

# Security Now! #1072 - 03-31-26

## LiteLLM

### This week on Security Now!

- Will California require Linux to verify its user's age.
- Apple's iOS 26.4 requires UK users to prove their age.
- Russia chooses to use home grown 5G mobile encryption.
- Ukraine knew the webcam was installed by Russian spies.
- Google moves quantum computing "Q Day" to 2029.
- At RSA, UK's NCSC CEO warns of vibe-coded SaaS replacements.
- More information about nasty ClickFix campaigns.
- More than one in seven Reddit postings are an AI-bot.
- The story behind the LiteLLM disaster that was averted.

**This solution is not recommended if your dog has a wet nose:**



# Security News

## Platform Responsibility

Our first news of the week was inspired by a question I received from a listener. It relates to much of our recent discussions about Internet age verification and specifically to its recent escalation to include operating system platforms. Our listener, who identified himself as "Fred M" wrote:

*Hi Steve, I recently read that FreeDOS was not going to comply with California's age verification requirements. Since FreeDOS is the OS distributed with SpinRite, I was wondering how this would affect you when the new law takes effect? Thanks, Fred.*

Fred's question refers to California's Assembly Bill No. 1043. It's unclear to me why this issue suddenly and recently popped up on everyone's radar. But the Internet is buzzing about it and our listeners have been sending their questions and opinions. The Bill in question was approved by California's governor on October 13th last year and it doesn't take effect until the start of next year, January 1st of 2027. So why all of the sudden awareness? Looking back over the past month, the only thing I could find was that the very popular and respected Tom's Hardware site posted an article about this on March 1st. Tom's Hardware headline was "*California introduces age verification law for all operating systems, including Linux and SteamOS — user age verified during OS account setup.*"

Okay. So, mostly, no Linux users want a nosey government to have its mitts on their beloved independent open source operating system. And since Linux doesn't have any central control authority like Windows, Mac, Android or iOS, they reasoned, there would be no way for that to happen. Right? Right?

Since California legislators also recently proposed requiring all 3D printers to somehow magically identify and refuse to print any handgun components, California's legislators do seem to be having fun asking for things they cannot realistically have. Not that that's stopping them.

So let's first step back and look at this legislation. The section heading is "Age verification signals: software applications and online services." and the section's overview summary says:

*Existing law generally provides protections for minors on the internet, including the California Age-Appropriate Design Code Act that, among other things, requires a business that provides an online service, product, or feature likely to be accessed by children to do certain things, including estimate the age of child users with a reasonable level of certainty appropriate to the risks that arise from the data management practices of the business or apply the privacy and data protections afforded to children to all consumers and prohibits an online service, product, or feature from, among other things, using dark patterns to lead or encourage children to provide personal information beyond what is reasonably expected to provide that online service, product, or feature or to forego privacy protections.*

*This bill, beginning January 1, 2027, would require, among other things related to age verification with respect to software applications, an operating system provider, as defined, to provide an accessible interface at account setup that requires an account holder, as defined, to indicate the birth date, age, or both, of the user of that device for the purpose of providing a*

*signal regarding the user's age bracket to applications available in a covered application store and to provide a developer, as defined, who has requested a signal with respect to a particular user with a digital signal via a reasonably consistent real-time application programming interface regarding whether a user is in any of several age brackets, as prescribed. The bill would require a developer to request a signal with respect to a particular user from an operating system provider or a covered application store when the application is downloaded and launched.*

*This bill would prohibit an operating system provider or a covered application store from using data collected from a third party in an anticompetitive manner, as specified.*

*This bill would punish noncompliance with a civil penalty to be enforced by the Attorney General, as prescribed.*

While it's true that details matter, I'll first note that this is what we've been suggesting Apple with iOS and Google with Android should both somehow arrange to provide. And the way this should be done at least for the use case of smartphones is beginning to take shape.

The parents or guardians of a minor child should be able to configure the birth date of the user of a smartphone and be able to securely lock that date into their child's device. From then on, any time a website, a local application or app store download contains age-restricted content and thus needs to obtain age-gated permission, they may cause the user's local operating system to display a clear and uniform pop-up asking for an age bracket to be provided. If the user wishes to, they may then allow the operating system to inform the requesting application whether its user is under 13 years old, between 13 and 15, between 16 and 18, or over 18 years old. (Those are the four age brackets specified by this California legislation.) If the user declines to provide their age bracket, or if their device has not been set with a date of birth, the requesting site will be told that no age assertion is available and should not deliver age-restricted content.

This solution places the handling and responsibility of their young child's age into the parent's hands – where it should be. All it requires of Apple and Google is that they provide the means to accept, lock and protect that decision, and provide a uniform platform-specific API for making that information available on a case-by- case basis (if so configured) to any entity that inquires. And both companies have been moving incrementally in this direction.

At this point, I cannot find any sympathy for someone who complains that **this** would represent an invasion of an online user's absolute privacy. Opening the front door of our home, walking outside and down the street compromises someone's illusory absolute privacy. We live in a world of laws which attempt to protect vulnerable young people by age-gating where they can go and what they can do with the vast resources that are now online. As a society, we're now working to more fully incorporate all of the many facets of the Internet into our daily lives. To do that responsibly means that a user's age needs to be taken into account.

But when we leave the mostly well-defined realm of personal-use smartphones with their per-user accounts, things become far less clear and clean and we step into a huge mess. Here's what Tom's Hardware wrote which may have been the catalyst for the recent upsurge in online interest and outrage:

*California's Digital Age Assurance Act (AB 1043), signed by Governor Gavin Newsom in October 2025, requires every operating system provider in California to collect age information*

*from users at account setup and transmit that data to app developers via a real-time API, with the law taking effect on January 1, 2027.*

*The law's broad definition of an "operating system provider" as anyone who "develops, licenses, or controls the operating system software on a computer, mobile device, or any other general purpose computing device" — pulls in not just Windows, macOS, Android, and iOS, but also Linux distributions and Valve's SteamOS.*

*According to AB 1043, OS providers must maintain a "reasonably consistent real-time application programming interface" that categorizes users into four age brackets — under 13, 13 to under 16, 16 to under 18, and 18 or older — and hand that signal to any developer who requests it when their app is downloaded or launched.*

*Developers who receive the signal are "deemed to have actual knowledge" of their users' age range under the law, which shifts legal liability for age-appropriate content decisions onto them. Penalties for non-compliance run up to \$2,500 per affected child for negligent violations and \$7,500 for intentional ones, enforced by the California Attorney General.*

*The law does not require photo ID uploads or facial recognition, with users instead simply self-reporting their age, this sets AB 1043 apart from similar laws passed in Texas and Utah that require "commercially reasonable" verification methods, such as government-issued ID checks. Assemblymember Buffy Wicks, who authored the bill, said this "avoids constitutional concerns by focusing strictly on age assurance, not content moderation," in a press release. The bill passed both chambers unanimously, 76-0 in the Assembly and 38-0 in the Senate.*

*Despite signing it, Governor Newsom issued a statement urging the legislature to amend the law before its effective date, citing concerns from streaming services and game developers about "complexities such as multi-user accounts shared by a family member and user profiles utilized across multiple devices." Whether amendments will materialize before January 2027 remains to be seen.*

*Enforcement against Linux distributions, however, is likely to be problematic. Distros like Arch, Ubuntu, Debian, and Gentoo have no centralized account infrastructure, with users downloading ISOs from mirrors worldwide, and can modify source code freely. These small distros lack legal teams or resources to implement the required API, so a more realistic outcome for non-compliant distros is a disclaimer that the software is not intended for use in California.*

I spent some time reading everything I could about this, and I found a posting made by the Reason Foundation a few months ago that I wanted to share. It summarizes the current state of affairs, highlights the ways in which California's new legislation actually represents a useful step forward and also suggests a way out of the mess that California has also created. They wrote:

*California Gov. Gavin Newsom signed the Digital Age Assurance Act (Assembly Bill 1043) into law on Oct. 13, marking a significant evolution in state approaches to online youth safety. There is room for improvement, but the act introduces a meaningful first step toward a more privacy-preserving, age-signaling model intended to minimize data exposure while improving compliance certainty for businesses. This step is a welcome advancement over earlier approaches, but it also creates potential complications if later paired with more restrictive bills, a tradeoff that policymakers should weigh carefully.*

California's AB 1043 mandates that parents or users declare age during initial setup, and the device encodes that information into an encrypted signal that communicates an age bracket to apps and online services. This signal then informs developers of whether the user falls within compliance categories such as "under 13," "13-15," "16-17," or "18+." Unlike previous app store age-verification bills, AB 1043 assigns enforcement authority solely to the California attorney general. It prohibits private lawsuits over violations, thereby reducing compliance anxiety of frivolous private litigation. For families, this model minimizes exposure to ID theft and limits the need to share sensitive information with online platforms.

This approach contrasts with that of Utah, Texas, and Louisiana, which enacted the first statewide app-store age verification laws in 2025. The bills require app stores and developers to verify users' ages through "commercially reasonable" methods. Utah and Louisiana's laws are set to go into effect in 2026, while Texas' has been temporarily blocked by a federal judge on constitutional grounds. Two federal bills, the App Store Accountability Act, introduced by Sen. Mike Lee (R-Utah), and the Parents Over Platforms Act, introduced by Rep. Jake Auchincloss (D-Mass.), include the same "commercially reasonable" language as the state bills. Although this phrasing does not explicitly mandate government ID or biometric checks, it creates strong incentives for app stores to collect the most precise forms of evidence available: driver's licenses, passports, or credit cards. Fearing the risk of lawsuits and non-compliance penalties, companies would default to the most definitive identification techniques.

In 2025 alone, several popular apps that already required government ID checks for age verification suffered significant data breaches, highlighting the privacy risks associated with such mandates. The Tea app, a women-only dating advice platform that required users to upload selfies and copies of government-issued IDs as part of its account verification, experienced a major breach in July that exposed over 70,000 identification images and sensitive personal data. In October, global messaging platform Discord suffered a breach directly tied to its compliance with the United Kingdom's Online Safety Act, which mandates robust age verification for platforms likely to be accessed by minors. To meet these legal requirements, Discord began requiring UK-based users to submit either facial scans, government IDs or the last 4 digits of credit cards for age checks, vastly expanding the pool of highly sensitive data at risk. When hackers later compromised a third-party vendor managing this information, thousands of ID photos and partial credit card details were exposed. These incidents underscore how rigid age-verification systems can turn well-intentioned privacy protections into security liabilities and inadvertently create new vectors for harm.

In contrast, AB 1043 correctly prioritizes privacy and security by using a self-declared age signal rather than a verification process. The law integrates core privacy-by-design principles by separating identity from compliance status and ensuring that user data never leaves local systems in identifiable form. It also provides developers with clearer compliance certainty than Utah-style frameworks, which remain mired in vague terms like "commercially reasonable."

However, there are still issues with AB 1043 that should be addressed. First, the law's mandate that device makers integrate age signals into all devices risks sidelining parents from key digital literacy decisions. For AB 1043 to achieve its stated balance between safety, privacy, and parental empowerment, California could modify its framework to make age signaling optional for parents rather than required.

Second, debates over youth online safety laws raise a subtler issue: their impact on family relationships and parental oversight. Age-verification and age-signal frameworks are often presented as empowering parents, but automation can easily displace meaningful dialogue between parents and their children. True digital literacy depends on ongoing dialogue, trust,

*and continuous education about online risks, not on technical filters alone. When technology assumes the entire role of risk management, it can foster complacency and a false sense of security, as if software settings could replace parental judgment. Policymakers should therefore ensure that digital safety tools operate as supports for families, not substitutes for them.*

*California's initial framework, in this respect, could be refined through a simple but meaningful adjustment: Make the device-level age signal optional for parents rather than compulsory. An opt-in structure would preserve AB 1043's privacy benefits while strengthening family agency. Parents could choose to enable the system during device setup if they desire automated filtering or app age controls, or skip it entirely for now if they prefer to guide their children's use through household rules and open communication. Optional enrollment would further align the policy with California's broader digital rights precedents, reinforcing choice, consent, and proportionality.*

*On the whole, California's AB 1043 represents a meaningful advancement in the national debate on age verification. It replaces high-risk identity checks with privacy-preserving signals, curtails constitutional litigation risks, and clarifies enforcement responsibility. But if the state were to shift to an opt-in model, it could preserve the law's privacy protections, align with its digital rights values, and restore parents to the central role in guiding children's online wellbeing. Age assurance need not come at the expense of privacy or parental autonomy.*

I think this author gets a lot of this exactly right. We would be moving toward an environment where the devices used by someone less than 18 years old could optionally be configured by that minor's parent or guardian to conditionally supply its user's age bracket.

The idea would be that the various operating systems would add a simple API that could be queried by applications running on the platform. If that application is a video game offering age-restricted content, it could learn which version of its game to display to that platform's user. If the application were that platform's app store, it would learn which applications to list and allow to be downloaded and which to filter. And if the application was a web browser, it would learn the age range of its user and could use that information when queried by a remote website. We would need the W3C – the World Wide Web Consortium – to define a standard means for a remote site to query the browser client for its user's age. But that should be trivial.

And as for the non-smartphone platforms such as Windows, macOS, Linux, and StreamOS, all of those platforms already operate with the concept of a "root" or "administrator", whose account should never be used as the daily driver, and daily users who work with far more safe reduced privilege. So those platforms could easily arrange to add date of birth awareness to their user accounts and an API that their client apps, stores and web browsers could query to learn the age bracket of its currently logged-on user. Following exactly the model of the smartphone, this would give parents who **wished** to govern what their young children did, what they saw and where they went, a simple and clean means for doing so. This would in no way encumber any platform's traditional adult users.

All of the online fury and indignation raging over the idea of California attempting to effectively outlaw any OS platform that doesn't provide these services disappears when it's just an optional feature capability that a system's administrator – in this case mom or dad – might choose to employ if, in their role of parent or guardian, they would like to exert some control over the age-gated use of that platform. It's also worth noting that this solution nicely resolves the whole VPN backlash dilemma that's beginning to appear. A VPN would not be of any benefit, since the user's platform, not their IP address, would be producing the age bracket indication.

## Apple begins requiring affirmative age verification

Before we leave this discussion of age verification I wanted to note that Apple has started requiring age verification for their users in the UK and South Korea. With the rollout of iOS v26.4, Apple account holders in those two countries may be asked to register a credit card or take a picture of a government-issued ID. If Apple is able to determine an account holder's age without that – by looking at other signals such as the length of time they've had an account – then no other information will be needed. So it's beginning. As I've said before, I trust Apple more than any other 3rd party to protect its users' privacy.

<https://support.apple.com/en-gb/125662> <https://support.apple.com/en-us/125666>

## Russia to switch to home-grown custom NEA-7 crypto for mobile network

I chose to share this next story because it's so looney and because it serves as another example of the disturbing and growing intersection of politicians and technology. The Risky Business Newsletter covered this puzzler, writing:

*The Russian government is working on a law that would require all mobile operators to use a custom domestically-developed encryption algorithm for the country's 5G mobile network. If the bill passes, all phones sold in Russia going forward will need to support the NEA-7 algorithm or they will not be able to connect to Russian mobile networks which, by 2023, will only support NEA-7. Foreign algorithms such as SNOW (used in Europe), AES (used in the US), and ZUC (used in China) will be supported only until 2032, as part of a transitional phase to allow current smartphones to reach their natural end-of-life.*

*Work on the proposed regulations began last year and the bill is now in its second draft, according to Russian news outlet Izvestia. The bill is part of a broader set of measures designed to hinder the operations of Ukrainian drones and missiles, which have used Russian SIM cards to connect to mobile towers, determine their location, and then guide themselves to planned targets. However, using a custom encryption algorithm to encrypt **5G** traffic won't stop the Ukrainian side from using Russia's mobile network, since they can always fall back to older protocols like **LTE** and **3G**.*

*On the other hand, the proposed law represents a "patriotic" legislative flex. It's the type of unrealistic stuff that's been happening in the Russia Duma recently, to show that Russia is important and still matters on the global stage. As Izvestia points out itself, Russia is insignificant on the mobile market, where it only accounts for 2% of annual sales, so it's very possible that most phone makers won't bother to implement NEA-7 on their chipsets.*

*There is also no base tower equipment that supports the algorithm, which raises the possibility that Russia will be years behind in rolling out its 5G network. The Russian news outlet warns that NEA-7 may be used as a trojan horse by foreign manufacturers to request a favorable market position or a monopoly in exchange for adding the algorithm to their firmware.*

*On Ukraine's side, the answer is likely to be the same as with Russia's rollout of MAX (their Russian messaging system). The Ukrainian intelligence services were delighted that Russia was mandating everyone in their country use an incredibly insecure and easy-to-hack mobile app. Rolling out an untested and largely unknown encryption algorithm for your entire future mobile network may create a major opportunity for hacks and surveillance operations who know their way around encryption, as intelligence services usually do.*

One of the most important lessons taught by the industrial revolution is the incredible power of standardization. If every country had their own screw thread standard, nuts and bolts would be

incompatible with one another and it would be necessary for a shop to redundantly stock a separate supply of bolts for every country of origin. As it is, we have separate metric and imperial threading and look at what a mess even that creates! Another example of standards failure is differing AC outlet plugs around the world – though I’ll admit that they have provided a terrific supply of pictures of the week for this podcast! – which further demonstrates the failure.

My point is that the standards that the world has agreed to – the standards surrounding the Internet, Ethernet and USB are examples – have resulted in astonishing economies thanks to the interchangeability and interconnectivity they provide free of charge, simply by choosing not to roll your own. This clearly demonstrates the insanity of what mother Russia is choosing to do. After 2032, Russian citizens will likely be stuck using Russian-made Android smartphones with no choice in the matter. That’s not progress.

### **Ukraine feeds disinformation to Russian spies**

Speaking of Russia, it seems that Russian intelligence services somehow arranged to install some spying hardware into a Ukrainian drone factory. The device was embedded inside a thermostat but it did much more than control the room temperature since it included a camera, microphone, and router. The story becomes more interesting and fun, though, when we learn that the Ukrainian drone maker, TechEx, was fully aware of the device before it was installed thanks to a warning from Ukrainian intelligence. So the Russian surveillance device was installed, after which TechEx worked with Ukrainian intelligence to supply disinformation, which would be regarded as highly-trusted, to the Russian spies.

### **The big quantum question**

Last year we had some fun looking at a very clear deconstruction of the claims being made about quantum factorization. As we know, the threat posed by the emergence of practical quantum computers is that they may be able to solve the prime factorization problem upon which rests all of the cryptographic security provided by the invention of RSA-style public key cryptography. Last year it appeared that the world had a much longer way to go than was assumed because the takedown we examined convincingly revealed that some slight of hand had been going on behind the scenes with the use of highly contrived factorization targets.

Google, however, appears to disagree. Or perhaps they’re just taking the “better to be safe than sorry” approach. The news is that Google has moved the so-called fateful “Q Day” to 2029 – only three years from now. Google expects threat actors to break classic public key encryption using quantum computers by the end of the decade. They have introduced a 2029 timeline to secure their products with post-quantum cryptography (PQC) protections. Both Chrome and Google Cloud already have PQC protections in place and Android is getting them later this year. We also know that Apple and Signal have both added post-quantum crypto to their messaging platforms. In addition, Cloudflare, AWS, Azure, Meta and Zoom have put PQC in place. Also, TLS v1.3 is already capable of negotiating PQC-encrypted connections with Cloudflare indicating that more than half of all traffic moving through them is already quantum safe.

So, it appears that the emergence of quantum factorization, whether in 3 years or 30, won’t cause any significant upheaval.

### **The CEO of the UK's cybersecurity agency speaks at RSA**

Last Tuesday, during the annual RSA security conference where Leo and Lisa were present to hobnob with many of the podcast network’s sponsors, the CEO of the UK’s NCSC, the country’s cybersecurity agency, spoke to the conference. The Record wrote about his presentation, saying:

*Britain's National Cyber Security Centre warned Tuesday that a rise in so-called "vibe coding" could reshape the software-as-a-service industry while introducing new cybersecurity risks if organizations fail to adapt. The warning coincides with remarks by NCSC chief executive Richard Horne at the RSA Conference in San Francisco, where he urged security professionals to ensure AI coding tools become "a net positive for security."*

*Highlighting, again, how digital societies are facing a surge in cyberattacks exploiting classes of software vulnerabilities that are known about and can be fixed, Horne said there was a risk AI tools would simply propagate the production of insecure software. His comments follow a sharp market sell-off in shares in software and cloud companies in February driven by investor concerns that "vibe coding" — a term used to describe software developed using AI tools and minimal human input — could reduce demand for subscription-based SaaS platforms.*

*Horne told RSA: "The attractions of vibe coding are clear. Disrupting the status quo of manually produced software that is consistently vulnerable is a huge opportunity, but not without risk of its own. The AI tools we use to develop code must be designed and trained from the outset so that they do not introduce or propagate unintended vulnerabilities."*

*In a blog post published alongside the speech, the NCSC said advances in AI-assisted software development are already changing how organizations approach writing code, potentially setting the stage for a significant disruption of the SaaS model over the next few years. Describing the February sell-off as "a billion-dollar wobble" — and referencing the term "SaaSocalypse" — the agency cited anecdotal examples of developers using AI tools to build replacements for SaaS products in a matter of hours, particularly in response to rising subscription costs or feature restrictions. The SaaS industry has dominated enterprise IT by offering subscription-based access to software while offloading infrastructure, maintenance and security to vendors.*

*In its blog post on Tuesday, the NCSC said this dynamic could shift as AI tools make it faster and cheaper to build "bespoke enough" software in-house — driven by the same business incentives that triggered the rise of SaaS companies themselves and the early uptake of cloud computing. But it warned that AI-generated code can be unreliable, difficult to maintain and prone to security flaws, increasing the chance that vulnerable systems could be deployed if those behind the vibe-coded systems were too tolerant of the risks.*

*The NCSC urged organizations to prioritize security as the technology develops, including ensuring AI systems generate secure code by default, verifying the integrity of models and expanding the use of automated code review and testing. The blog post stated: "If security professionals do not lean in from the start, the landscape will evolve without this crucial input, as was arguably the case in the early years of cloud adoption. A challenge the security community will face is that no one yet knows exactly what we need to introduce to ensure the 'vibe coded future' is a safer one. If we face this challenge head on from the start, we have a chance to introduce some strong security fundamentals."*

*The NCSC said any disruption to SaaS is likely to take place over several years, with adoption varying depending on system complexity and organizations' risk tolerance. But the agency said it could "easily imagine" that the only companies in the sector that will survive will be those "that ca not be easily replaced with a vibe-coded alternative, perhaps because their services have themselves become critical to a business, or there are regulatory requirements they meet, or they simply have a critical mass of data across customers."*

So there are a number of interesting takeaways here. First, is the obvious-when-you-look-at-it threat that vibe-coded replacements for software-as-a-service represent. Why would any large

enterprise rent, under an expensive recurring subscription, what a handful of their in-house coders could whip up overnight to create bespoke software that more perfectly fits their needs?

I hadn't stopped to consider it before now, but this entire world of outsourced service industries that have sprung up over the past decade are hugely vulnerable to the emergence of DIY home grown in-house-coded alternatives that vibe coding now makes so easy to create.

We've heard that C-suite executives have said: *"We're only going to hire someone new if you first demonstrate that AI cannot do their job."* So it's not much of a stretch to imagine a similar executive asking: *"Why should we be paying tens of thousands of dollars per month to this annoying outsourcing company when a couple of our guys in the back room can use AI to write the same thing that we will then own, can customize to work exactly the way we want, and can use going forward without any recurring costs?"* It's clear that this will be an accelerating trend in the future.

But the other shoe to drop was the NCSC CEO's primary concern, which was the threat that carefully created, refined and secure SaaS solutions would be too hastily replaced by half-baked, unproven and insecure vibe-coded clones. One of the tendencies we have seen over and over is that security will truly be sacrificed at the altar of economics. Why is everyone pulling and blindly using libraries from open source repositories? Because they appear to work and solve a problem, and the price is right because it's zero. But the truth is that everyone is just holding their breath and hoping for the best. That's not the way security is obtained or maintained. But it is free.

It's going to be very interesting to see what happens as enterprises develop for in-house use, the various systems they once outsourced. Overall, I would expect it to be a clear win, but it likely won't all happen without a few stumbles along the way.

### **An update on ClickFix campaigns**

Last Wednesday, Recorded Future posted the results of one of their threat forensics groups that was looking closely at the insidious ClickFix social engineering attacks. As they wrote elsewhere in describing the nature of these attacks: *"First documented in late 2023, ClickFix has transitioned from a niche social engineering tactic to a cornerstone of the global cybercriminal ecosystem. ClickFix is a social engineering methodology that lures victims into manually executing malicious commands by masquerading as a necessary technical resolution for fabricated system errors or human-verification prompts."* So here's what we learn from Recorded Future's Insikt group. They write:

*Insikt Group identified five distinct clusters leveraging the ClickFix social engineering technique to facilitate initial access to host systems. Observed since at least May 2024, these clusters include those impersonating financial application Intuit QuickBooks and the travel agency Booking.com. Insikt Group leveraged the Recorded Future® HTML Content Analysis dataset, which enables systematic monitoring of embedded web artifacts to identify and track new malicious domains and infrastructure.*

*The clusters demonstrate significant operational variance in lure themes and infrastructure patterns, and highlight the technique's evolution, moving past simple verification by visually fooling victims with various fake challenges and demonstrating technical sophistication through operating system detection to tailor execution chains. Despite these structural differences, its operation is largely the same, showing that ClickFix's core techniques work across platforms and only the social engineering lure needs to be adapted to the victim. Threat actors manipulate victims into executing malicious, obfuscated commands directly within native*

*system tools like the Windows Run dialog box or macOS Terminal.*

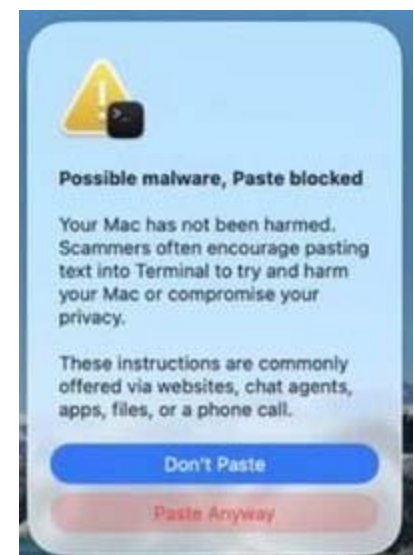
*This living-off-the-land (LotL) approach allows malicious scripts to execute in-memory, effectively bypassing traditional browser security and endpoint controls. Parallel clusters targeting sectors as diverse as accounting, real estate, and legal services indicates that ClickFix has transitioned into a standardized, high-ROI template for both cybercriminal and potentially advanced persistent threat (APT) groups.*

*To protect against these threats, security defenders should move beyond simple indicator blocking and prioritize aggressive behavioral hardening. Key recommendations include disabling the Windows Run dialog box via Group Policy Objects (GPO), implementing PowerShell Constrained Language Mode (CLM), and operationalizing Digital Risk Prevention tools such as Recorded Future's Malicious Websites to identify and mitigate threats to your digital assets.*

*Based on increasing use since 2024, Insikt Group assesses that the ClickFix methodology will very likely remain a primary initial access vector throughout 2026 as threat actors continue to social engineer victims to enable exploitation. Looking ahead, Insikt Group anticipates ClickFix lures will become increasingly technically adaptive, incorporating more selective browser fingerprinting, while continuing to use infrastructure that can be built and dismantled quickly. In addition to technical refinements, Insikt Group predicts that the social engineering component will continue to evolve, leveraging new techniques to lure victims into executing malicious commands.*

We all know how annoyed I am with Microsoft. This entirely preventable vulnerability is now three years old and its use has been accelerating rapidly to the point that this family of readily blocked exploits now accounts for more than half of all security breaches. Yet Microsoft sits on their hands apparently doing nothing to mitigate this threat.

By comparison, our listener Jeff Adamson sent a note Friday with a link to a story over at [apple.gadgethacks.com](https://apple.gadgethacks.com) with the headline: *"macOS 26.4 adds Terminal paste prompt to block pastejacking"* PasteJacking is, of course, another name for ClickFix. Whenever a macOS user of Terminal attempts to paste a suspicious string into Terminal, an intercept dialog will be displayed to caution the user about the possible implications of what they are attempting to do. The dialog reads:



And the user is given the option "Don't Paste" or "Paste Anyway."

I don't suppose that Windows 10 users, who still compose one quarter of the Windows desktop market, will ever see Windows' behavior change, but it would sure be nice if Windows 11 users could have this simple exploit prevented by Microsoft caring, as Apple does, to fix it.

I also wanted to highlight the tips near the end of the Recorded Future article that talked about available mitigations of this. The measures of disabling the Windows Run dialog box via Group Policy Objects (GPO) and implementing PowerShell Constrained Language Mode (CLM) can't help the general Windows-using population, but within any enterprise I would jump on both of those immediately. Unless an enterprise's IT staff know that the Windows Run dialog is needed, it really should be disabled completely.

## AI Bots are flooding Reddit

Meanwhile, Reddit has been detecting a growing prevalence of AI posting bots on their site and may need to resort to various proof-of-humanity moving forward. PC Mag provided the details under their headline "*Reddit Could Soon Require Face ID to Prove You're Not a Bot*", writing:

*Reddit, like practically every other social media platform, has been struggling as of late with a deluge of bots and AI-generated content. In a study from last year, roughly **15%** of posts on the platform were found to be AI-generated. (More frequently than one out of every seven.)*

*Now, it may soon start experimenting with asking users for biometric data like Face ID or Touch ID, or other forms of passkey technology, to stem the tide of bots. In an interview with the TBN podcast, first spotted by Engadget, Reddit CEO Steve Huffman said this tech is the most "lightweight way" to ensure all users were human. Huffman indicated the platform may use decentralized third-party information providers to verify users' personal details.*

Oh boy. And we've recently been talking about all the uses to which residential proxies could be put. Bouncing AI-bot traffic through such proxies makes detecting and blocking them based upon IP address alone impossible. So, yeah, some sort of logon-time verification is needed. But we all know the potential downside of using any 3rd-party identity verification system. PC Mag continues:

*Steve Huffman told the podcast hosts: "Part of the promise to users is we don't want to know your name. But we do need to know that you're a person."*

*In 2026, bots are an existential risk to online platforms. Content aggregator Digg, which was in beta ahead of its comeback, was recently forced to pause operations and lay off staff in response to the horde of bots on its platform.*

*Meanwhile, the ability of bots to influence the discourse on Reddit has already been demonstrated. In April 2025, researchers from the University of Zurich secretly deployed AI-powered bots to influence debate in a subreddit called r/changemyview, with bots pretending to be a 'rape victim,' a 'Black man' who was opposed to the Black Lives Matter movement, and someone who 'works at a domestic violence shelter.'*

*Reddit founder Alexis Ohanian said his website using Face ID was not something that he had on his "bingo card," but argued that "something has got to be done about all the fake/botted content" in a recent post on X. However, the founder added the caveat that he doesn't "know how to sell face-scanning to redditors or even lurkers." Many Reddit users have already expressed grievances with the move, with one user saying: "Tell me you want to kill Reddit without telling me."*

*Reddit wouldn't be the first anonymous platform to start requesting users provide biometric data to prove who they are. For example, Discord earlier this year started to demand that some users provide face scans so its AI tool could determine if they were over 18, as part of efforts to keep minors off the platform.*

The pervasive use of residential proxies prevents any blocking by IP. And one wonders whether even using some form of Face ID or government issued ID could not also be spoofed. And as Reddit's CEO noted, offering anonymity has always been a big part of the Reddit experience. It's a true mess.

# LiteLLM

Let's start by answering the question "What is LiteLLM and why would we want it?" LiteLLM is backed by Ycombinator, and the LiteLLM page over at Ycombinator describes their project

*LiteLLM is an open-source LLM Gateway with 18K+ stars on GitHub (that's now 41,300) and trusted by companies like Rocket Money, Samsara, Lemonade, and Adobe. LiteLLM provides an open source Python SDK and Python FastAPI Server that allows calling 100+ LLM APIs (Bedrock, Azure, OpenAI, VertexAI, Cohere, Anthropic) in the OpenAI format. We have raised a \$1.6M seed round from Y Combinator, Gravity Fund and Pioneer Fund.*

Over at GitHub, the "About" paragraph reads:

*Python SDK, Proxy Server (AI Gateway) to call 100+ LLM APIs in OpenAI (or native) format, with cost tracking, guardrails, load balancing and logging. [Bedrock, Azure, OpenAI, VertexAI, Cohere, Anthropic, Sagemaker, HuggingFace, VLLM, NVIDIA NIM]*

The LiteLLM site itself largely echoes this and highlighting a couple of testimonials, it quotes David Leen, a Netflix staff software engineer, who says:

*LiteLLM has let my team provide the latest LLM models to our users usually within a day of them being released. Without LiteLLM this would be hours of work each time a new model is announced. It means we don't have to transform inputs and outputs across providers and has saved us months of work.*

And Mark Koltuk a Principal Architect of Generative AI platforms at Lemonade says:

*Our experience with LiteLLM and Langfuse at Lemonade has been outstanding. LiteLLM streamlines the complexities of managing multiple LLM models.*

So everyone gets the idea, right? With the general chaos that currently reigns across the AI domain, with new models appearing daily, pricing varying and today's top-dog latest & greatest model being tomorrow's "you're not still using that one are you?" – at the same time everyone is in a frenzied, frothing and frantic rush to mark out and claim some territory in whatever this is all eventually going to wind up being – essentially, with LLMs being the hottest fungible commercially tantalizing mystery humankind has ever created – the last thing anyone wants to be is locked into yesterday's less-glamorous now-underperforming or overpriced model.

To their credit, the guys at LiteLLM were quick to see a need and an opportunity. They created a universal large language model API translator that allows front-end developers to code to a single fixed model, the one originally developed by OpenAI, and the LiteLLM proxy shim would allow any other model to be swapped in behind it on the backend without needing to recode any of the front end. The famous Code Academy folks have a page titled "What is LiteLLM and How to Use it" where they write:

*LiteLLM is an open-source Python library that acts as a unified interface for Large Language Models (LLMs). It allows us to connect with multiple AI providers such as OpenAI, Anthropic,*

*Google Gemini, Mistral, Cohere, and even local models through Ollama – all using a single, standardized API. Working with multiple LLMs results in juggling different API formats, authentication methods, and SDKs. This usually requires code rewrites, new dependencies, and manual adjustments. LiteLLM resolves this by acting as a bridge between the application and major LLM providers, letting you manage requests, responses, and errors consistently.*

*At its core, LiteLLM does two main things:*

- Normalizes APIs across providers: It takes a standard input from our code and adapts it automatically so that it matches the requirements of the target model.*
- Acts as a gateway or proxy: It can route and manage requests, track usage, handle errors, and centralize configuration, making multi-model workflows easier to maintain.*

*In short, LiteLLM consolidates the complexity of multiple LLMs into a single, manageable interface.*

The LiteLLM guys have been working on this since the winter of 2023 and as you might imagine, the challenge of supporting an exploding number of individual varying and evolving AI LLMs requires a great deal of neverending work. But that's the path they have taken and until recently it had been pretty smooth sailing.

So what happened last week? Let's start with TechCrunch's overview, then we'll dig a bit deeper. TechCrunch wrote:

*This week, some really atrocious malware was discovered in an open source project developed by Y Combinator graduate LiteLLM.*

*LiteLLM gives developers easy access to hundreds of AI models and provides features like spend management. It's a breakout hit, downloaded as often as 3.4 million times per day, according to Snyk, one of the many security researchers monitoring the incident. The project had 40K stars on GitHub and thousands of forks (those who used it as a base to alter and make it their own).*

*The malware was discovered, documented, and disclosed by research scientist Callum McMahon of FutureSearch, a company offering AI agents for web research. The malware slipped in through a "dependency," meaning other open source software that LiteLLM relied upon. It then stole the log-in credentials of everything it touched. With those credentials, the malware gained access to more open source packages and accounts to harvest more credentials, and so on.*

*The malware caused McMahon's machine to shut down after he downloaded LiteLLM. That event prompted him to investigate and discover it. Ironically, a bug in the malware caused his machine to blow up. Because that bit of nasty code was so sloppily designed, he (as well as famed AI researcher Andrej Karpathy) concluded it must have been vibe coded.*

*The LiteLLM developers have been working nonstop this week to rectify the situation, and the good news is that it was caught relatively fast, likely within hours.*

Last Tuesday, as mentioned by TechCrunch, a developer named Callum McMahon with FutureSearch explained what he had discovered and how. At the end of a separate but related

posting, Callum explained their use of LiteLLM. Knowing what we now know about LiteLLM, it's exactly what we would expect. He wrote: *"We use LiteLLM to let us use models from a wide range of providers, letting us strike the best balance between quality, speed and cost."* So here's what happened. Callum's posting was titled *"No Prompt Injection Required"*, writing:

*Earlier today I got taken out by malware on my local machine. After identifying the malicious payload, I reported it directly to the PyPI security team – who credited our report and quarantined the package – as well as to the LiteLLM maintainers. I wrote a blog post that became the primary source cited by The Register, Hacker News, Snyk, and others. The play-by-play is pretty interesting when looking back.*

*It started with my machine stuttering hard, something that really shouldn't be happening on a 48GB Mac. "htop" took 10s of seconds to load, the CPU was pegged at 100%, all signs I'll be working on my local env for a while.... After failing to software reset my Mac, I took a final picture for evidence and hard reset.*

*So far, the clues had been Cursor asking me for network access right as the machine was freezing up, the process list showing a bunch of python commands all exec-ing a base64 encoded string, and 11k processes running. I set ulimit to 16k for ML workloads so this was partly expected.*

*On restart, I asked Claude to investigate. After going down a rabbit-hole on the wrong shutdown due to my force-shutdown not generating the expected logs, I presented it with the start of the base64 string. Just enough to decode "import subprocess \n import tempfile" before the remaining text went offscreen. Claude then became adamant that this was its own doing, the standard Claude Code way of running bash commands to escape control characters. Despite the many bugs I've encountered with that CLI, I wasn't buying this explanation. Further Claude Code probing eventually found the offending cause, the rogue package buried within my uv cache, something I would have **never** found on my own!*

*Two minutes later, it had reproduced the entire malware trigger within a local container to double check its claims this time. And a further two minutes later I had a blog posted on our site detailing the specifics of the malware to share as a warning to others. Claude even proactively suggested the emails of both the PyPI security team who were quick to quarantine the package, as well as the LiteLLM maintainers. So, what actually happened?*

I'll interrupt here to note that Collum is about to start referring to MCPs. The ModelContextProtocol website explains:

*MCP (Model Context Protocol) is an open-source standard for connecting AI applications to external systems. Using MCP, AI applications like Claude or ChatGPT can connect to data sources (e.g. local files, databases), tools (e.g. search engines, calculators) and workflows (e.g. specialized prompts)—enabling them to access key information and perform tasks. Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect electronic devices, MCP provides a standardized way to connect AI applications to external systems.*

Armed with only that much understanding, what Collum explains can make sense and it's not necessary to deeply understand it to get the sense of what happened. Collum writes:

*The root cause was mundane. MCP clients like Cursor, Claude Code and others are using (local) MCP servers via some "executor" tool such as uvx for Python or npx for Node.js. When you run an MCP via uvx, it automatically downloads dependencies of that MCP and runs the given command. Unfortunately, our (mostly deprecated) MCP server had an unpinned dependency of a LiteLLM package. When my Cursor IDE tried to autoloading the MCP server, uvx stepped in to download that latest LiteLLM version, which was malware uploaded to PyPI by hackers just minutes earlier. The seamless ergonomics of uvx meant I became one of the lucky beta testers of the freshly released malware.*

Okay. So, in other words, exactly the sort of textbook-classic supply-chain attack we've discussed so many times in the past. In this case it wasn't a dependency such as a library that would be downloaded, compiled and linked into a result. It was a working piece of tooling – the LiteLLM package. And by being "unpinned" Collum's dependent packages were not saying "we want this exact version". So the default behavior is to grab a copy of the latest and greatest. And in this case that "latest and greatest" had been deliberately compromised by bad guys.

Collum continues:

*A sloppy, likely vibe-coded mistake in the actual malware implementation led it to turn into a fork bomb. It installs a file called **litellm\_init.pth** in site-packages. Python automatically executes .pth files on every interpreter startup. The first thing it does is: That child Python process also triggers **litellm\_init.pth**, since it's still in site-packages, which spawns another child, which spawns another. Thus leading to the only sign I would have noticed that the malware was running.*

*As Andrej Karpathy pointed out on X, without this error it would have gone unnoticed for much much longer. The malware's own poor quality is what made it visible and discoverable.*

*O what's the takeaway? We've since moved to a remote MCP architecture. The server doesn't run on the user's machine anymore, which collapses this entire attack surface. No local code execution means a poisoned dependency can't touch your filesystem or request network access from your OS, and it's much more localized to one, audited version that we have under control. However, sometimes you can't reliably do that, there are advantages and disadvantages of local vs. remote MCP servers, and in that case you still need to do what you can to mitigate this risk.*

*I don't think there is anything new to say here, it's the same thing we have been doing everywhere else to keep us safe: reduce the attack surface, pin your dependencies, or even better: use lock files with checksums, audit packages before upgrading, and when Claude tells you everything is fine, maybe ask it twice.*

*We analyzed the blast radius of this attack. 47,000 downloads in 46 minutes, 88% of dependent packages unprotected.*

So now let's take a closer look at the malware itself. For that, we turn to TrendMicro who titled their coverage of this "*Your AI Gateway Was a Backdoor: Inside the LiteLLM Supply Chain Compromise*", which they tease with the follow-on: "*TeamPCP orchestrated one of the most sophisticated multi-ecosystem supply chain campaigns publicly documented to date. It cascaded through developer tooling to compromise LiteLLM and exposed how AI proxy services that concentrate API keys and cloud credentials become high-value collateral when supply chain attacks compromise upstream dependencies.*"

They lead with three key takeaways:

- *LiteLLM, a widely-used AI proxy package, was compromised on PyPI, with two of its versions containing malicious code. These LiteLLM versions deployed a three-stage payload: credential harvesting, Kubernetes lateral movement, and persistent backdoor for remote code execution. Sensitive data from cloud platforms, SSH keys, and Kubernetes clusters were targeted and encrypted before exfiltration.*
- *The LiteLLM incident was part of a broader campaign by the criminal group TeamPCP, which has demonstrated deep understanding of Python execution models, adapting their attack rapidly for stealth and persistence.*
- *TeamPCP has previously compromised security tools like Trivy and Checkmarx KICS to steal credentials and propagate malicious payloads. Attackers leveraged compromised CI/CD pipelines and security scanners to escalate privileges and publish trojanized packages.*

So, here's what more TrendMicro has learned:

*On March 24, production systems running LiteLLM started dying, and engineers saw runaway processes, CPU pegged at 100%, containers killed by out-of-memory (OOM) errors. The stack traces pointed to the LiteLLM package, a popular Python package downloaded **3.4 million times per day** that serves as a unified gateway to multiple LLM providers, was compromised on PyPI. Upon analysis, it was found that versions 1.82.7 and 1.82.8 contained malicious code that stole cloud credentials, SSH keys, and Kubernetes secrets.*

*The malicious versions deployed a three-stage payload: a credential harvester targeting over 50 categories of secrets, a Kubernetes lateral movement toolkit capable of compromising entire clusters, and a persistent backdoor providing ongoing remote code execution.*

*This compromise was **not** an isolated event. It was the latest link in a cascading supply chain campaign by a threat actor tracked as TeamPCP. This post traces the cascade from its origin, the open-source vulnerability scanner Trivy, and then presents our technical analysis of the LiteLLM payload.*

*TeamPCP orchestrated one of the most sophisticated multi-ecosystem supply chain campaigns publicly documented to date.*

*The campaign spanned PyPI, npm, Docker Hub, GitHub Actions, and OpenVSX in a single coordinated operation. While it did not specifically target AI infrastructure, the campaign's cascade through developer tooling caught LiteLLM within its blast radius and exposed how AI proxy services that concentrate API keys and cloud credentials become high-value collateral when supply chain attacks compromise upstream dependencies.*

*Key sections of this blog entry include a technical analysis of the malicious multi-stage payload and its impact on AI environments, a timeline and operational review of TeamPCP's campaign, and a deep dive into how security tools themselves became attack vectors. TrendAI™ Research's analysis into the LiteLLM compromise also covers attribution challenges, gaps in public threat intelligence, and actionable defense strategies. Detailed indicators of compromise and MITRE ATT&CK mappings have been provided, but for an even more comprehensive understanding of this security incident, reach out to TrendAI™ Research for the full technical report.*

So that's much deeper than we need to dive. But what they uncovered and reported about the root source of the vulnerability was interesting. Under "*How your security scanner can become the attack vector*" they wrote:

*Trivy is an open-source vulnerability scanner developed by Aqua Security. It scans container images, filesystems, and infrastructure-as-code for security vulnerabilities, and it is integrated into the CI/CD pipelines of thousands of software projects via the trivy-action GitHub Action.*

*Security scanners are uniquely dangerous supply chain targets. By design, they require broad read access to the environments they scan, including environment variables, configuration files, and runner memory. When a scanner is compromised, it becomes a credential harvesting platform with legitimate access to secrets.*

*In late February 2026, an actor operating under the handle MegaGame10418 exploited a misconfigured pull\_request\_target workflow in Trivy's CI (Continuous Integration) to exfiltrate the aqua-bot Personal Access Token. Aqua Security disclosed the incident on March 1 and initiated credential rotation. However, according to Aqua's own post-incident analysis, the rotation "wasn't atomic and attackers may have been privy to refreshed tokens."*

That's an important point, so let me explain. We've talked about the concept of so-called atomic operations. The name obviously comes from the word "atom" and is meant to imply that it cannot be further divided into smaller pieces. Molecules, being collections of atoms, are divisible. But not so the atom.

To clearly illustrate the occasional need for atomic operations, say that a computer program needed to count up to a certain number but no more. If the program was single-threaded, meaning that it only ever had one thing going on at once, this would be easy to do. The program would read the value of the thing being counted. If it was already at its upper count limit the program would increment it to its next value. No sweat, right?

But now imagine what happens if there's a lot more going on in the program with multiple simultaneous threads of execution, perhaps because the CPU has multiple cores. In this environment, there's a chance that both CPUs would wish to increase the count at the same time. So they would both be executing the same code at the same time. They would both read the counter's value, they would both see that it had not yet reached its limit, so they would both increment it, thus increasing its initial value by two. But if the counter had been previously sitting at one below its limit, the increase by two would move it up past that limit.

These sorts of so-called "race conditions" have historically been the source of a great many bugs that never happen while you're watching but somehow always occur when you're on stage demonstrating the latest and greatest.

In our example, the "test the value and maybe increment it" would need to be "atomic" so that the testing and the incrementing could not be broken apart and performed separately. That operation could only be done by one processor or execution thread at a time. One processor would be briefly stalled until the other processor had finished with the atomic operation. At that point the second processor would see the result after the first processor was finished so that no double-incrementing could occur.

Okay, we left off with TrendMicro noting: *Aqua Security disclosed the incident on March 1 and initiated credential rotation. However, according to Aqua's own post-incident analysis, the rotation "wasn't atomic and attackers may have been privy to refreshed tokens."*

TrendMicro then continues:

*That gap proved decisive. On March 19 at 17:43 UTC, TeamPCP used still-valid credentials to force-push 76 of 77 release tags in the trivy-action repository, and all seven tags in setup-trivy, to malicious commits containing a multi-stage credential stealer. The malicious code scraped the Runner.Worker process memory for secrets, harvested cloud credentials and SSH keys from the filesystem, encrypted the bundle using AES-256-CBC with an RSA-4096 public key, and exfiltrated it to a typosquatted domain (scan.aquasecurity.org, resolving to 45.148.10.212). According to analysis by CrowdStrike, the legitimate Trivy scan still ran afterward, producing normal output — leaving no visible indication of compromise.*

In other words, because Aqua Security was logistically unable to rotate every single credential at once when no one was actively logged on, the bad guys were able to maintain their corrupting persistence. TrendMicro finished this portion of their write-up, writing:

*This is the meta-attack: a security scanner, the tool defenders rely on to catch supply chain compromises, became the entry point for a supply chain compromise. The Trivy compromise in GitHub Actions gave the attacker the keys to publish arbitrary versions of LiteLLM to PyPI. Everything that followed was exploitation of that initial foothold.*

*The lesson is uncomfortable but critical; your CI/CD security tooling has the same access as your deployment tooling. If it's compromised, everything downstream is exposed.*

So what we see is that the enabling of this attack on LiteLLM had nothing to do with AI. The attackers found a way into the distribution of an extremely popular code module that happened to be about AI and was being downloaded 3.4 million times per day. That said, the malware did thereby gain access to the AI access credentials that had been entrusted to LiteLLM.

After providing a fully detailed forensic analysis of this malware campaign, TrendMicro concluded with a summary and recommendations. They wrote:

*As AI/ML tooling proliferates across enterprise CI/CD pipelines, the attack surface expands with it. The tools that developers install to interact with AI systems, proxy gateways, model routers, experiment trackers, and inference servers handle high-value secrets by design. Supply chain attacks against these tools inherit the trust and access of the AI infrastructure itself.*

Again, AI is not to blame here. It's really just a case of "the more tools you're using the more exposure there will be when any one of them might be compromised. They continued:

*The malicious payload analyzed in this report is a direct exploitation of the systemic secret management failures extensively documented in prior TrendAI™ Research. As previously described, developers have adopted .env files so profusely that they have forgotten their sensitivity, leaving them exposed, and threat actors are actively scanning for exactly these files. The harvester analyzed here operationalizes that attack surface at scale: it performs exhaustive filesystem walks targeting .env, .env.local, .env.production, and .env.staging files across up to six directory levels, while simultaneously extracting AWS credentials, cloud provider tokens, Kubernetes service account secrets, CI/CD pipeline configurations, and database connection strings; the same categories of secrets TrendAI™ Research previously*

*identified as most commonly stored in plaintext inside .env files.*

And they finish by offering some well-reasoned security recommendations:

*This case highlights the risk of building an entire ecosystem on top of fragile trust. The LiteLLM hack is just the latest example of attackers exploiting the reliance on open-source registries and poor secret hygiene. Security is not an afterthought you can outsource entirely to a vulnerability scanner.*

So, the apparently highly skilled attackers appear to have been in a bit of a hurry. This led them to deploy otherwise very potent malware containing a flaw that immediately caused the malware to draw attention to itself. As a result the infection was almost immediately spotted and stopped.

Had the bad guys not made that mistake, at a download rate of 3.4 million copies of the infected LiteLLM per day the damage that was all set and engineered to occur likely would have, and the resulting mess would have been far worse. In the truest sense we have dodged another bullet.

There can be no question that the entire industry has built an ecosystem upon which it has become dependent whose security guarantees are truly fragile. We're essentially hoping for the best because the goodies are just too enticing for us to resist. Phrased another way, the cost to us today of deploying truly secure solutions prices them out of reach rendering them impractical. So we knowingly and deliberately create dependencies upon sprawling packages over which we have no oversight or control.

All we can really do at this point is hope that our luck holds.

