PRE-PUBLICATION DRAFT. This is the author's accepted version of an article accepted for publication in *IEEE Security & Privacy* (Notes from the Field). It has not been copyedited or typeset by IEEE, and the published version will differ in wording and layout. Please cite the published version: E. H. Spafford, "Don't Blame the Auditor: Why the Vulnerability Flood Is 40 Years of Willful Technical Debt," *IEEE Security & Privacy*, to appear, Nov./Dec. 2026, doi: 10.1109/MSEC.2026.3727025.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Don't Blame the Auditor

Why the Vulnerability Flood Is Forty Years of Willful Technical Debt

Eugene H. Spafford, Purdue University

Abstract — The current vulnerability flood is an audit, not a crisis: AI tools are reading flawed code the industry shipped and deferred fixing for decades. That is willful technical debt. The remedy is engineering discipline and AI-assisted rewriting, not more patching — and the liability disclaimer that financed deferral is eroding.

Keywords — software security; technical debt; vulnerability disclosure; memory safety; AI-assisted vulnerability discovery; software liability; software engineering practice; operational technology

On July 14, 2026, Microsoft released fixes for at least 570 vulnerabilities, more than it had ever shipped in a single month (counts vary with how trackers tally Edge and republished third-party advisories) [1]. This nearly triples the record set one month earlier, a record that had itself displaced one set the previous October. Roughly 60 of those fixes were rated critical, and about 250 addressed privilege escalation [1], an attack outcome enabled by defect classes that were characterized long before most of the code containing them was written.

Five days before the release, Microsoft executive vice president Pavan Davuluri told Windows customers to expect this as the new baseline. Writing on the Windows Experience Blog, he said advances in AI are "making it possible to find more issues, faster, across more code," and that customers should expect "a higher volume of security updates included in each security release" going forward [2]. That prediction was tested a month later. On August 11, 2026, Microsoft shipped fixes for nearly 400 vulnerabilities, with trackers again disagreeing on the exact count. Dustin Childs of the Zero Day Initiative called the volume "the new normal — at least for now."

Microsoft is not an outlier. Adobe shifted to twice-monthly security bulletins in July 2026, citing AI-driven discovery, and Ivanti's Chris Goettl noted that Google's June 2026 patch batches totaled more than 900 security fixes [1]. Oracle's quarterly Critical Patch Update for July 2026 contained 1,449 new security patches, spanning product families from banking platforms to telecommunications network functions. Although released on different cadences, taken together they approach three thousand security fixes in a single season from three of them, out of the hundreds of vendors whose products run critical applications.

The pattern extends well beyond a few large vendors. By mid-August, the 2026 CVE tally at cvedetails.com was past 51,000, already more than the whole of 2025, with four months of the year remaining (Figure 1). The first half of 2026 alone produced 35,364, a half-year total that exceeds any full

year in the program's history before 2024 [3]. That count undercounts actual flaws. The vulnerability tracker Flashpoint catalogs more than 100,000 publicly disclosed flaws that never received a CVE identifier.

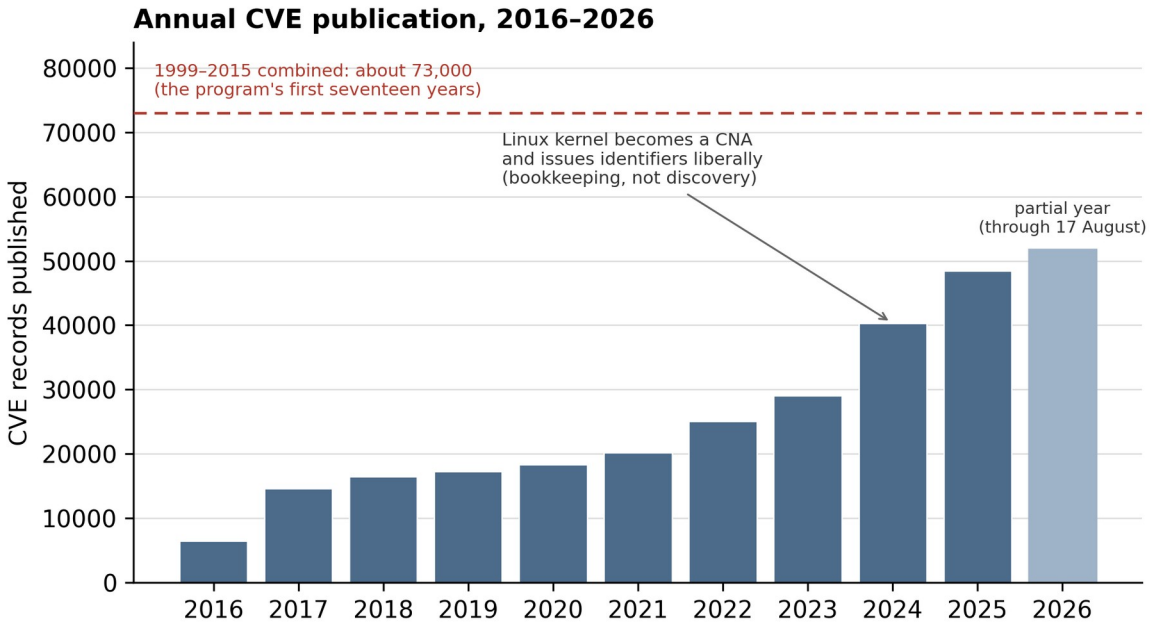


Figure 1. Annual CVE publication, 2016–2026. Bars are cvedetails.com annual counts, retrieved August 17, 2026; the 2026 bar is a partial year, covering January through August 17. The 1999–2015 comparison line is from Gamblin's cumulative catalog (308,920 records through 2025) [29] less the years charted; that tally excludes rejected records and so runs slightly below the bars. The 2024 step reflects both expanded CVE numbering authority activity and growth in discovery; the two cannot be separated from the published totals alone.

The CVE program began in 1999. Over its first seventeen years, 1999 through 2015, the program produced about 73,000 entries in total. Mid-2026 projections for the single year 2026 range from FIRST's revised estimate of about 66,000 to Jerry Gamblin's extrapolation above 70,000 [3]. One year of disclosure is now on pace to approach the program's first seventeen years combined. Admittedly, some of that corpus growth is bookkeeping rather than discovery. New CVE numbering authorities, including the Linux kernel maintainers' 2024 decision to issue identifiers liberally, inflate the raw counts. So do misidentified flaws: with no resources to validate the flood, false positives enter the record and stay there, which means simple counts overstate the threat even as they understate the work. The vendor-confirmed patch totals are the cleaner signal, and they point the same way. In April 2026, NIST gave up on fully enriching the National Vulnerability Database, conceding that record productivity gains were not keeping pace with submissions that grew 263 percent between 2020 and 2025 [4].

Prevailing narratives treat these numbers as an AI crisis. Capable agents became widely available, and the flood followed. The sequence is accurate, but the blame is misplaced. AI models wrote almost none of the flawed code. Instead, AI found flaws that were already present — some for decades — in products that sold well and were deployed nearly everywhere, including in hospitals, utilities, finance, and defense systems, despite long public histories of fault and misbehavior in those same products.

The age of that inventory is documented, increasingly by the toolmakers themselves. Anthropic's report on its Mythos preview dates several of its findings. One flaw was in OpenBSD's TCP stack, in

code added in 1998. Another gave unauthenticated remote root execution in FreeBSD's NFS code and went unnoticed for seventeen years [5]. Alexopoulos and colleagues ran a heuristic over thousands of CVEs in major open-source projects to recover the average interval between a vulnerability's introduction and the commit that fixed it: about four years [6]. Chromium's averaged two. OpenSSL's averaged seven. RAND's survival analysis of a private zero-day stockpile, a different population measured differently, lands in the same range [7]. Some recent cases ran longer. A character-set conversion flaw entered glibc in 2000 and was disclosed in 2024 as CVE-2024-2961. A netfilter flaw, CVE-2024-1086, was in Linux for a decade until ransomware crews exploited it.

Google's Android measurements find that vulnerability density is highest in new code and decays sharply as code ages [8]. New code is where defects are densest; the installed base is where they accumulate. The decay curve, though, is measured in discovered vulnerabilities per unit of code. Discovery was expensive throughout the period those measurements cover, and old code looked safe partly because nobody expended the resources to look at it closely. Cheap discovery should therefore flatten the observed curve — a prediction the next few years will test.

Those patch totals also count only reported and acknowledged security problems. The same production processes emit defects that carry no security label: crashes, wrong answers, corrupted state, functions that fail under load. The security-relevant fraction is the portion most likely to receive attention and investment, because disclosure programs, exploit markets, liability exposure, and publicity all focus triage toward it. The ordinary faults compete for whatever engineering attention remains. A reasonable inference from hundreds of security fixes per month for any vendor is that the code and its design carry proportionate populations of unglamorous defects that no advisory will ever enumerate — many of which may never be fixed. The patch record sets a floor under the quality problem.

The flood of vulnerability disclosures signals a code problem. When an auditor walks into a firm that has not balanced its books in forty years, the resulting report is voluminous. The scandal is *not* the audit.

The cost of looking collapsed

Over the last few years, the price of discovery collapsed while the supply of defects kept growing.

The capability grew quickly. In November 2024, Google's Big Sleep agent produced what Google believed was the first public example of an AI agent finding an unknown exploitable memory-safety flaw in widely used software, a stack buffer underflow in SQLite. At the DARPA AI Cyber Challenge finals in August 2025, the seven autonomous finalist systems found, by DARPA's published accounting, 54 of 63 vulnerabilities planted in open-source infrastructure code and 18 real vulnerabilities that were not part of the Challenge. By mid-2026, Microsoft was running an agentic analysis system against the Windows networking and authentication stack; in May 2026, the company reported that the system found 16 of the bugs fixed in that month's release [9].

The tools also broke the field's longstanding triage heuristics. Anthropic's red team reported producing proof-of-concept code for 13 of 14 patched flaws that Microsoft rated "exploitation less likely" or "exploitation unlikely" — ratings calibrated to the effort profile of human attackers [10]. Proof-of-concept generation is not in-the-wild weaponization, but it is what the rating was designed to evaluate, and that rating no longer evaluates it correctly.

In 2014, Casey Ellis, founder of the bug-bounty platform Bugcrowd, proposed an amendment to Linus's Law, crediting Peter Gutmann for the underlying observation. He restated it in a July 2026 essay: "With many eyes and the right incentive, all bugs are shallow." The collapsing cost of discovery, he

argues, has given incentives "teeth on both sides" [11]. His framing doesn't go far enough. Incentives explain why the eyes arrived when they did, not why they found forty years' accumulated flaws. A census does not create a population.

One number offers temporary solace amid the flood. Of the 35,364 CVEs published in the first half of 2026, only 85 (0.24 percent) had reached CISA's Known Exploited Vulnerabilities catalog by July 1 of that year, and by design that catalog trails discovery [3]. For now, discovery appears to be outrunning confirmed exploitation by orders of magnitude. The field's current answer to that arithmetic is probabilistic triage. FIRST's Exploit Prediction Scoring System, first published in 2019 and issued daily since 2021, estimates the chance a given flaw will be exploited in the next thirty days, which suggests to a defender what to patch this week but nothing about the size of the backlog or the rate at which it grows.

One reading of these numbers is that discovery is up, patches are shipping, and confirmed exploitation remains rare. In that view, the systems are working as designed, and this essay has mistaken instrumentation for pathology. The refutation lies at the consumption end: the constraint has moved from finding flaws and shipping fixes to absorbing them. Verizon's 2026 Data Breach Investigations Report, drawing on more than 22,000 confirmed breaches, found exploited vulnerabilities to be the leading way in, displacing credential theft from the top spot for the first time in the report's nineteen years. That dataset closed at the end of October 2025, before the collapse described here, so it is a floor on the problem rather than a measurement of the present. A small exploited subset does most of the damage while defenders drown in the remainder. The crisis is accumulated engineering liability meeting cheap disclosure, a triage catastrophe before it is a compromise catastrophe.

A debt taken out on purpose

Ward Cunningham's original debt metaphor, introduced in a 1992 OOPSLA experience report, described an engineering choice: "Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite" [12]. That metaphor presumes intent to repay. What the AI tools are exposing is a different choice: Much of the industry borrowed with no repayment plan at all. Their wager was that discovery would remain expensive, that the collector would never arrive, and that the cost of default would fall on customers rather than on the balance sheet that reflected the loan. In a conversation with cybersecurity journalist Cynthia Brumfield, published in July 2026, I called this "willful debt" [13]. Some of the industry's arrears are ignorant debt, incurred by people who did not know better, but much of it was incurred by organizations that knew precisely what they were deferring, and said so in their own engineering retrospectives.

A primary exhibit is language choice. More than four decades after their role in failure modes was characterized, memory-unsafe languages are still the industry's default for systems code. Microsoft reported in 2019 that roughly 70 percent of the CVEs it assigned each year were memory-safety issues, and Google reported a similar fraction for serious Chromium bugs [14]. CISA urged vendors to move new development into memory-safe languages and to publish roadmaps for the code they already had [14]. In December 2025, MITRE's Top 25 listed three classic memory-safety weaknesses under their own identifiers: buffer copy without size checking (CWE-120) at eleven, stack-based buffer overflow (CWE-121) at fourteen, and heap-based buffer overflow (CWE-122) at sixteen. They surfaced there because MITRE stopped folding them into parent categories, not because anything changed in the code.

The composition of the flood and the composition of the damage differ. Counted by volume, the 2026 CVE corpus is dominated by web and access-control defects in the long tail of plugins and applications,

and memory-safety issues barely register in the top tier [3]. This essay concerns the second stream: by severity, and in vendor patch streams carrying kernel, driver, and protocol code, memory unsafety still dominates.

None of the most useful countermeasures is exotic or recent. Ada's 1983 standard required bounds checking; Modula-3 (1988), Java (1996), C# (2000), and Rust (1.0 in 2015) all enforce it, and static and dynamic analysis tools are commercial products refined over decades. The hazard was understood earlier still. Accepting the Turing Award in 1980, C. A. R. Hoare recalled offering to let customers switch off the subscript checking his Elliott ALGOL 60 compiler performed. They unanimously refused: they already knew how often such errors occurred, where failure to detect them could be disastrous. "I note with fear and horror that even in 1980, language designers and users have not learned this lesson," he said. "In any respectable branch of engineering, failure to observe such elementary precautions would have long been against the law" [15].

Research about memory safety, defensive programming, and program verification has been published and taught for decades. No vendor can claim that the warnings or the tools arrived only recently, although academia bears some blame. Much of that work appeared in venues industry did not read, and for most of those decades the standard systems curriculum included memory-unsafe languages as the default. Curricula answer to what employers hire for, and graduates who could not write C were graduates who could not get hired: a demand signal from the same firms now holding the debt. The standards bodies have a share too. The ISO C and C++ committees set the semantics that made the unsafe choices cheap, preserved undefined behavior as a performance affordance, and made safety opt-in: C11's bounds-checked interfaces arrived in a conditional annex, and within five years implementers reported to the committee that no viable conforming implementation had emerged, so they proposed removing it [16].

Safer practices lose, quarter after quarter, to arguments about schedule and feature count. That is what makes the debt willful rather than ignorant: The industry kept the statistics and kept shipping the defective products.

Microsoft's own case records hold 28 confirmed vulnerabilities spanning five years in a single driver, clfs.sys — a set against which it benchmarked its agentic scanner [9]. The Desktop Window Manager produced zero-days in consecutive months in early 2026, and the Remote Access Connection Manager was patched for exploited or likely-exploitable flaws three times between October 2025 and February 2026. In October 2025, attackers exploited a flaw in an Agere modem driver, decades-old third-party code still shipping in Windows, which Microsoft resolved by deleting rather than patching. Deletion was the correct call. It was also available at any point in the preceding twenty years.

Kev Breen of Immersive observed in December 2025 that of the five bugs Microsoft flagged that month as "more likely" to be exploited, most were in components either exploited before or documented enough by prior CVEs to make weaponization easier. Documented risk acceptance is a legitimate engineering instrument. A risk re-accepted in the same driver, year after year, through repeated exploitation, has stopped being acceptance and become policy.

Microsoft appears throughout this article not because it is the worst example but because it publishes more engineering data than its peers: exploitability ratings, memory-safety statistics, agentic-system results. Transparency draws attention, and vendors that disclose nothing are sometimes viewed as better. The recurrence record is nonetheless its own indictment.

Industry offers a defense: for most of the last few decades, deferring latent-flaw remediation was a sound expected-value decision. Discovery was rare and expensive, exploitation rarer still, and money

spent hardening code no one would probe was money not spent competing. Such risk acceptance is ordinary engineering economics, the argument runs, not negligence. More than one development manager has likely said, "I would rather have it wrong than have it late." Writing in this magazine in 2005, Eric Rescorla put the sharper version of the case: Finding and disclosing security holes had poor expected value, because the inventory was deep enough that removing any one flaw barely shifted an attacker's odds [17]. RAND put numbers on that equilibrium: from 207 zero-day vulnerabilities held by a private exploitation group between 2002 and 2016, an average life after discovery of 6.9 years and a median annual rediscovery rate of 5.76 percent [7]. That was the arithmetic of expensive discovery. Whether it still holds is what this essay turns on, though rediscovery resists easy reading: RAND's own rate ranges from under one percent to forty depending on the interval chosen, and private and public researchers hunt differently enough that overlap may reflect method as much as density.

But that defense carries an expiration date. Microsoft published the 70 percent memory-safety figure in 2019. CISA put vendors on written notice in 2023 [14]. The collapse in discovery costs has been visible in public milestones since 2024. Continuing the same practices past visible markers, on the same terms and in the same components, was no longer a bet made in ignorance. One official answer to this charge is CISA's Secure by Design pledge, which more than 250 manufacturers signed within a year of its May 2024 launch and 378 had signed by August 2026, committing to reduce entire vulnerability classes. The pledge set a one-year window and asked signatories to publicly document measurable progress. By August 2026, only 55 of them did. It set no baseline and defined no measure of completion, so each signatory grades itself. A commitment without a metric is a press release.

Stopping the borrowing, where tried, works: As Android shifted new native code to memory-safe languages, Google reported that the share of its vulnerabilities related to memory safety fell from 76 percent in 2019 to 24 percent in 2024 [8], and below 20 percent in 2025 [18]. The decline held in absolute counts as well as in proportion, even as the codebase grew [8]. Google puts memory-safety defect density in Android's Rust code about a thousand times below that of its C and C++, and reports that Rust changes spend 25 percent less time in code review and are rolled back about four times less often [18]. That is the clearest large-scale demonstration that changing behavior changes the outcomes.

The charges here are about the past; the test of reform is about the future. The warnings have been clear since about 1980, and most of the borrowing happened between the early 1990s and 2019. A vendor should not be judged by its 2026 patch count alone, which measures how much old debt is surfacing rather than how much new debt it is taking on. The harder question: Has new issuance stopped? Is current code being written in safe languages, specified, and reviewed? Android's numbers are such a measurement, which is why they matter more than any month's total.

Beneath the recurrence data across the field resides a point my co-authors and I have argued at book length. As Young, Boebert, and Kain observed in 1985, "A program that has not been specified cannot be incorrect; it can only be surprising" (cited in [19]). Most of what we call vulnerabilities are surprises in systems never specified against any security policy, produced by a penetrate-and-patch culture that treats the patch pipeline as a substitute for engineering. Ease of patching is not good security. It never was. Buyers rewarded whoever shipped first, and the producers obliged.

Where the debt meets physical systems

Some of the worst instances of this pattern run industrial processes. Operational technology (OT — the sensors, SCADA systems, controllers, and field devices behind power, water, and manufacturing) was

built on some of the same unsafe foundations as IT, with an additional rationalization: isolation. The air gap was the *claimed* security architecture. Then customers and vendors connected those systems to corporate networks and the Internet because remote monitoring and management were new features that helped reduce headcount, and they never priced in the redesign that safe connectivity demands [19].

OT amplifies the willful-debt problem in nearly every direction. Control-system assets often remain in service for decades. The code the new discovery tools are reading dates from an era with even fewer engineering controls than today's, written for protocols that assumed a trustworthy network because the network was a local wiring closet. No one can simply recompile that installed base into a memory-safe language, and operators cannot even confidently inventory much of it.

The penetrate-and-patch remedy fails outright here. Operators of continuous physical processes cannot absorb a 570-fix month. Downtime is one obvious reason, though not the only one. Most operators cannot apply an operating-system or firmware patch until the automation vendor validates it against the installed system, because an unvalidated patch voids support and, where a safety function is involved, invalidates the functional-safety assessment. That validation lags Patch Tuesday by weeks to quarters, and for some products it never arrives. The party controlling that queue is the same party that wrote the debt. Meanwhile, the operator still owes the paperwork. As one example, NERC CIP-007-6 requires bulk-power operators to evaluate available patches at least every 35 days and, within 35 days of an evaluation, either apply the patch or file a dated mitigation plan. Nothing in the rule requires that the patch ever be installed.

Frozen code does have one property that favors the defender: its defect population is finite. RAND made the point about industrial control systems directly, observing that because end-of-life codebases stay static, every flaw found reduces the total rather than sampling a moving target [7].

A vulnerability flood that is an expensive nuisance for IT is, for OT, a standing catalog of access paths to critical systems. Exploitation consequences are measured in pressure, temperature, and voltage. This debt is unpayable on patch terms. Addressing it requires redesign, compensating controls, enforced network segmentation rather than assumed segmentation, and retirement schedules that treat connectivity as a liability to justify, not a feature to presume. Stuxnet and the Volt Typhoon intrusions have shown these liabilities and consequences are not merely theoretical.

The co-signers

Vendors could not have carried this debt for so many decades without buy-in, and that financing came from customers. As Ed Felten famously noted years ago, "Given a choice between dancing pigs and security, users will pick dancing pigs every time."

The market has rarely discriminated on engineering quality, and never for long. Vendors keep their customers and their margins while shipping hundreds of patches a month for known defect classes that even their own publications call preventable. However, Microsoft's Trustworthy Computing initiative, launched after the malware embarrassments of the early 2000s, showed that a vendor can invest in quality when reputation demands it.

Purchasers reward the bright and the shiny: new features, new integrations, an AI assistant in every menu. Safety, simplicity, and engineering provenance are invisible at the point of sale. Feature lists appear in the advertisements, and defect recurrence rates appear nowhere. In most other engineered product categories, buyers and regulators long ago stopped tolerating this: The National Electrical Code and the fire codes that accompany it exist because purchase decisions alone proved insufficient. Software

already has versions of that. In avionics, automotive, and rail, certification regimes have long required traceable requirements and, at the higher integrity levels, enforced coding standards and systematic static analysis. The practices exist. Some of us demanded them of everyone else; the market was not listening.

Procurement makes the failure concrete. Selection criteria favor features, price, and delivery date. A vendor's patch history almost never appears as a line item, even though it is public, quantifiable, and a predictor of total cost of ownership. Internal practices, code provenance, and testing procedures are priced only when a compliance regime forces the question, and those regimes score process coverage rather than defect recurrence or language choice. Every organization that renewed a contract after the hundredth identical patch cycle underwrote the next round.

Many end-users never had a real voice in the purchasing decisions. Desktop operating systems, enterprise suites, and most OT verticals are markets of one or two practical choices, where switching costs are measured in years and retraining budgets, and where OEM bundling and compatibility constraints narrow the field further. Even so, decades of criticism have produced little market consequence for the unending stream of damaged goods.

Ross Anderson observed in 2001 that security markets resemble a market for "lemons": Buyers cannot observe quality before purchase, so quality is driven out of the market. A rural water utility running a 30-year-old controller did not finance the technical debt. It inherited the debt, bundled into its system acquisition. The co-signature belongs to buyers who had leverage: large enterprises and governments, which spend it on process compliance rather than engineering quality.

Open source has its own version of the technical debt co-signature. Unpaid maintainers pay interest on debts other people incurred, as the curl experience below illustrates. The co-signature belongs to consumers of open-source components, who assume someone else audits the code — the many-eyes belief my co-authors and I cataloged as a myth [19]. The eyes exist where the incentives align, which for open source means features and stability rather than security, as Ellis notes [11]. The 2020 FOSS Contributor Survey found respondents spent only 2.27 percent of their contribution time on security issues, with one calling the work a "soul-withering chore" [20].

Customer remediation behavior completes the account. During 2025, organizations fully remediated only 26 percent of the flaws Verizon's report counts as critical — those listed in CISA's Known Exploited Vulnerabilities catalog — where they turned up in their own environments, down from 38 percent a year earlier, while median time to patch rose from 32 to 43 days. The industry does not fully consume even the patches it gets. The debt is willful on all sides of the purchase order.

Paying interest without touching principal

The flood's costs are measurable, and they are being paid in the wrong currency: labor and readiness rather than engineering.

Triage labor came first. In January 2026, Daniel Stenberg shut down curl's bug bounty. Confirmed vulnerabilities had run above 15 percent of submissions in earlier years. From the start of 2025, that rate fell below 5 percent, and the maintainers' scarcest resource, attention, was going to disproving fabrications. Stenberg had named the experience "death by a thousand slops" six months before he closed the program. AI-assisted analysis by skilled researchers also drove real curl fixes and CVEs during the same period. The technology did not choose between those outcomes. The incentive structures around it did.

A reader could dismiss that as the predicament of an underfunded volunteer project, except the world's most valuable company hit the same wall. By mid-2026, Apple had written the experience into its bounty rules: reports of issues "discovered by AI without proper validation" are ineligible, researchers are told to avoid "lengthy descriptions generated by AI tools," and repeat submitters can have their reports paused for 180 days or be removed from the program. Apple's stated reason is this essay's argument in a vendor's own words. Investigating such reports, it warns, "can prevent us from acting quickly to resolve serious and critical security issues."

NVD's April 2026 retreat to enriching only prioritized CVEs leaves, by a Cloud Security Alliance estimate made shortly afterward, 80 percent or more of the flow without authoritative analysis. HackerOne paused its Internet Bug Bounty program in March 2026 as remediation capacity fell behind discovery. Jerry Gamblin and Éireann Leverett, writing for FIRST's vulnerability forecasting effort, stated the constraint in their mid-2026 forecast commentary: "the constraint is no longer discovery; it is the human capacity to verify, coordinate, and patch" [21]. The databases, scoring systems, remediation mechanisms, and advisory pipelines the field built for a few thousand disclosures a year embodied institutional assumptions about the cost of discovery. They are failing together now that the assumptions are proving false.

Readiness costs are surfacing in places that measure them professionally. Federal News Network reported in mid-July 2026 that rolling cybersecurity quarantines had locked U.S. Air Force and Space Force personnel out of their machines for hours to days [22]. As patch tempo rose, devices that missed an update window were automatically cut off under the Comply-to-Connect program. An unofficial account put the affected devices in the tens of thousands, although the Air Force declined to confirm a number, citing operational security. Cutting off an unpatched endpoint is the designed behavior of a posture gate, and the alternative is worse. What failed was the sizing: An update window calibrated to a previous patch rate, with no graceful-degradation path when the rate tripled. The patch flood directly translates into lost mission time in service, paid for by end users who never had a vote in the engineering that made the tempo necessary.

For some end users, the flood is existential. Ransomware crews exploit the same flaws cataloged in the advisories and were involved in 48 percent of the breaches in Verizon's 2026 report. A Fortune 500 firm absorbs a remediation bill, but a school district, a county water utility, or a small college may not. Lincoln College in Illinois closed permanently in 2022, with a ransomware attack among the proximate causes. The patch flood is priced in currencies that include continuity of existence.

Meanwhile, the same discovery windfall serves attackers, whose incentive structure, as Ellis observes, pays them to find flaws and pays them more for keeping them alive. Defenders carry backlogs that attackers never have to. The 0.24 percent exploitation figure cited earlier is a snapshot of an early moment. Every unremediated finding in the public record is prospecting data for someone whose cost of exploitation is also falling.

Retiring the principal

If the debt is willful, the remedies are acts of will. Three follow directly, and a fourth is coming whether the industry wants it or not.

Practice and training before features. One common vendor response to AI has been to use it to emit more code and add more features. That response expands the attack surface by introducing machine-generated code of often questionable quality. It is new debt issuance at machine speed, on the same terms

as the old. The productive response is not about output at all: Adopt memory-safe languages for new components, require specification and review for security-critical code, and train developers in sound construction rather than prompt-driven assembly. When a defect ships anyway, the most productive question is what allowed it: the training, the specifications, the libraries, the schedule, or the missing verification steps.

People matter more than tooling. Producing safe software for critical use requires trained, experienced engineers in the loop. Staffing production with people who assemble model output they cannot evaluate will compound the problem. The correction underway is from replacing people with machines to using machine breadth under human supervision. Engineering discipline is a workforce property before it is a toolchain property, and too many universities and employers have spent recent years cutting the structures that build it.

None of this is a new prescription. It is a standing one, available for decades and overruled every quarter by the schedule. AI does not change what sound engineering is. It changes the cost of skipping it, by guaranteeing the skipped work is found.

Refactor toward resilience instead of chasing patches. The same AI tools flooding the CVE pipeline can be pointed at retiring the debt rather than itemizing it. AI-assisted translation of security-critical components into memory-safe languages, machine-generated test suites for legacy code that never had them, and refactoring toward explicit specifications are all plausible at scale for the first time. DARPA's TRACTOR program has pursued automated C-to-Rust translation on this premise since 2024. Costs have fallen far enough to remove the last practical excuse for not pointing these tools at legacy code.

This deserves critical evaluation. Verifying a machine-rewritten component is an open problem, and a bad rewrite trades one liability for another. Equivalence checking, differential testing, and staged deployment are necessary companions. The question for any such effort is whether the rewritten component carries a specification and evidence the original never had. Otherwise the exercise has translated the surprise into a newer language. But the strategic distinction matters. A vendor using AI to find-and-patch is mucking out a stable while the horses generate more. A vendor using AI to rewrite the code is reducing the herd. Findings per month should be read as a burn-down chart. Customers repeat the vendors' preference for reaction over repair: IBM's 2026 Cost of a Data Breach report found that half of breached organizations had deployed AI agents for threat detection and containment, while only 18 percent had applied them to vulnerability management [23].

Buy like an actuary. No rating agency downgrades a vendor for shipping defect classes it has known about for years, so the buyer is the only rating mechanism the engineering ledger has. The largest buyer in the world held that lever and let it go: in January 2026 the Office of Management and Budget rescinded the requirement that agencies obtain a secure-development attestation before deploying software, returning the question to each agency's own risk assessment [24].

Patch-rate history is public data. Procurement should read a vendor's defect record the way an underwriter reads loss experience, with one correction made earlier in this essay: A raw 2026 patch count measures how hard a vendor is looking, not how much new debt it is taking on. Signals that survive that objection include new-code language policy, software engineering rigor, recurrence of the same defect class in the same component, and elapsed time from report to fix. Buyers should also ask for memory-safety roadmaps by name, because CISA, NSA, the FBI, and their Australian, British, Canadian, and New Zealand counterparts jointly made the case for them in 2023 [25]. The co-sealed request matters: It gives

buyers in several countries an authority to cite, and it does not evaporate when one government changes its policy.

For essential and important entities under the EU's NIS2, now written into national law in all but a few Member States, supplier security is not discretionary. The directive requires security in the acquisition, development, and maintenance of systems, including vulnerability handling; extends the duty to direct suppliers; and makes management bodies approve and oversee those measures, on pain of personal liability. What to ask a vendor for is still the buyer's judgment. That the buyer must ask is not.

For OT, the bar is higher, and the instruments already exist. The IEC 62443 series covers secure development lifecycle, component security, and the zone-and-conduit assessment that yields the segmentation described earlier. Outside a few regulated buyers, certification against it is seldom demanded. Europe is about to narrow that discretion: the EU machinery regulation applies from January 2027 and writes protection of safety-related control systems against intentional corruption into the requirements a product must satisfy to be sold.

The familiar reflex of buying more AI-branded defensive products to stand in front of unspecified, unmaintained code will not work. No product alone makes an organization secure, and no product ever has.

Any metric used for decisions invites gaming. Campbell's Law, stated for social indicators in 1976, holds that the more an indicator drives decisions, the more it will be corrupted and the more it will distort what it was meant to monitor. CVE counts are already imperfect, as was detailed earlier. If patch counts drive purchases, some vendors will sit on reports, dispute findings, or refuse identifiers to manage the numbers. Pair the defect record with signals that are harder to game: exploited-in-the-wild history, time from report to fix, and code-level assessment by an independent party. Note what is not available. A vendor's insurance claims experience is confidential, and SOC 2 Type II and ISO 27001 certify process rather than memory safety or defect recurrence.

The disclaimer will not hold forever

The fourth pressure does not depend on anyone's good intentions. In most other engineering practices, knowingly building with substandard materials invites lawsuits and sometimes prosecution. To date, the computing industry has avoided this through a combination of license exclusions and sustained, successful lobbying against the creation of any such regime. Litigation over defective software has been rare because vendors sell under agreements that disclaim nearly all liability, and courts have mostly let those terms stand. That description applies to the United States. European product liability law goes further than limiting such disclaimers: It forbids contracting out of liability within its scope, in terms binding on national law as well as on contracts [26]. Even in the US, the 2023 National Cybersecurity Strategy proposed shifting liability toward software vendors, though *President Trump's Cyber Strategy for America*, which replaced it in March 2026, dropped that agenda. Legislators have been heavily lobbied to avoid broader regulation with the claim that it would hurt competition and innovation. That claim is not empty, but it has also never been weighed against the costs of the status quo. Aviation, automobiles, and telecommunications carry far heavier rules and remain innovative and profitable.

That is unlikely to last. The evidence available to plaintiffs and regulators is different now: published vendor statistics acknowledging dominant defect classes, government guidance urging specific remedies by name, decades of academic instruction in preventive methods, and recurrence records showing the

same component patched for the same classes of flaws across years. Choices documented that thoroughly are difficult to defend as unforeseeable accidents.

Under the Restatement (Second) of Contracts §195, a term exempting a party from tort liability for harm caused intentionally or recklessly is unenforceable on public policy grounds. *Henningsen v. Bloomfield Motors* [27], the 1960 New Jersey decision that dismantled an automaker's warranty disclaimer, asked a question that reads well against a modern computing license agreement: Where can the buyer go to negotiate for better protection? Software vendors have so far fared well against such arguments — in *M.A. Mortenson v. Timberline Software* [28], the Washington Supreme Court enforced a limitation-of-remedies clause even though the vendor's own internal memorandum documenting the bug predated the sale. The disclaimer is only a secondary barrier. A rule against recovering purely economic losses in tort usually stops the claim before anyone reaches the contract. Europe lands in the same place, because its product liability regime also excludes pure economic loss.

The 2023 National Cybersecurity Strategy conceded the state of play, observing that software makers can "leverage their market position to fully disclaim liability by contract." That is an admission that the common law has not done this work, and an indication that something else must. Insurance is not a reliable backstop. Liability insurance covers negligence, and it usually excludes losses the insured expected or intended. Conduct a plaintiff would characterize as reckless is close enough to that exclusion that insurers will litigate it.

Some regulators outside the United States are not waiting. The EU's Cyber Resilience Act imposes secure-development and vulnerability-handling obligations on products sold in Europe; its reporting obligations began in September 2026, and the main obligations take effect in December 2027. Article 13 requires a declared support period of at least five years unless the product's expected use is shorter, security updates throughout it, and the end date stated at the point of sale.

Europe has gone further and legislated liability itself: The product liability directive agreed in 2024 brings software into the Union's product liability regime, and as Ruohonen and Kang observed in these pages, it narrows the gap between security and safety by codifying cyber insecurities that lead to safety incidents [26]. Two qualifications keep that from being the arrival of software liability. The directive applies only to products placed on the market from December 2026 onward, which means it does not cover the installed base this essay examines. It also compensates natural persons. It covers death and injury, damage to property not used exclusively for professional purposes, and corruption of data not used professionally, but not pure economic loss. An enterprise reading this will find no route to recover its breach losses under the directive, though it will find fresh exposure there if it also sells software.

In another respect it reaches further than it first appears. It is a no-fault regime and not a negligence one. A claimant must still establish defectiveness, damage, and causation, which Ruohonen and Kang judge difficult for vulnerabilities, but Article 10 supplies rebuttable presumptions. One of them presumes defectiveness where a product fails a mandatory Union safety requirement, and from December 2027 that will include the Cyber Resilience Act's essential cybersecurity requirements. Whatever its execution, it exists because purchase decisions alone were insufficient.

Lobbying against creating a liability regime has worked in the past. Given the mounting losses and compromises, that is unlikely to keep working. Vendors would do well to get ahead of the change.

Conclusion

The auditor did not cook these books. The audit found decades of deferred maintenance carried on the ledger as savings, approved at every renewal by the customers underwriting it. Vendors chose the debt; the buyers with leverage financed it; both can stop. If they do not, the fourth pressure will not wait: The disclaimers that made the deferral affordable are beginning to be written out of European law for the people the directive protects; other jurisdictions may be expected to follow. The evidence that would support a liability claim accumulates monthly in the vendors' own advisories. The AI systems presenting the bill will serve either purpose, padding the feature list or rebuilding the foundations. For the first time in the field's history, rebuilding the foundations is cheap enough to take seriously.

References

[1] B. Krebs, "Microsoft Patches a Record 570 Security Flaws," Krebs on Security, Jul. 14, 2026. [Online]. Available: https://krebsonsecurity.com/2026/07/microsoft-patches-a-record-570-security-flaws/

[2] P. Davuluri, "Evolving Windows Vulnerability Management to Meet the Speed of AI-Powered Discovery," Windows Experience Blog, Jul. 9, 2026. [Online]. Available: https://blogs.windows.com/windowsexperience/2026/07/09/evolving-windows-vulnerability-management-to-meet-the-speed-of-ai-powered-discovery/

[3] J. Gamblin, "CVE Mid-Year 2026 Check-In: Volume Vertical, Exploitation Rare," JerryGamblin.com, Jul. 1, 2026. [Online]. Available: https://jerrygamblin.com/2026/07/01/3528/

[4] National Institute of Standards and Technology, "NIST Updates NVD Operations to Address Record CVE Growth," Apr. 15, 2026. [Online]. Available: https://www.nist.gov/news-events/news/2026/04/nist-updates-nvd-operations-address-record-cve-growth

[5] Anthropic Frontier Red Team, "Assessing Claude Mythos Preview's Cybersecurity Capabilities," Apr. 7, 2026. [Online]. Available: https://www.anthropic.com/research/mythos-preview

[6] N. Alexopoulos et al., "How Long Do Vulnerabilities Live in the Code? A Large-Scale Empirical Measurement Study on FOSS Vulnerability Lifetimes," in *Proc. 31st USENIX Security Symp.*, Boston, MA, USA, Aug. 2022, pp. 359–376.

[7] L. Ablon and A. Bogart, *Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits*, RR-1751-RC. Santa Monica, CA, USA: RAND Corporation, 2017, doi: 10.7249/RR1751.

[8] J. Vander Stoep and A. Rebert, "Eliminating Memory Safety Vulnerabilities at the Source," Google Online Security Blog, Sep. 25, 2024. [Online]. Available: https://security.googleblog.com/2024/09/eliminating-memory-safety-vulnerabilities-Android.html

[9] T. Kim, "Defense at AI Speed: Microsoft's New Multi-Model Agentic Security System Tops Leading Industry Benchmark," Microsoft Security Blog, May 12, 2026. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2026/05/12/defense-at-ai-speed-microsofts-new-multi-model-agentic-security-system-tops-leading-industry-benchmark/

[10] Anthropic Frontier Red Team, "Measuring LLMs' Impact on N-Day Exploits," Jun. 8, 2026. [Online]. Available: https://www.anthropic.com/research/n-days

[11] C. Ellis, "The Amended Linus's Law," cje.io, Jul. 13, 2026. [Online]. Available: https://cje.io/2026/07/13/the-amended-linuss-law/

[12] W. Cunningham, "The WyCash Portfolio Management System," in *Addendum to the Proc. OOPSLA '92*, Vancouver, BC, Canada, Oct. 1992, pp. 29–30, doi: 10.1145/157709.157715.

[13] C. Brumfield, "AI Won't Save You From Weak Cybersecurity Fundamentals," *The NIST 2.0 Cybersecurity Framework* (cyberriskbook.com), Jul. 13, 2026. [Online]. Available: https://cyberriskbook.com/ai-wont-save-you-from-weak-cybersecurity-fundamentals/

[14] B. Lord, "The Urgent Need for Memory Safety in Software Products," CISA, Sep. 20, 2023 (rev. Dec. 6, 2023). [Online]. Available: https://www.cisa.gov/news-events/news/urgent-need-memory-safety-software-products

[15] C. A. R. Hoare, "The Emperor's Old Clothes," *Commun. ACM*, vol. 24, no. 2, pp. 75–83, Feb. 1981, doi: 10.1145/358549.358561.

[16] C. O'Donell and M. Sebor, "Field Experience With Annex K — Bounds Checking Interfaces," ISO/IEC JTC1/SC22/WG14 N1967, Sep. 25, 2015. [Online]. Available: https://www.open-std.org/jtc1/sc22/wg14/www/docs/n1967.htm

[17] E. Rescorla, "Is Finding Security Holes a Good Idea?" *IEEE Security & Privacy*, vol. 3, no. 1, pp. 14–19, Jan.–Feb. 2005, doi: 10.1109/MSP.2005.17.

[18] J. Vander Stoep, "Rust in Android: Move Fast and Fix Things," Google Online Security Blog, Nov. 13, 2025. [Online]. Available: https://security.googleblog.com/2025/11/rust-in-android-move-fast-fix-things.html

[19] G. Spafford, L. Metcalf, and J. Dykstra, *Cybersecurity Myths and Misconceptions: Avoiding the Hazards and Pitfalls that Derail Us*. Boston, MA, USA: Addison-Wesley, 2023.

[20] F. Nagle, D. A. Wheeler, H. Lifshitz-Assaf, H. Ham, and J. L. Hoffman, *Report on the 2020 FOSS Contributor Survey*. San Francisco, CA, USA: The Linux Foundation and The Laboratory for Innovation Science at Harvard, Dec. 2020. [Online]. Available: https://www.linuxfoundation.org/hubfs/2020FOSSContributorSurveyReport_121020.pdf

[21] J. Gamblin and É. Leverett, "The 2026 Vulnerability Forecast Update: Navigating the AI Epoch," FIRST Blog, Jun. 15, 2026. [Online]. Available: https://www.first.org/blog/20260615-vulnerability-forecast-update

[22] R. S. Cohen, "Air Force, Space Force Network Lockouts Hit Troops and Civilians," Federal News Network, Jul. 14, 2026. [Online]. Available: https://federalnewsnetwork.com/defense-main/2026/07/air-force-network-lockouts-hit-troops-and-civilians/

[23] IBM, *Cost of a Data Breach Report 2026*. Armonk, NY, USA: IBM Corp., Jul. 2026. [Online]. Available: https://www.ibm.com/reports/data-breach

[24] Office of Management and Budget, *Adopting a Risk-Based Approach to Software and Hardware Security*, Memorandum M-26-05, Jan. 23, 2026. [Online]. Available: https://www.whitehouse.gov/wp-content/uploads/2026/01/M-26-05-Adopting-a-Risk-based-Approach-to-Software-and-Hardware-Security.pdf

[25] Cybersecurity and Infrastructure Security Agency, National Security Agency, Federal Bureau of Investigation, Australian Signals Directorate's ACSC, Canadian Centre for Cyber Security, UK National Cyber Security Centre, New Zealand NCSC, and CERT NZ, *The Case for Memory Safe Roadmaps*, Dec. 2023. [Online]. Available: https://www.cisa.gov/resources-tools/resources/case-memory-safe-roadmaps

[26] J. Ruohonen and E.-Y. Kang, "Software Liability Came to Europe," *IEEE Security & Privacy*, vol. 24, no. 1, pp. 61–64, Jan.–Feb. 2026, doi: 10.1109/MSEC.2025.3642586.

[27] *Henningsen v. Bloomfield Motors, Inc.*, 32 N.J. 358, 161 A.2d 69 (1960).

[28] *M.A. Mortenson Co. v. Timberline Software Corp.*, 140 Wn.2d 568, 998 P.2d 305 (2000).

[29] J. Gamblin, "2025 CVE Data Review," jerrygamblin.com, Jan. 1, 2026. [Online]. Available: https://www.jerrygamblin.com/2026/01/01/2025-cve-data-review/

Eugene H. Spafford is a Distinguished Professor of computer science at Purdue University and the founder and executive director emeritus of Purdue's CERIAS. His research interests include software assurance, security engineering, and cybersecurity policy. Spafford received his Ph.D. in information and computer sciences from the Georgia Institute of Technology. He is a Fellow of the ACM and AAAS, a Life Fellow of IEEE, and a member of the Cyber Security Hall of Fame. Contact him at spaf@purdue.edu.

Portions of this text were drafted and structured with the assistance of Anthropic Claude Opus 5; the core ideas, arguments, and final editorial decisions — and especially the em-dashes and semicolons — are the author's.