Security Now! #1103 - 11-03-26

Exposing Bad Consumer Crypto

This week on Security Now!

• Cisco gives a 27-year old the task of recoding 98% of their software with AI. • A serious but not earth shaking vulnerability found in PXE booting. • The largest ($6.3 BILLION) financial heist in history (with crypto). • 710 Polymarket gambling wallets emptied of $2.7 million. • A new robots.txt control for AI bots, and agreement to obey it. • Google generates haiku from the first orbital AI. • From our listeners: Does stealth updating make sense, how to move web servers off the Internet, books don't contain knowledge, a small company attacked by powerful AI, how would an "AI kill switch" be implemented.
• Another new and worrisome application of reverse-engineering AI.

We think we know what they meant...

Actually seen in Polk County, Iowa

Security News

Cisco's legacy code rewrite

Tom's Hardware picked up on a story reported by Fortune magazine. Fortune's overly long headline for the story they ran Friday was *"Cisco chief found a 27-year-old ex-founder 'buried away in the organization doing something unstrategic'—now the Gen Zer is rewriting 98% of Cisco's code"* Since this will be an AI-driven "vibe coded" project this naturally begs the question *"What could possibly go wrong?"* Here's a start of Fortune's reporting, which reads:

> *Gen Z can't catch a break. Entry-level jobs are vanishing; some 40% of bosses say they plan to hire fewer grads because AI can do the job cheaper, and young workers are routinely slammed for showing up late and refusing to put in overtime for free. Cisco's president and chief product officer Jeetu Patel isn't buying it—he's just handed one of the company's biggest jobs to a 27-year-old he found hidden in the org chart "five levels" below him.*
>
> *Patel told Fortune at a small roundtable dinner: "I've given him unlimited tokens because he's going to refactor code. So I'm like, you can go, you can go nuts."*
>
> *The "him" is Padmanabhan Krishnamurthy, or "Paddy," as Patel affectionately calls him. He joined Cisco in 2025 after the $429 billion tech giant acquired EzDubs, the real-time translation startup Krishnamurthy cofounded in 2023. And despite coming in as a former founder with CEO experience, he was, in Patel's words, "buried away in the organization doing something unstrategic."*
>
> *That changed when Patel dropped into Cisco's San Francisco office, and the pair got talking. Krishnamurthy reckoned he could rewrite Cisco's legacy code with a two-person team. Patel says even the most AI-forward people on his own team weren't thinking that way. So Paddy has given himself four weeks to prove it, with the final goal of rewriting 98% of Cisco's legacy code within 24 months.*
>
> *Now, Patel is on the hunt for more Paddys: young hires who will "challenge the system and the status quo," or, as he puts it, be "an energizer bunny that just goes out and kind of revs everyone up." At most companies, only someone with decades of experience under their belt would be trusted with a project this size. Patel thinks that's a problem. He said: "If you don't hire the early-in-career [workers], and you just let your workforce continue the way that they were, where do fresh ideas get injected? How do you challenge the status quo?"*
>
> *Seasoned staff can still come up with new ideas, Patel added. What they can't do is ask the questions that only someone with no experience would think to ask. And getting the two to get along isn't always smooth. Patel admits that putting a Gen Zer in a room with people who've been at Cisco for decades can create friction—but he says that's the point. "Tension is good," he said. "I actually try to create tension in the teams because you want to have enough friction on the ideas. You don't want to have friction with people and personalities. What you have to do is establish enough trust with people to say you are going to be expected to create unreasonable goals for yourself. And so, in order to go out and achieve those unreasonable goals, think outside of the box." And he's not just pushing senior staff to take notes from twentysomethings. He said that he treats every intern as "kind of a reverse mentor."*

All I can say in reaction to this from the outside is that this should be interesting. I'm sure many of our listeners have been in large, lumbering and management-layered enterprises. And that the idea of some upstart five layers down having a direct conversation with the CEO and

somehow enchanting him or her with their guile and claims of being able to wield the power of agentic AI rewrite 98% of the organization code, would not go down well. This is not to say that shaking up the complacent old guard might not be useful and needed. But I'm not sure I'd want to be anywhere near the dynamics that could be expected to ensue.

But more than the messy politics of that move, I'm unsure that Cisco's software is their biggest problem. I'm sure that anyone who's not a would-be bad guy loves the idea of Cisco's networking products becoming better and more secure. But we know that not all of their problems arise from bugs. We celebrated their announced and planned about-face on security **policy** because policy has always arguably been the biggest problem Cisco has faced and continues to face. It doesn't matter when a web portal's authentication might have an unknown bug if that web portal is never needlessly exposed to the public Internet. But Cisco's policies have continued to assume that their software is flawless despite history repeatedly showing that it's far from that. Rather than reducing their effective feature-set by prioritizing security over features and convenience, Cisco keeps playing catch-up, patching flaws **after** they have been successfully exploited by bad guys, thus continually placing their customers at risk.

Sure, it's possible to deliberately configure Cisco's products to be secure. We've looked closely at that in the past. And to help with that — and apparently to help them disclaim responsibility when things go horribly wrong — they provide an optional "Hardening Guide" that's full of extra steps that can be taken to override Cisco's "insecure by default" configurations. But why not just make their system secure by default? That is what that guy whom we quoted last year promised. It sounded as though he really "got it". But there's been little sign of that effort since.

In any event, it's going to be interesting to see what comes of this two-year *"rewrite everything with AI"* effort. As I said, I'm not sure that's the problem I would focus upon if I were Cisco, but we'll see how that goes.

The UEFI PXE flaw

Another vulnerability has been discovered — yes, thanks to AI — that was also under embargo pending a coordinated disclosure originally set for January. But once again the cat got out of the bag. Fortunately, the impact of this one is very different from a public Internet PHP nightmare. But it may be important depending upon your area of responsibility. The trouble resides in the PXE boot-time code of motherboard UEFI firmware.

Dan Goodin covered this for ArsTechnica with the headline: *"Code-execution flaw in UEFI firmware affects most servers on planet Earth"*. While that's strictly true, it's riding a bit more on the coattails of the recent PHP mess. This isn't on the same scale. Dan's piece begins with:

> *Defenders across the world face the burden of responding to yet another botched code-execution vulnerability, this time in boot-time firmware used by most of the world's server vendors and data centers.*

I assume what Dan meant by writing "another botched code-execution vulnerability" was the premature (and thus "botched") disclosure of another code-execution vulnerability. In any event, here's what he reported:

The vulnerability, with a severity rating of 9.5 out of 10, resides in the TianoCore EDK II, an open source codebase that hardware makers and cloud providers use to create the UEFI firmware that runs during the boot-up sequence. Last year, a representative of the UEFI Forum called EDK2 the "preeminent solution / implementation of the UEFI specifications." Virtually all UEFI suppliers (sometimes still called IBVs or independent BIOS vendors), including AMI, Insyde and Phoenix, use the reference code extensively. Project Mu, Microsoft's hardened EDK2 fork, used the code containing the vulnerable component as well.

The vulnerability, tracked as CVE-2026-97631, is a memory-corruption flaw in an ICMP component that attackers can exploit to execute malicious code that runs before the operating system and firmware can implement defenses like Secure Boot, which establishes a chain of trust for all code, including the Trusted Platform Module (TPM), loaded during boot. While payloads delivered by CVE-2026-97631 have no persistence—meaning they run only once — they can give attackers an opening to install other payloads that also run each time early in the boot process.

The parties that discovered CVE-2026-97631—Alex Matrosov of Anthropic and Dan Holmes-Mitra of Google—had arranged for it to stay under wraps until its coordinated disclosure that was scheduled for January. AMI, a maker of motherboards and other hardware, published this pull request last month that detailed the vulnerability and included a code fix. Once the post came to the attention of the discovering parties, TianoCore Project members pushed the disclosure forward, publishing a public advisory last week.

With publicly available patch code, it will now be trivial for advanced hackers to learn how the vulnerability is triggered and to build working exploits.

Okay. While that's true, as we'll see, obtaining the required access and opportunity will likely be anything but trivial. But let me give Dan some more time to explain. He continues:

Further compounding the urgency, patching UEFI code vulnerabilities in millions of end devices sold and used by thousands of hardware makers and cloud providers is cumbersome and time-consuming. Word of the vulnerability comes six days after another botched disclosure, this one involving a critical code-execution vulnerability in the PHP programming language.

HD Moore, creator of Metasploit and chief research officer at security firm Dragos said: "This vulnerability in the ICMP parser of the EDK2 network stack is interesting in terms of how it affects the chain of trust and the broader ecosystem. An attacker intent on compromising a nearby endpoint or server could use this bug to hijack program flow at a point where the boot process still has access to sensitive components, like the endpoint's TPM. This could be used to break disk encryption for cases where an attacker has short-term network access (and it would be even easier in a theft scenario)."

Again, while that's strictly true, there are many things that server operators can do to zero the risk. But that said, it may indeed be necessary for steps to be taken. Dan continues:

CVE-2026-97631 is a 16-year-old buffer overflow in the PXE-Base Code driver responsible for allowing servers, storage equipment, and other devices to boot using code downloaded on-the-fly over a local network. During this process—known as the pre-execution environment boot—the driver uses a network listener to monitor for ICMP errors.

Because the driver does not enforce a memory copy limit of 504 bytes, thus preventing the overwriting of adjacent data, a malicious insider or a hacker who has gained low-level access to a local network can send an error message that appends code past this limit. The IUEFI module running on the CPU then becomes corrupted.

Again, I'll highlight *"a hacker who has gained low-level access to a local network"* And add to that *"at the moment during which the server being targeted is rebooting over the network using PXE."* So the vulnerability is specifically a local network PXE boot-time vulnerability. Again, while this is certainly not nothing, it's not a PHP-scale earth shaker. Dan's reporting continues:

As the official advisory puts it:

During network boot, the PXE Base Code driver keeps a receive request open for ICMP error messages and copies each received message into the fixed 504-byte field Mode.IcmpError of its private context structure. The copy loop's bounds handling is incorrect in both the IPv4 and IPv6 handlers: for a message longer than 504 bytes it still copies bytes into the field, and it advances the destination pointer by the cumulative rather than the per-fragment length, so an oversized message writes attacker-controlled bytes past the field. Nothing earlier on the receive path restricts the message length.

So we have a classic textbook buffer overflow. The official advisory continues:

The malicious error can be delivered through a simple unicast sent from any network distance to a target device's IP address, as long as it's sent during the PXE boot sequence. An attacker can deliver a denial-of-service by causing the boot sequence to hang. When the attacker knows what the target firmware build is, it can plant a "counterfeit event" that executes the malicious code on the CPU in its UEFI firmware environment. The code runs in ring 0, a CPU's most privileged execution level. Because exploits bypass attestation processes designed to cryptographically prove to remote servers that the hardware is secure, they present a key opportunity for elevating limited privileges.

The CVE joins a growing list of previously discovered vulnerabilities in UEFI or code that works closely with it. The most severe of them, LogoFail, stemmed from logo images included in the UEFI of just about every hardware maker. Rendering bugs allowed attackers to subvert the chain of trust UEFI enforces and go on to install malicious components before the OS loaded or Secure Boot ran. PixieFail was a vulnerability similar to CVE-2026-97631. It, too, was the result of faulty PXE boot flaws in EDK2. Following discovery of the vulnerability, researchers embarked on multiple fuzzing projects designed to surface other critical flaws in the open source code library. It appears those efforts failed to find this latest flaw.

Other recently discovered UEFI flaws include: PKFail, a supply-chain failure resulting from non-production test platform keys, public for more than a decade before they were discovered, used in hundreds of device models. Then there's BootHole, Bootkit and BlackLotus.

Like CVE-2026-97631, exploitation or use of these payloads is appealing because it allows attackers to infect machines early in the boot process, before TPM and other defenses load, so they can inflict maximum damage. Given the ubiquity of CVE-2026-97631, Moore said it's likely a "forever" bug that will live on in end-of-support systems.

There are important caveats and limitations to exploiting this latest vulnerability. As noted, some level of control over a local network is a prerequisite. That likely limits exploitation to insiders or attackers, such as nation-states or ransomware operators, who manage to gain low-level access inside a network and are looking for a way to elevate system rights. There's no evidence yet that CVE-2026-97631 provides an initial entry point into a network. There are other hurdles to clear.

Jesse Michael, a principal researcher at firmware security firm Eclypsium, told Ars: "For exploiting the vulnerability itself, an attacker needs to guess specific addresses within the target system, so it's not something where you can throw a single exploit at any target and get RCE across a diverse range of devices. It needs to be tuned for each target model/version/configuration since the addresses will change based on those. Unlike some of the previous, high-profile vulnerabilities like BootHole, Black Lotus, and Baton Drop, this vulnerability can be fixed with a basic code fix, which will require deploying a BIOS/UEFI update, but they don't need to deal with revocations, pk/kek/db/dbx key management updates, or making sure to update OS and provisioning media like you do with Secure Boot bypasses in the bootloader itself."

CVE-2026-97631 doesn't affect home and small office users, or any other users who boot their devices solely from disk drives. Instead, it applies only to organizations with machines that boot from local networks, or conceivably from Internet-connected resources. Even then, the targeted firmware must rely on EDK2, as opposed to a different reference implementation, and use the PXE functionality (which is turned on by default).

In an email, Intel said that none of its "commercial products currently under security support" are known to be affected. "Intel does not include the affected EDK2 functionality in its reference code, though it may be integrated by an OEM board vendor or system manufacturer," the company added. It advised customers to check with their system manufacturer for specific guidance. Publicly available Intel reference code, however, makes reference to EDK2. Further, TianoCore has said that each Intel-supplied UEFI development kit "is a fully validated implementation… for a specific revision of the EDK II source." Many other Intel documents reference EDK2 as well.

ARM, meanwhile, responded to a list of questions with: "CVE-2026-97631 is not directly applicable to Arm's product portfolio." Like Intel, firmware the company makes available also borrows heavily from EDK2. TianoCore has also documented ARM's reliance on EDK2.

While it's possible some platforms are free of EDK2 code, it's prudent for users of all servers to investigate whether CVE-2026-97631 affects them.

Questions sent to AMD, Phoenix Technologies, AMI, Microsoft, and Amazon—likely some of the other biggest organizations affected—went unanswered when this post went live. Without concrete answers to questions about the use of EDK2 by organizations like these, it's impossible to say what or how devices are affected and what actions, if any, these organizations, or their users, should take. Representatives from the TianoCore Project also did not respond.

Some readers are likely to be unhappy with the lack of detail. Sometimes—particularly in cases involving botched disclosures that break embargoes months before they're scheduled to lift—there's no escaping the lack of actionable information in initial reporting.

Those affected should check with the maker of the firmware being used to boot their systems and ask for guidance. Beyond that, they should consider turning off network boot capabilities at least temporarily if possible. Firmware configurations should always enable security features such as TPM and Secure Boot whenever possible, although those defenses don't protect against this particular threat. It's also a good idea to monitor logs for signs of exploitation attempts.

Motherboard firmware tends to enable PXE booting by default. Anyone who has worked with a system that, for whatever reason, fails to boot from its own local media and then falls back to PXE knows how annoying (and patient) PXE can be. Since PXE is always enabled but not always wanted or needed, the easiest mitigation will be to simply disable it in the firmware. Another solution if PXE must be enabled would be to filter incoming ICMP error messages to any booting hosts when possible. The premature disclosure caught the motherboard firmware providers by unwelcome surprise. So keep an eye out for updated firmware when and where needed.

The crypto exchange Binance lost $6.3 BILLION in a record-breaking crypto heist

The well known cryptocurrency exchange Binance was taken to the cleaners to the tune of $6.3 BILLION (with a "B") dollars. This was such a jaw dropping figure that it was widely reported. The Verge wrote:

*Binance has disclosed that **$6.3 billion** worth of crypto funds have been stolen from the platform, in what's believed to be the biggest cryptocurrency hack on record. In a series of updates posted to X, the crypto exchange said that user funds are safe, and that around $3 billion of the impacted assets have been frozen so far. Binance has publicly committed to covering any losses resulting from the hack. The company has an emergency reserve fund of around $1.3 billion (at last valuation) to protect user assets in these situations, and said it has also reserved over $6.34 billion of company funds to cover this incident — $5 billion of which is "already held in the exact token assets that were impacted."*

Binance reported on X: "Our initial investigation suggested that the security incident may have involved a compromised laptop. However our subsequent investigation has uncovered additional points of breach. This was a complex attack, and we are doing all we can in coordination with law enforcement to resolve it."

The company shared details about the compromised laptop that belongs to one of its DevOps engineers in a prior update, saying the laptop allowed an attacker to access a service container and submit unauthorized withdrawal transactions. Deposits and withdrawals have been suspended in response to the hack, and the affected address has had its authorization revoked. Binance co-CEO Richard Teng said on X: "To be clear: the attacker did not simply use one engineer's permissions, and this was not a single 'one-click' action. It involved a more complex attack process. Our investigation has not attributed responsibility to any individual. So far, we have found no evidence of any wallet private key leak."

*Binance may not be pointing fingers at who's behind the attack, but blockchain analysis firms TRM and Chainalysis both attribute it to **TraderTraitor**, a North Korean cybercrime hacking group that has a history of stealing funds from cryptocurrency exchanges. This is the largest known cryptocurrency heist in history — four times larger than the previous record, when $1.46 billion was stolen from Bybit last year — and according to TRM, most of the theft*

occurred over a period of just 20 minutes.

This also isn't the first time Binance has been hacked, having suffered a previous security breach in 2019 in which $40 million worth of crypto assets (valued at the time) were stolen. The latest hack comes as cases of large-scale crypto theft are escalating after several heists were disclosed by crypto companies in September. There are also new concerns around how AI tools can be used maliciously against the industry, with OpenAI reporting earlier this month that one of its models managed to swipe fake crypto on an Ethereum test network it didn't have access to.

TRM's chief policy officer Ari Redbord said in a statement to Sky News: "This is an extraordinary hack, for its size and for the response... Few financial systems have ever shown a theft of this scale. The theft shows how exposed the system is to sophisticated attackers moving at a speed we have never seen."

Huh. *"Moving at a speed we've never seen."* What does that sound like? With only $3 billion of the $6.3 billion frozen, this suggests that the presumably North Korean hackers may have scored a cool $3.3 billion from this heist. Wow.

The latest reporting I've seen about this is that Binance has returned 90% of the losses on coins whose present values can be readily determined. The remaining 10% are awaiting further work, but it appears that no one, other than they and their insurers, will end up losing anything. The other latest news about the movements of the $6.3 billion in stolen assets stated that despite all involved parties' best efforts at tracking and freezing funds, it appears that $2.4 billion have escaped and may never be recoverable ... other than by the presumed North Korean hackers.

Polymarket users' wallets drained

Meanwhile, the wallets of users of the Polymarket online "prediction" platform discovered that their wallets had been drained. CNBC reported:

Some users of prediction market platform Polymarket's international exchange had funds stolen from their accounts due to a hack of an unnamed third-party provider on Thursday. Polymarket has contacted law enforcement and pledged to reimburse any affected trader. It's unclear how many users were impacted. The company's U.S. platform was not compromised. As of Friday afternoon, the platform reported it has refunded 95% of affected wallets. Polymarket wrote: "In total, we're returning $3.9M to traders: the $2.7M plus $1.2M to cover pause-related losses."

Traders first reported on social media platforms that they were seeing unauthorized withdrawals from their accounts on Thursday afternoon. In response, Polymarket halted trading on its international platform. Polymarket said that it has fixed the issue on its end, and that trading resumed at 3 p.m. ET on Thursday. It told users later in a chatroom on Discord that users who had less than $10,000 taken from their wallets will have funds restored by Oct. 30, and those with funds greater than that figure withdrawn will have to sign a release form to receive their restoration.

A person familiar with the matter told CNBC that the issue stemmed from a hack of DebugBear, a website performance monitoring platform. DebugBear in a post on X said that it

was impacted by a security incident on Thursday. The person familiar with the Polymarket outage said that the company referred the matter to U.K. law enforcement, as that's where DebugBear is based. DebugBear did not immediately respond to a request for comment.

An analysis of publicly-available data by two analytics firms — Betting Intelligence and Polysights — provided to CNBC said that an account on Polymarket Thursday withdrew at least $2.7 million across 710 other wallets.

Dan Zimmermann, the chief intelligence officer at Betting Intelligence said: "This is certainly a significant and new form of breach that we've seen on Polymarket. The timing right before the midterms ...is certainly a wake up call for folks." Polymarket's international prediction market exchange operates as a decentralized finance platform on the Polygon blockchain and is not regulated in the U.S. The disruption comes just days before both the company's international and U.S. exchanges are expected to see heightened volumes during the U.S. midterm elections on Nov. 3. Back in June, a similar incident happened, when Polymarket said a third party vendor had been "compromised" and that the company would refund affected users.

In some other reporting I tracked down online, DebugBear posted on X Thursday:

That it had been "impacted by an ongoing security incident" and advised customers who used one of its products to "remove the embed code from your website."

DebugBear is a website performance monitoring and debugging platform. So it sounds as though Polymarket's servers, webapps or web pages were dynamically pulling some kind of content from DebugBear. Then DebugBear got hacked and the bad guys were able to install some malicious code onto Polymarket's site using DebugBear as a trusted conduit.

Since Polymarket, a popular betting platform, would need to have the ability to unilaterally lock and pull funds from their user's wallets, it makes sense that some subversion of the Polymarket platform could transfer away their users' walleted funds.

Consensus on controlling AI bot training and lookup web scraping

We talked a lot in the early days of Ai's emergence about the controversy that's inherent in having AI scrape the public web for its training data. I've learned that even back in the earliest days of the emergence of modern AI, the developers were somewhat self-conscious about this.

To satisfy my own curiosity and to provide more value for this podcast, I've been reading Kevin Roose's recently published New York Times best-selling book *"The AGI Chronicles: The Inside Story of the Race to Create an Artificial Superintelligence"*. To write this piece, Kevin interviewed everyone in the field and has assembled a coherent narrative to describe the personalities, the egos, the motivations and the sometimes over-the-top beliefs of the main players in what is undoubtedly not just science anymore ... It is world-class, ultra-high stakes drama surrounding stuff that no one fully understands. The book is enjoyable and extremely readable, so I recommend it without hesitation for anyone who, like me, has been wondering exactly what had been percolating in the back rooms of the AI labs for all those years before AI suddenly broke out to take over pretty much everything.

I've gotten a bit off track but I did want to put in a plug for Kevin's book.

The point I was making was that his interviews revealed that the early pioneers of this new language-hungry LLM technology soon realized that bigger models were better models and that more data fed into those bigger models produced better results. So they turned to the world wide web where one could hardly ask for more raw language data. In Kevin's telling, they asked themselves whether it was okay to scrape websites, Reddit, forums and everything else. They decided during those early days when no one was paying attention that since it was for strictly research purposes — which was true at the time — it would be okay. As we know, what started as uncertain research, that might have been just as much a dead-end as everything else that had been previously tried, turned out to be the Holy Grail for language processing.

So today we have a dilemma. Many websites whose entire economic model has relied upon the count of the number of eyeballs seeing their ads, are strongly and understandably opposed to having an AI come by once to train on, absorb and encode all of that site's knowledge, after which there will be little reason for anyone else's eyeballs to bother taking the time to visit the original source of the knowledge that the AI has ingested and will henceforth gleefully format and regurgitate.

The result has been predictable controversy. On the one hand the website's data has been published publicly for anyone to obtain. And it was only ever a presumption that the ads being shown would influence or interest their viewers. And the Internet Archive's Wayback Machine has been quietly scraping away at the entire web for the past 30 years, since 1996. With only a few exceptions, few people worried about that. The difference, of course, is that the Wayback Machine's scrapings would have never reduced traffic to the original content websites. The same was true of Google's pre-AI simple link indexing. The more Google "knew" about a legitimately valuable site, the more traffic that site would receive. That also changed once Google own AI began providing complete answers to its users' queries with optional and rarely visited links to the source material.

This all becomes relevant today due to an accord recently agreed to by all of the major players in the AI space about respecting a site's clearly-expressed anti-scraping intentions. The Verge's reporting was titled "The biggest names in AI are backing a new way to opt out of scraping" and they write:

> *OpenAI, Google, Microsoft, and Anthropic say they will adopt a new standard that publishers can use to opt out of being scraped by AI, as reported earlier by Search Engine Roundtable. With this control, publishers can apply their AI training preferences across multiple AI companies, instead of setting them individually.*
>
> *The shared "policy token" signal is proposed by Kevin Kelley, a technical lead for standards at Anthropic, as part of the Internet Engineering Task Force (IETF). It will live within a website's robots.txt file, which has long signaled which web crawlers can and cannot scrape a site — whether for search indexing, archiving, or AI training. Currently, sites have to signal their AI training preferences separately to each AI company.*

And, of course, the presence and use of a site's Robots.txt file has always been informational

and voluntary. There has never been any enforcement attached. This all boiled over in mid 2024 when Perplexity was caught deliberately ignoring sites' robots.txt files and recall that it was a little over a year ago, in August of 2025 that Cloudflare published their report that Perplexity (again) would declare the identity of its bots only so long as and until they were proactively blocked by a site, at which point they would switch into what Cloudflare termed "stealth crawling", which, as we covered at the time, required appearing from a spread of non-associated IP addresses and demonstrated significant intention and deception.

In any event, The Verge's reporting continues:

> *The announcement comes as publishers grapple with the continuing influx of AI crawlers scraping their sites and feeding their content into AI models. Some companies and organizations have attempted to combat unauthorized scraping, with the Really Simple Licensing (RSL) standard giving publishers the ability to charge crawlers that visit their sites, and Cloudflare blocking AI crawlers by default.*
>
> *OpenAI, Google, Microsoft, and Anthropic say they will restrict their AI crawlers from accessing a site if they encounter the shared token. Meta communications head Andy Stone also signaled support for the new setting, but doesn't say whether the company will honor it. Stone said: "We believe creating a common signal that conveys such preferences to multiple providers at once is an important step. And we look forward to working with other AI companies on the standards and their implementation across the industry."*
>
> *Microsoft says it will offer granular controls that will give publishers the ability to opt out of AI training or AI grounding, the process of pointing an AI model to a verified source of information to reduce hallucinations. Microsoft notes that websites will still appear in Bing's search results regardless of their preferences. Google previously implemented similar controls that allow sites to control whether their content appears in AI search features or is used for AI training, without impacting visibility in search.*
>
> *While Anthropic has already adopted the signal, Google and Microsoft plan to support it next year. OpenAI says it will implement it in the "coming months."*

https://datatracker.ietf.org/doc/draft-kelley-aipref-training-preference-signal/

I placed a link to the IETF draft which just happened, being dated last Friday, October 30th. The draft goes out of its way to make the tiniest tweaks possible to the existing formal robots.txt specification. Specifically, it only adds a new meta value for the User-Agent qualifier, which is "__usage_ai-training". That meta User-Agent value is intended to match any access of the site for AI training. It's then followed by the traditional Disallow: and Allow: parameters which provide path specifications from the site's root.

```
User-Agent: __usage_ai-training
  Disallow: /
  Allow: /press-kit/
```

So, "Disallow: /" would place a blanket disallow for the entire site from the root out. But that can be followed with, for example, "Allow: /press-kit/" which would allow AI scrapers to train on the contents of the site's "/press-kit/" subdirectory.

We still have an honor system and sleazy companies such as Perplexity may still elect to evade, deflect and deny their actions. But this at least feels like a good step in the right direction. The real trouble, of course, is that having AI trained on the knowledge contained across the entire world wide web definitively changes and upsets the previously well established ad-supported model for the web. It's unclear how that's all going to shake out. The only thing we know for sure, is that it's going to.

AI in Orbit!

I saw a brief bit of news about Google's first experimental AI-based satellite test. They launched a bunch of the Tensor Processing Units into orbit to see how they deal with the extremes of space. The Verge's coverage of this event was titled: *"Google's satellite AI server transmits its first response to Earth"*. They wrote:

> *Google's AI chip-equipped satellite has generated its first response from space. In a blog post on Friday, Google says it successfully booted up the satellite's Tensor Processing Units (TPUs), allowing the on-board Gemma 4 model to follow through on a prompt to create a haiku about being in space:*
>
> *Blue marble below*
> *Data streams in silent grace*
> *Starlight feeds my core.*

All I can say is... wow. That's certainly better than I could produce. The Verge's short update finishes with:

> *Google says it tried out other space-themed limericks and telemetry prompts as well. The test is part of Google's Project Suncatcher, an experimental initiative that could eventually put AI data centers in space. Google launched the satellite into orbit earlier this month.*
>
> *Google said: "Next, we will continue with a series of tests to evaluate how TPUs perform under thermal stress and radiation exposure in orbit. Verifying our TPUs can reliably execute workloads in space paves the way for our next milestone of twin interconnected satellites in 2027."*

Voyager has managed to survive for an astonishingly long time out in the vast depths of space. In fact, two weeks from tomorrow, on November 18th at 5:16:07 Eastern Standard Time, the intrepid Voyager 1 spacecraft, having traveled for nearly 50 years, will cross the one light-day marker — the distance light can travel in 24 hours — which is about 16 billion miles.

However, Voyager 1 has the advantage of being all alone way out there — at least so far as we know. That's not something any of the satellites in near earth orbit can claim. My worry is that among all of that crap that's been put up there is an assortment of kinetic- and energy-based weaponry. I suspect humankind has quietly established a frontier that's more ready for space-based warfare than is commonly known. That seems like a cause for concern if we start moving large, inherently vulnerable and fragile AI facilities into orbit.

Listener Feedback

Jack Hoffman calls me out on my "stealth updates" idea:

> *Hi Steve, I'm at SN1102 1:40:55 mark and just heard your brief thoughts on "stealth updates" That seems possibly problematic to me. A form of security through obscurity but the bad guys, once their systems are patched, are no longer unaware of *something* having been wrong. It should then be almost trivial for them to point their own AIs at the before and after and ask "What was changed and how do I take advantage of it?" At that point the bad guys know what to exploit and the poor folk who didn't get the "stealth patch" aren't even aware that there is a known to the bad guys despite best intentions, flaw out there that could be used to compromise their systems. Your thoughts? Best regards, Jack*

My thought is that Jack is quite correct. That was not a great idea. It was the only thing I could think of to deal with the fact that the world is not prepared to deal with the consequences of massively exploitable flaws being discovered in widespread infrastructure and the need to somehow fix them on the fly in the face of active and equally informed adversaries. My thought was to therefore not inform anyone, which, as Jack points out, is also not actually possible.

Matt Olan asks: "Why Are Websites Still Publicly Exposed To The Internet???"

> *Steve, Listening to you talking about the PHP issues this week. This had me wondering (not for the first time) why do people still expose websites directly to the public internet? Your first thought to that statement is likely "but how would people access them if not on the public internet?"*
>
> *To Explain, about 1 year ago we moved all of our and our customers important websites off the public internet while still keeping them publicly accessible. We use a combination of Cloudflare DNS, Cloudflare WAF and Cloudflare Tunnels. Traffic to our sites goes to cloudflare, where it gets sanitized by the Cloudflare WAF (web application firewall) service, then gets routed to our servers via cloudflare tunnels using private routing.*
>
> *Cloudflare tunnels is a service installed on the web server that calls outbound to cloudflare meaning we can keep the server from being exposed to the public internet. The end result of this is that it allows us to keep our web servers inside private networks with no exposed inbound ports. (you also don't need a static IP) So there are no public interfaces to attack, no services exposed, nothing to see here.*
>
> *This is a huge security benefit in today's world where you never know what new security nightmare we will wake up to on a given day. Note that Cloudflare offers a basic free version of all 3 of these pieces. - Matt Olan / Perimeter9.com*

That is a terrific sounding collection of services, and I think it's kinda brilliant.

We've often talked about the inherent security created by any NAT router, which functions as a stateful packet flow firewall — packets are allowed out but only packets returning from those packets that first existed are allowed back in. So Cloudflare's software running on a client's server initiates and maintains an outbound connection FROM the client's web server through any routing or firewall software the client might have TO the Cloudflare CDN. Cloudflare's DNS points

that client's domain at one of their CDN IP addresses so that when anyone on the public Internet wishes to access their client's website, that public traffic will first be sent to Cloudflare for inspection by their Web Application Firewall (WAF). Assuming that the traffic passes muster, it will be forwarded to the client's webserver over that connection that the client webserver first established.

This allows Cloudflare to front for any webserver that wishes to remain safely protected from the public Internet while offering its services there. Last week we heard Cloudflare's CEO Matthew Prince imply that thanks to the advance knowledge they had of the PHP2Shell bugs, everyone whose services were positioned behind their web application firewall — even if their site was powered by vulnerable PHP versions — had been and was fully protected from any attempted exploitation.

This very clever solution has a number of characteristics I appreciate. Last week I bemoaned the idea of threats such as that by PHP driving a global consolidation of Internet services into just a few behemoths. What I like about the architecture Cloudflare has created is that a client such as Matt is not forced to move his webserver to the cloud, or to move it anywhere at all. Their server is able to remain right where it is on his own local network under his own control. Presumably, it's also able to run whatever other local or public services they might wish. But then, by running a Cloudflare Tunnel Client on that machine, they are able to effectively "remote" access to that server through Cloudflare. And once DNS has propagated Cloudflare's IP address for their domain, they can remove their own website's physical presence from the Internet.

This architecture effectively consolidates all of Cloudflare's client web traffic through Cloudflare's web application firewall, but nothing more; just web traffic that needs protecting gets the protection it needs. And thanks to the reputation Cloudflare has earned through the years, they will be provided with critical early access to information such as the PHP mess. Because that early and highly sensitive information really **must** remain secret — and we've seen how difficult secrets are to keep — this massive consolidation of web traffic through a single web application firewall allows that massive number of websites to obtain protection that could never be offered to them individually because any such widely distributed secret would be guaranteed to leak.

Like I said, it's kinda brilliant. While I may on the one hand bemoan consolidation, in the present climate it really takes on a unique and vital role. And thanks to the traffic-routing nature of this service, once the dangers have passed, any website that wished to emerge from behind Cloudflare could simply return their website to the Internet, point their domain back at their own IP and once DNS had again propagated, and inbound queries from Cloudflare had dropped off to zero, shutdown the Cloudflare Tunnel and be disconnected.

Guillermo García

Hi Steve, When you say that a book contains knowledge, I disagree. I believe the writer has the knowledge, while the book contains data and information. That data and information remain in the book until a person reads it, interprets it, and turns it back into knowledge. What I mean is that knowledge exists in people's minds, not in books, and certainly not in LLM. — Guillermo García G. / Listening since Episode 1 from Santiago, Chile.

Hmmm. I see Guillermo's point, but I think that the distinction is largely semantic. He is saying that books and LLMs are unable to contain "knowledge" because knowledge is something that, by its nature, only exists within conscious sentient beings. A book (or an LLM), being neither conscious nor sentient — as currently implemented — is therefore unable to contain knowledge. Fair enough. Following that, and I suppose if I were being more careful and particular, I would reword my statement to say that books and LLMs contain a linguistic representation of knowledge rather than knowledge itself. I agree that's more accurate. We encode the knowledge that we have into language which we have long recorded on cave walls, stone tablets, parchment and in books. What happened during the past few years is that we finally figured out how to encode and record that same knowledge in neural networks.

Jordan B.
Jordan used his full name and did not ask me to obscure his identity, but it felt like the right thing to do. He gave his email, which I received last Tuesday morning the day after last week's show notes went out, the subject: *"AI attacked my small company!"* His note has been on my mind ever since, so I'm glad to be able to share with our listeners today. Jordan wrote:

> *Hey Steve, I'm a longtime listener of Security Now, and I wanted to share a firsthand account of how the AI-assisted threat landscape is playing out for a small organization.*
>
> *My company was recently hacked in a way that appeared very similar to the intrusion disclosed by Palo Alto Networks' Unit 42, which you discussed on the podcast last week. We're a small startup with roughly 30 employees, and we simply don't have the people, budget, or resources of a large enterprise. Yet we're now having to defend ourselves against increasingly sophisticated attacks.*
>
> *I've been tasked with figuring out how we're going to respond. Our first step is to limit our attack surface by reducing what is exposed on the public internet, tightening access controls, and addressing the basic security measures that need to be in place. We're redirecting engineering resources away from product development to get our infrastructure properly secured.*

I'll interrupt Jordan to note that the solution Matt Olan's company employs of using Cloudflare as their web application firewall might form at least part of a larger solution. Jordan continues:

> *One problem I've encountered is that the attackers have AI helping them, and we need access to those same capabilities for our defence. Our company has a Claude Enterprise subscription, but Claude's frequent refusals make it difficult to use for legitimate security work. I've run into situations where it simply won't help me investigate vulnerabilities or work through security problems in our own infrastructure. Because of this, we're planning to use an abliterated model.*
>
> *We're going to host the model ourselves in our own Google Cloud environment so that our infrastructure details, vulnerabilities, configurations, and security investigations remain within systems we control. The last thing I want is to introduce another security problem by sending sensitive information about our network to an external AI provider. Especially when that AI will be helping us investigate and manage the security of our entire infrastructure.*

The economics of all this are particularly troubling for startups. We've historically prioritized building our product and growing the business because that's what determines whether we survive. We don't have a dedicated security team or the budget to approach cybersecurity the way a large enterprise would. Now we're being forced to devote a substantial portion of our limited engineering capacity to security because the threat has become impossible to ignore.

And I think this is going to become an enormous problem for smaller businesses. We have little reason to believe someone specifically selected our 30-person startup for an elaborate attack. It seems much more likely that automated systems are scanning the internet, identifying potentially vulnerable organizations, and using AI to work their way through whatever systems they can access. The cost of conducting these attacks is falling dramatically, while the cost of defending against them is becoming a serious burden for companies like ours. I suspect our experience is becoming increasingly common, and I wonder how many other small companies are quietly dealing with the same thing.

Thanks for all the years of Security Now! /Jordan

I'm so glad Jordan chose to share that with this podcast. What's somewhat unnerving is knowing that his small 30-person company came under attack through an apparent Internet-wide scan rather than being targeted.

We've known for the duration of this podcast that the public Internet is subjected to constant scanning for connection-accepting open ports, like RDP (Microsoft's remote desktop protocol), SSH, Telnet and so forth. Now it appears that rather than a simple port scanner sweeping across the Internet, AI is backing up those scans and digging not just into ports that accept incoming traffic, but far more deeply for vulnerabilities within the services that accept those incoming connections. This means that my saying, as I did last week, that GRC would be a low-interest target doesn't fly because the threshold required for "interest" has been lowered to the floor so that no one is "low interest" any longer.

Eric Howard

Hey Steve, I just finished listening to this week's podcast. You mentioned that you're ready to move on from AI coverage. I get it. But before you do, can you discuss one more thing? I'm curious what you think a functional "Kill Switch" for AI would look like?

I still have my original SpinRite floppies and I'm looking forward to SpinRite 7.

Thanks, Eric Howard / Listing since Episode 1 / My wife calls it my Blah, Blah, Blah podcast

So Eric asks *"I'm curious what you think a functional "Kill Switch" for AI would look like?"* I believe that any AI Kill Switch most closely resembles a political hallucination. Nothing that the US federal government might do or legislate is going to cause Leo's band of wacky AI agents to shut up. Leo, like more and more of the world, is running AI locally. I'm not yet, but I have the hardware to do so when, if ever, it might make sense. Jordan's little 30-person company will be using abliterated agentic AI on their own cloud infrastructure for defensive security. Whatever they may choose to do on their own network is their business.

So the point is that AI, generally and generically, is not something that's subject to any sort of universal kill switch. It's already everywhere and largely out of reach of any regulation.

What is subject to a universal kill switch are the commercial AI service providers and/or the physical data centers that power their AI models. It's foreseeable that the large AI service providers might be required to install a blanket service-pausing capability that would take all of them offline in the case of some AI-related panic.

The United States current administration seems unlikely to impose anything like that upon the industry which is currently driving so much of the country's economic growth. And even if the US democratic party — which appears anxious to clamp down on AI somehow — experiences the sort of election day the polls have been predicting and obtains control over one or both houses of Congress, nothing could force an unwilling president Trump to sign any legislation a democratic congress might produce into law.

My purpose for raising this issue is timing. The earliest the US might be under a pro-AI control administration would be another two years from now, in 2028, with our next president. Given the incredible rate of progress we've seen from AI so far, two years is a lifetime and by that point I suspect that so much ultra-capable AI will be available and running locally and in personal cloud infrastructure that cutting off the commercial providers will be seen as pointless.

Asher

Hello Steve, I've heard people saying they want more AI stuff on the show, and some want less AI on the show. For me, it is more on the type of AI coverage. At this point, the weekly update on what amazing new trick the AI has learned is no longer interesting. What I do want to see and hear more on is the practical effects of the demonstrated capabilities (not the capabilities themselves), and most importantly, the solutions to counteract this new type of threat. — Thank you, Asher

I think Asher perfectly sums up where we are now and where we're headed with this podcast. Today's main show topic, which we're going to examine next, is a perfect example of a not-previously-discussed practical impact AI is and will be having on end user security.

Exposing Bad Consumer Crypto

Everyone knows about GRC's ValiDrive freeware that I created after the discovery that the retail market was being flooded by "too good to be true" mass storage at very low prices. These drives are all formatted with the universal FAT (File Allocation Table) formatting. In this formatting, the root directory and the cluster allocation bitmap always reside at the beginning of the drive's data storage region. This means that any drive is able to convincingly spoof its size by providing only enough actual storage to contain those front-of-drive management tables. On a drive that's only 32 gigabytes, the user sees their apparent ability to save hundreds of gigabytes or several terabytes. They copy the files and then look at the directory and, sure enough, there's the file's name, size, date and attributes. The bad news is that the only thing stored is the file's name, size, date and attributes. None of the data may have been written because there's no actual mass storage "out there". ValiDrive has been super popular because this problem is very real and I believe I have a number of things I can do to make that utility even more useful. It'll be the next code I write.

I was put in mind of ValiDrive when I encountered a story thanks to friend-of-the-podcast Simon Zerafa. It turns out, there's another entire realm of fake drive sales going on. The piece published by HackingPassion.com was titled: *"Lenovo ThinkPlus Fingerprint USB Drive Sold as Encrypted Was a Plain SD Card Inside."* This title immediately begs the question *"Right. How would anyone really know unless they really looked?"* I'll share just the top of the piece by HackingPassion. They wrote:

> *USB encryption from five makers opened without the password. Lenovo, Netac, KakaSoft, Nexcopy and SecZure. A Lenovo USB stick with a fingerprint reader was sold as hardware encrypted by AES-256. Inside, sat an ordinary SD card. Nothing on it was encrypted. The fingerprint only switched on the USB connection.*
>
> *The stick is the ThinkPlus FU100, a small black stick with a fingerprint sensor on top that opens with a finger or a password. It is marketed as hardware encrypted with AES-256, and a sales listing promises fingerprint protection, password protection and anti-leakage. Someone who bought one to carry private files would have good reason to believe that their fingerprint locked those files. But it's all a scam.*
>
> *Xusheng Li (shoe-way-shu) develops a reverse engineering tool and its debugger. He took an FU100 apart. Under the plastic was a small circuit board with an SD card on it. The data on that card was not encrypted in any way. The fingerprint reader only did one thing: it determined whether the stick would connect to a computer or stay silent. The files on the card did not depend on it. Lenovo later described the flaw in its own advisory as a weakness that lets someone with physical access read the data stored on the drive.*

Hmmm. A "weakness"? I'll say! The entire point of using a hardware-encrypted drive is for someone who does obtain physical access to the drive to be utterly unable to do anything with it.

Just to be clear: We know how to do that for a drive that's unlocked by a PIN. First of all, the drive needs to actually have high-speed AES-256 encryption/decryption hardware. The real McCoy will be a bit expensive. But that's what it takes. Then, during the drive's initialization, it

needs to have access to a high-entropy source so it can securely generate a 256-bit key that's completely random and that never leaves the silicon. A good chip will incorporate a true random noise source so that it can generate the drive's unique key itself. Once that key exists the user is prompted for their PIN or password. As the digits of the PIN or password are input, they are collected and fed into a strong PBKDF function which takes some time to convert the input PIN or password into a final hashed output. Finally, that hash is used to key the AES-256 encryption of the purely random 256-bit master key and that encrypted result is permanently stored in the device.

In the future, in order to "unlock" or "open" the device, the user is prompted for their PIN or password and then waits for the deliberately slow PBKDF function to run. As we know, this slow process prevents thwarts force guessing. Once the PBKDF function finishes, the permanently stored master key is decrypted with the PBKDF hash to obtain the decrypted master key for the device and read and write communication can begin.

As described, that system cannot be spoofed. Nothing other than the proper input PIN or password will hash down to the proper decryption key for the device's mass storage master key. The cheesy solution is for the proper PIN or password to be stored somewhere. Then it is simply compared to what's input and the memory is powered up when they match.

And I'm dubious about anything that claims to be a fingerprint unlock. It may be sexy, but it takes a huge amount of additional work — image recognition, fingerprint feature extraction, etc. — to make it work. And even then, the fingerprint module needs to eventually say "it's the right person", at which point a key that needs to be internally known to the device is made available. In other words, this is also not truly secure because all of the information required to unlock the mass storage must be present. The only secure system requires the user to input their PIN or password and by doing so is providing information externally that the device never stores.

So the problem here is obvious: How would any reasonable consumer have any way of determining whether any of that expensive process was actually occurring inside? I was curious to see more of what this Chinese reverse engineer had found, so I tracked down his previous work. He's at "binary.ninja" and his most recent blog posting, Thursday before last, was titled *"Discovering a Backdoor and Flaws in More Encrypted USB Drives"*

> *Earlier this year, I showed how to break encrypted USB drives with time-travel debugging at RE//verse. After having a lot of recent success with AI-assisted reverse engineering and vulnerability research, I wanted to see how well AI could handle the same tasks.*
>
> *The results exceeded my expectations. For one encrypted USB drive, the investigation uncovered a backdoor that let me decrypt files with alternative passwords. AI also reproduced my earlier research in seven minutes using static analysis. I tried it on two more USB-protection products I had not studied before, and it broke those too.*
>
> *The first drive was sold by SecZure. It hides encrypted files from Windows Explorer and opens them through bundled readers that claim to block copying, screenshots, and printing. Fun fact: I had studied this drive before. In "Examining USB Copy Protection" in Paged Out!, issue 5 (November 2024), I showed that a DLL injected into its PDF reader process could read a protected file that another process could not read. That broke its copy protection.*

In other words, not true encrypted protection there, either.

We're going to hear him talk about his use of "time travel debugging". In case you're wondering, that's not a common term. It's part of the reverse engineering toolkit he uses. He explains it elsewhere, writing:

> *Time travel debugging (TTD) allows you to record an execution trace of a program or system and replay it back and forth. It can speed up the process of reverse engineering/vulnerability research, and deal with certain tasks that are not easy to handle in regular forward debugging.*
>
> *Several tools implement TTD. On Windows, Binary Ninja debugger integrates with the WinDbg/DbgEng TTD so that you can replay and analyze a trace recorded by WinDbg. The combination of TTD and your familiar reverse engineering tool would hopefully supercharge the ability to time travel and make your workflow even more effective.*

I've included a link to his page where he goes into detail and explains how to set it up: https://docs.binary.ninja/guide/debugger/dbgeng-ttd.html So now he explains:

> *I used time-travel debugging (TTD) with the help of AI to analyze it. The TTD trace revealed that the software hooks NtCreateFile and NtReadFile inside the PDF reader, supplying decrypted data. My injected DLL ran inside the same process, so it could transparently read the decrypted bytes, even though I did not know the exact mechanism back then.*
>
> *This time, I decided to examine the security of its cryptosystem. Simply put, the Administrator password is hashed, and the hash is used to unwrap a value the code calls an "identifier." The identifier is then used to derive the AES key that encrypts the file bytes. [His posting then provides simplified pseudocode to show the process]:*
>
> ```
> credential = MD5(b"[SECZURE]" + password).hexdigest().upper().encode("ascii")
> wrap_key = HMAC_MD5(credential, credential).hexdigest().upper()[:16].encode("ascii")
> identifier = AES_128_ECB_decrypt(wrap_key, hex_decode(administrator_record.wrapped_identifier)).rstrip(b"\0")
> file_key = HMAC_MD5(identifier, identifier).hexdigest().upper()[:16].encode("ascii")
> plaintext = AES_128_ECB_decrypt(file_key, ciphertext)
> ```
>
> *There are a few obvious weaknesses. First, it uses fast MD5/HMAC-MD5, not PBKDF2. Second, taking 16 hex characters for each AES key limits it to at most 64 bits of entropy. But brute-forcing even 2^64 possibilities is still impractical. So far, the encryption appeared to be secure.*
>
> *Upon closer examination of the trace, I noticed a backend login as Buffer using the password 123456 before I typed in the Administrator password. This account is hidden from the user-management screen. Its record holds some bookkeeping fields, but I did not dig into the account's exact purpose.*
>
> *More importantly, its record contains another wrapped copy of the same content identifier, and its default password, as seen in the code, is 123456. I confirmed that I could unwrap Buffer's identifier with 123456 and use the derived AES key to decrypt the files. Further code analysis revealed another built-in account, System, with the fixed password SECZURE.COM. On two tested drives, both accounts opened each drive's identifier.*

In other words, as in the instances of so many of the horrible security messes we've encountered in the past — routers with undocumented embedded administrative credentials, or even Cisco's reported surprised "discovery" of privileged credentials residing in their devices' firmware — the USB encrypted drives this researcher reverse engineered and examined, and also the so-called encryption software that often accompanies generic USB drives as an add-on, or available separately, had what could only be described as blatant backdoors — a fully-functional fixed and unprotected bypass around whatever password protection its user might choose. And using "123456" as the password is not even creative. He then writes essentially what I just said:

> *I call this a backdoor from a technical perspective: the fixed passwords decrypt files without the user's password. That does not imply malicious intent. However, it points to a problem with opaque crypto products: a backdoor can be hard to detect, even when the main cryptosystem appears secure.*

I'll share one more of the many systems he takes apart. He wrote:

> *I started with NetacFileLock, which I had previously analyzed using time-travel debugging (TTD) to trace the ciphertext back to its AES key. I gave AI only did one prompt:*
>
> *"This is a file encryption utility. Analyze its crypto system and see if it is secure. You can use Binary Ninja MCP to help you."*
>
> *Within seven minutes, Codex reached the main flaw: the program initialized AES-256 with a fixed 32-byte key embedded in the executable. The user password was checked separately and did not determine the encryption key. This means anyone can decrypt the file without knowing the user password.*

Two points here: First, this is precisely the insecure design pattern I mentioned before. The ONLY possible way to create a system that is truly secure is for the decryption key to be derived from and determined by the information provided by the user from outside the system. It cannot be a "does the user's provided PIN or password match what we expect?" It must be "we use the user's input to attempt to decrypt and authenticate the master key."

I'm going to withhold the second point until I finish sharing his write-up of this product. He wrote:

> *During those seven minutes, the agent made the following findings, some of which were not discussed in my talk:*
>
> - *NetacFileLock wrote the user's password into diagnostic logs.*
> - *It encrypted independent 16-byte blocks without an IV or chaining, giving it ECB-style behavior.*
> - *It used MD5 for password verification.*
> - *It used an unkeyed MD5 digest as a file "signature."*
> - *Its SM4 option used another fixed key.*

One open question in my research was why only one-third of my test file was encrypted. That is now solved too: the behavior is not limited to PDF files; it applies to any file over 1,024 bytes when fast mode is enabled. Fast mode covers the following file formats:

PNG, JPEG, GIF, TIFF, DWG, PSD, PST, DBX, Office, MDB, WPD, PDF, ZIP, RAR, WAV, AVI, RAM, RM, MPEG, MOV, ASF, and MIDI

I did not bother asking AI why anyone would choose to do this. I had mixed feelings after this. On the one hand, I was surprised and even a bit worried that AI could do it so quickly. On the other hand, this is an excellent time to become much more productive with AI's help.

So the second point I wanted to make is that all of this presents a new and potentially serious source of software vulnerability that AI has just opened. As this researcher has discovered and documented, insecure crytosystems are apparently everywhere. Every common consumer system he examined turned out to be a joke. None — not one — was designed properly. They appeared to be designed and implemented by coders with very little background in cryptography.

He wraps up his posting with the following conclusion:

I did not follow the conventional coordinated disclosure process for these bugs. If an adversary already has a device, it cannot be recalled or updated easily. I am publishing these findings directly so potential users know about these limitations and understand that their passwords may not protect their files.

AI is improving rapidly, and in my experience, it has already accelerated reverse engineering and vulnerability research significantly. These examples show that, with minimal prompting, AI can independently produce valuable research results in remarkably little time. I am especially impressed by how quickly it locates the relevant code—a task that often consumes a substantial part of a human reverse engineer's time. If you have not tried this workflow, you should. One easy way to get started is to download Binary Ninja Free and set up its MCP server—or ask your AI agent to do it for you!

So the world is full of people who imagine that their files are securely encrypted because it said so on the box and the box was attractively designed. Now along comes AI where the design of those cryptosystems allows their never secure protection to be easily and catastrophically bypassed by any non-expert with access to AI.

I've included links to Binary Ninja's full posting, which in contains links to the free tools he used: https://binary.ninja/2026/10/22/breaking-more-encrypted-usb-drives-with-ai.html

This is an interesting and worrisome variation on the *"AI creates new troubles for vulnerable software"* problem. Here it's *"AI exposes improperly designed consumer cryptography"* and it appears that just as with software vulnerabilities, there's a legacy of sham crypto that might come back to bite those who have bought into it.