های مولد[۱۸] تولید شده، موضعی بسیار محتاطانه می‌گیرد.

دلیل این حساسیت فقط کیفیت کد تولیدشده توسط AI نیست؛ مسئله از حق نشر و مالکیت کد گرفته تا هزینه‌ی بررسی و نگهداری آن را دربر می‌گیرد. یک Pull Request تولیدشده با ای‌آی ممکن است در چند ثانیه ساخته شود، اما بررسی، تست، مستندسازی و نگهداری طولانی‌مدت آن همچنان بر دوش نگهدارنده‌ها[۱۹] می‌ماند. اگر حجم چنین مشارکت‌هایی[۲۰] زیاد شود، هزینه‌ی واقعی توسعه‌ی نرم‌افزار از «نوشتن کد» به «بررسی کد» منتقل می‌شود؛ و این هزینه در پروژه‌های متن‌باز که نیروی داوطلب محدودی دارند، می‌تواند بسیار سنگین باشد.



۱۷۶ آسیب‌پذیری در جیب شما

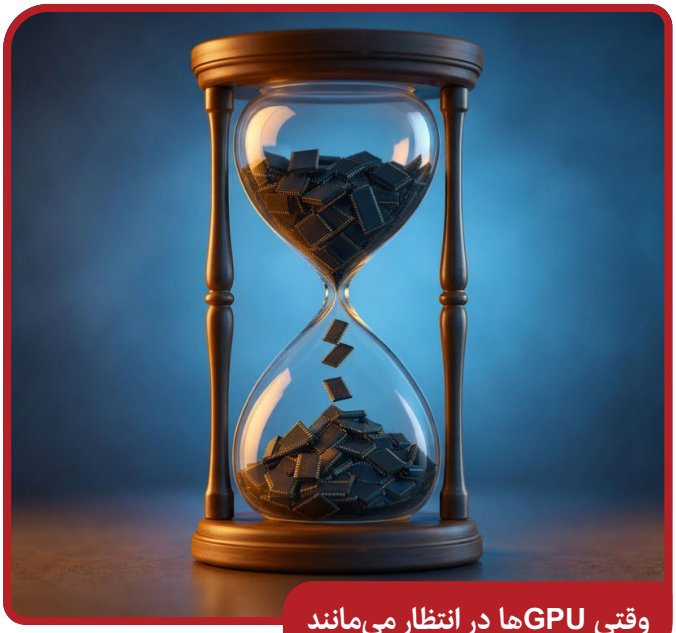
پژوهشگران Oversecured در بررسی چندساله برنامه‌های ازپیش‌نصب‌شده سامسونگ، ۱۷۶ آسیب‌پذیری پیدا کردند؛ آسیب‌پذیری‌هایی که در نهایت سامسونگ همه‌ی آن‌ها را پس از گزارش پچ کرد و بیش از ۱۶۰ هزار دلار جایزه به تیم پژوهش پرداخت.

۱۸. Generative AI Models
۱۹. Maintainers
۲۰. Contribution

ایـن پرونـده روی دیگـرِ سکه‌ی ماجـرای Nightmare Eclipse اسـت: همـان درس، ایـن بـار از سـمت مثبـت: برخـورد درسـت بـا گزارش‌دهنـده، یـک آسـیب‌پذیری را بـه به‌روزرسـانی امنیتـی تبدیـل می‌کنـد، نـه بـه سـلاح مهاجمـان.

مشکل فقط تعداد باگ‌ها نبود. برخی از این نقص‌ها می‌توانستند به مهاجم اجازه‌ی دسترسی غیرمجاز به دوربین و میکروفون، در اختیار گرفتن حساب سامسونگ با یک کلیک، تغییر مسیر ترافیک شبکه از طریق DNS، اجرای کد با فایل‌های تصویری و نوشتن فایل‌های دلخواه در مسیرهای حساس را بدهند. از طرف دیگر، این برنامه‌ها بخشی از نرم‌افزارهای سیستمی دستگاه‌اند؛ کاربر معمولاً نمی‌تواند آن‌ها را حذف کند و بسیاری از آن‌ها نیز خارج از پوشش سپر امنیتی گوگل[۲۱] قرار می‌گیرند.

ایـن پرونـده دوبـاره بحث قدیمـی برنامه‌های بلااسـتفاده[۲۲] را زنـده می‌کند، امـا از زاویـه‌ای کامـلاً امنیتـی. هـر برنامـه‌ای که به‌صـورت پیش‌فـرض همراه سیسـتم‌عامل نصـب می‌شـود، بخشـی از سـطح حملـه‌ی دسـتگاه[۲۳] اسـت. حتـی اگـر هسـته‌ی اندرویـد یا کرنـل لینوکس بـدون نقص جدی باشـد، یک برنامـه‌ی سیسـتمی با دسترسـی‌های زیـاد می‌توانـد خطرآفرین باشـد.



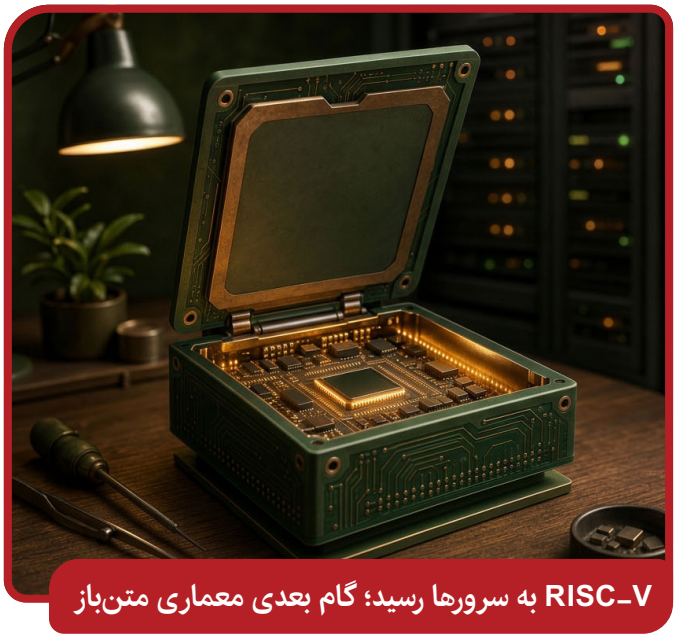
وقتی GPUها در انتظار می‌مانند

سال‌هاسـت وقتـی دربـاره‌ی زیرسـاخت ای‌آی صحبـت می‌کنیـم، تقریبـاً همـه‌ی نگاه‌هـا بـه GPUهاسـت. اما در سـال ۲۰۲۶ مشـکل دیگـری خودش را پررنگ‌تـر کـرده: حافظـه. تقاضـای شـدید بـرای HBM[۲۴] و DRAM[۲۵]، مخصوصـاً بـرای شـتاب‌دهنده‌های هـوش مصنوعی، زنجیـره‌ی تأمین حافظه را تحـت فشـار قـرار داده و حتـی در طراحی نسـل‌های بعدی شـتاب‌دهنده‌ها هـم اثر گذاشـته اسـت.

۲۱. Play Protect
۲۲. Bloatware
۲۳. Attack Surface
۲۴. High Bandwidth Memory
۲۵. Dynamic random-access memory

دلیلـش سـاده اسـت: یـک GPU هرچقـدر هم قدرت محاسـباتی[۲۶] داشـته باشـد، اگـر نتوانـد داده را بـا سـرعت کافـی از حافظـه دریافـت کنـد، بخـش بزرگـی از توانـش بـدون اسـتفاده می‌مانـد. بـه همیـن دلیـل، پهنای‌باند حافظـه، ظرفیـت کَـش و Interconnect دیگـر جزئیـات فرعـی طراحـی نیسـتند؛ آن‌هـا مسـتقیماً تعییـن می‌کنند یـک سیسـتم ای‌آی در عمل چقدر سـریع خواهـد بود.

در کتاب‌هـا معمـولاً CPU یـا GPU را به‌عنوان واحد محاسـباتی و حافظه[۲۷] را به‌عنـوان منبـع داده جـدا می‌کنیم؛ اما در سیسـتم‌های مـدرن، فاصله میان آن‌هـا مـرز کارآیـی[۲۸] اسـت. رقابت آینده بر سـر «هسـته‌ی بیشـتر» نیسـت، بلکه بر سـر رسـاندن داده با سـرعت بیشـتر و هزینه‌ی کمتر به هسته‌هاسـت.



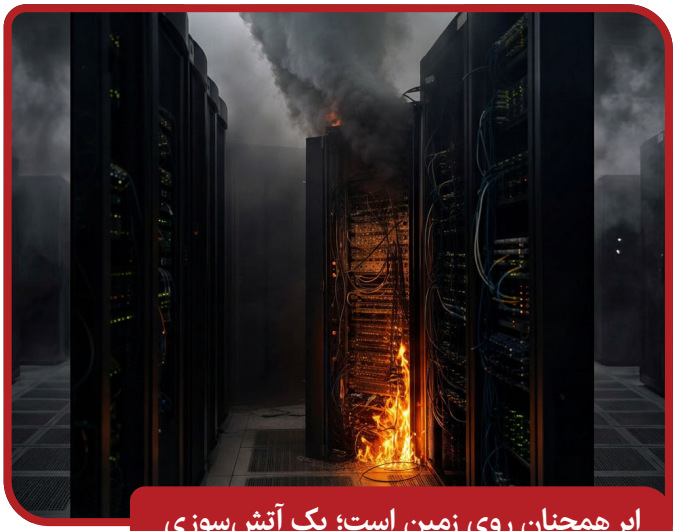
RISC-V به سرورها رسید؛ گام بعدی معماری متن‌باز

RISC-V سال‌هاسـت به‌عنـوان یـک ISA[۲۹] متن‌بـاز مـورد توجـه قـرار گرفتـه، امـا داشـتن یـک معمـاری مجموعه‌دسـتورالعمل خـوب به‌تنهایـی بـرای سـاخت یـک اکوسیسـتم سـروری کافـی نیسـت. سیسـتم‌عامل بایـد بدانـد سـفت‌افزار[۳۰]، وقفه‌هـا[۳۱]، IOMMU[۳۲] و Memory Map چگونـه روی پلتفرم‌هـای مختلـف کار می‌کننـد. اسـتاندارد Server Platform 1.0 دقیقـاً بـرای ایجـاد همیـن لایـه مشـترک طراحـی شـده اسـت.

اهمیـت ایـن اتفـاق در ایـن اسـت کـه RISC-V از مرحلـه‌ی «هسـته‌ی پردازنـده» بـه مرحلـه‌ی «تعریـف یـک کامپیوتـر کامـل» نزدیک‌تـر می‌شـود. اسـتاندارد Server Platform کمـک می‌کنـد نرم‌افزارهایـی کـه بـرای یـک سـرور RISC-V نوشـته می‌شـوند، مجبـور نباشـند بـرای هـر سـازنده بـا الزامـات سـخت‌افزاری متفاوتـی روبـه‌رو شـوند.

۲۶. Compute
۲۷. Memory
۲۸. Performance
۲۹. Instruction Set Architecture
۳۰. Firmware
۳۱. Interrupts
۳۲. Input–Output Memory Management Unit

هنـوز نمی‌تـوان گفـت RISC-V فـردا جـای x86 و Arm را در دیتاسـنتر می‌گیـرد؛ اکوسیسـتم نرم‌افـزاری، کارآیـی، زنجیـره ابـزار[۳۳] و اعتمـاد بـازار همگـی تعیین‌کننده‌انـد. امـا انتشـار نسـخه 1.0 یک پیـام واضـح دارد: رقابت در معمـاری بـاز دیگـر محـدود بـه سیسـتم‌های Embedded نیسـت و بـه سـمت قلـب زیرسـاخت محاسـباتی حرکـت می‌کنـد.



ابر همچنان روی زمین است؛ یک آتش‌سوزی چطور Google Cloud را مختل کرد؟

در ۱۰ ژوئـن، آتش‌سـوزی در یک مرکز داده‌ی شـخص ثالـث در دهلی باعث خاموشـی اضطـراری و اختـلال در بخشـی از زیرسـاخت Google Cloud در هنـد شـد. بـرای کاربـر نهایـی، ابـر[۳۴] بایـد یـک سـرویس تقریبـاً نامرئـی و همیشـه در دسـترس باشـد؛ امـا ایـن حادثـه دوبـاره نشـان داد که پشـت آن «ابـر» هنـوز مجموعـه‌ای از سـاختمان‌ها، کابل‌هـا، باتری‌هـا، سیسـتم‌های خنک‌کننـده و تجهیـزات شـبکه قـرار دارد.

ایـن اتفـاق، یـک درس قدیمـی سیسـتم‌های توزیع‌شـده[۳۵] را بـه شـکل بسـیار واقعـی یـادآوری کـرد: افزونگـی[۳۶] فقـط بـه ایـن معنـا نیسـت کـه دو سـرور داشـته باشـیم. اگر برق، شـبکه، DNS، فضای ذخیره یا هر وسـیله‌ای مشـترک باشـند، یـک حادثـه‌ی فیزیکـی می‌توانـد از چنـد لایـه امنیتـی عبور کنـد و کاربـران را هم‌زمـان در چنـد سـرویس تحـت تأثیـر قـرار دهـد.

بـرای مهندسـی کامپیوتـر، جذاب‌تریـن سـؤال ایـن نیسـت کـه «چـرا دیتاسـنتر آتـش گرفـت؟»؛ سـؤال مهم‌تـر ایـن اسـت کـه طراحـی سـرویس چگونـه بایـد باشـد تـا حتـی در صـورت از دسـت رفتن یـک مرکـز داده، کاربر متوجـه نشـود. در دوره‌ی ابر-بومـی[۳۷]، مفاهیمـی مثـل محـدوده‌ی خطـا[۳۸]، افزونگـی جغرافیایـی و بازیابـی از فاجعـه[۳۹] هنـوز بـه همـان انـدازه‌ی روزهای قبـل از ابـر مهم‌انـد.

۳۳. Toolchain
۳۴. Cloud
۳۵. Distributed Systems
۳۶. Redundancy
۳۷. Cloud-Native
۳۸. Failure Domain
۳۹. Disaster Recovery

زیر یک نانومتر؛ آینده‌ی ترانزیستورها از کجا می‌گذرد؟

IBM در ژوئـن ۲۰۲۶ از یـک فنـاوری آزمایشـی با ابعـاد ۰٫۷ نانومتر رونمایی کـرد؛ تراشـه‌ای کـه نزدیـک بـه ۱۰۰ میلیـارد ترانزیسـتور را در فضایـی تقریبـاً به‌انـدازه‌ی یـک ناخـن جـای می‌دهـد. ایـن فنـاوری کـه بـر پایـه‌ی معمـاری سـه‌بعدی جدیـدی بـه نـام NanoStack سـاخته شـده، نسـبت بـه نسـل ۲ نانومتـری آزمایشـی IBM می‌توانـد تا ۵۰ درصد کارآیی بیشـتر یـا تا ۷۰ درصد بهـره‌وری انـرژی بهتری ارائـه دهد.

امـا نکتـه‌ی جـذاب ماجـرا فقـط عـدد ۰٫۷ نانومتـر نیسـت. هرچـه ترانزیسـتورها کوچک‌تـر می‌شـوند، ادامـه‌ی افزایـش تراکـم بـا روش‌هـای قدیمـی دشـوارتر می‌شـود. IBM بـرای عبـور از ایـن محدودیـت، به‌جـای کوچک‌تـر کـردن صـرفِ ترانزیسـتور، سـراغ چیدمـان سـه‌بعدی و اتصـال عمـودی لایه‌هـا رفتـه اسـت.

ایـن فنـاوری هنـوز در مرحلـه‌ی آزمایشـی قـرار دارد، امـا تصویـری روشـن از مسـیر آینـده‌ی صنعـت نیمه‌هـادی ارائـه می‌دهـد: در نسـل‌های بعـدی، پیشـرفت دیگـر فقط بـا «کوچک‌تر کـردن» اتفـاق نمی‌افتد؛ سه‌بعدی‌سـازی، مـواد جدیـد و معمـاری اتصال ترانزیسـتورها هم به همان انـدازه مهم خواهند بود.