



## LiteLLM

**Description:** Will California require Linux to verify its users' age? Apple's iOS 26.4 requires UK users to prove their age. Russia chooses to use homegrown 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.

High quality (64 kbps) mp3 audio file URL: <http://media.GRC.com/sn/SN-1072.mp3>

Quarter size (16 kbps) mp3 audio file URL: <http://media.GRC.com/sn/sn-1072-lq.mp3>

---

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with a show about something that should send a chill into the heart of every coder, the nightmare PyPI exploit LiteLLM. We'll do a kind of deep dive on what happened, how it happened, and what we can do to prevent it in the future. Plus we'll talk about age verification on Linux, a good move from Apple on the ClickFix vulnerability, and is quantum computing moving closer? Steve has thoughts, next on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 1072, recorded Tuesday, March 31st, 2026: LiteLLM.

It's time for Security Now!. I know you waited all week for Tuesday. Best day of the week.

**Steve Gibson:** Leo's back. Leo's back.

**Leo:** Well, I am back, but so is Mr. Gibson. Steve Gibson's here.

**Steve:** Mikah did a great job last week...

**Leo:** Thank you, Mikah, for filling in for me.

**Steve:** ...holding the fort down.

**Leo:** I was at RSAC, the big security conference in San Francisco. I ran into a friend of yours, Marcus Hutchins, the hacker.

**Steve:** Oh.

**Leo:** In fact, I kind of relived old times because I said, yeah, we were - when, what was it, WannaCry that he...

**Steve:** We were following along with what he was doing.

**Leo:** And then he left Black Hat in Vegas and got picked up by the feds before he could board his plane.

**Steve:** In the airport.

**Leo:** In the airport. And was held. And because of his youthful indiscretions, not because - they didn't recognize the valuable contributions he'd made as an adult. Anyway, we went through that and everything. And I guess he was there because there's a new documentary being made called "Midnight in the War Room," a documentary of cyberwarfare. And so we went over there, and then he was sitting there signing hats. So I got a Marcus Hutchins hat. And it says, as if I didn't know who he was, it says Cybersecurity Guru on the side. By the way, we made a wonderful RSAC about an hour long kind of tour through RSAC. We talked to, oh, a dozen people or so, something like that. And you can watch that on the TWiT feed. It's on YouTube: "Securing the Agentic Era, RSAC 2026."

**Steve:** That is the challenge; isn't it. Those little pesky agents, they get up to all kinds of things.

**Leo:** Well, I was really you know, I specifically wanted to talk to people who are using AI defensively. And there are a number of companies doing it. There was one company called Aikido, AIkido; right? And we were talking about which models they were using and how they were getting the models to do the job. And the guy said, at one point, he said, you know, we had to - the best thing we found was to tell the models we will sue you if you don't find the flaw. And it worked. It scared them. It's like, we are living in a weird time with AI.

**Steve:** We're going to need AI HR before long.

**Leo:** Absolutely.

**Steve:** It's like, don't you mistreat, you know, you've got give your AI a little, literally, time to cool off.

**Leo:** Yes. Exactly.

**Steve:** Let those chips cool.

**Leo:** Oh, my goodness.

**Steve:** Let the fans take the heat off of them.

**Leo:** Well, and they do, the Claude Code was accidentally leaked from Anthropic today or yesterday. And they do have instructions in there to say be more personable. You know, be a little more sassy. They're telling it to have some personality because it makes it more sticky. If you go to [YouTube.com/twit](https://www.youtube.com/watch?v=twit), there it is, "RSAC 2026: Securing the Agentic Era." And you can hear me talk to Marcus Hutchins and a bunch of other people in that video, if you want to play it.

**Steve:** Cool.

**Leo:** Now let's talk about, though, what's going on today on Security Now!.

**Steve:** So probably the biggest news, the most, oh, a lot has happened in the last week. I actually had a couple pieces of email saying, hey, I thought you were going to talk about this or that, like the foreign routers being outlawed and, like, what?

**Leo:** I went up to Ubiquiti at RSAC. And I said, I want to talk to you about that. And they said no. No, we're not giving any interviews. They wouldn't talk to me.

**Steve:** Yeah. And I looked around, and I couldn't find any, like, definitive dates. I even looked at the official government document. And it's like, okay, like, when? Or where? Or, like, what? And, you know, my takeaway was, for most of our listeners, there are really good alternatives like running OPNsense on a little ARM box or a little...

**Leo:** I think that's what a lot of people are saying. I'll just run my own; right?

**Steve:** Exactly. You really don't need to get something from ASUS any longer. And arguably you get a much stronger and more capable result. But anyway, I want to talk about this big event with LiteLLM. It's such a perfect glimpse into a supply channel attack. So we'll sort of use that as our armature for discussing just in general the problem we're having with our supply chain because, boy, are we seeing an acceleration of that. You know, I think probably we first really touched on it with the Log4j exploit, which scared the entire industry. And it turned out to be a nothing burger because it was difficult to do. And what we've learned is that it's the easy attacks that a much larger percentage, much greater percentage of the population of hackers are able to jump on.

But overall, I mean, and we've been talking about, you know, all of the infections found in NPM and PyPI. So that's where we're going to focus on this Episode 1072 for the last day of March, March 31st. But we're also going to talk about, oh, the other real hot button, Leo, is this California looney tune law that says that operating system platforms must enforce age verification. And of course every Linux person's head exploded. It's like, wait a minute. I mean, the reason they are a Linux person is so that they can't have the government or anybody telling them what to do. So, wow. A lot happened there. We'll talk about that.

We've also got some new behavior from iOS 26.4, which Apple just moved their whole ecosystem to, which requires, is requiring UK users to prove their age proactively. And it turns, well, I don't want to - I have to calm myself down because we'll get there. Also Russia in this continuing move to just apparently withdraw from the world and society and good luck with that, has chosen to use a homegrown 5G encryption for their future mobile network, which won't be compatible with anything else, or anybody's phones. So, okay.

Oh, there was a great story of a Ukraine drone maker who was aware that a Russian spy agency was installing a bad spying thermostat in their facility and what happened with that. We've got Google moving forward - forward or backward? - closer to what they call the "Q Day," the day when we actually think that we have to worry about quantum computing. They've moved it into this decade, into 2029. So we're going to look at that.

Also at RSA, where you were, the UK's CEO of the NCSC, you know, their big cybersecurity agency, stood up onstage and warned about the danger of vibe-coded SaaS, you know, Software-as-a-Service replacements. We're going to touch on that. We've got more information about nasty ClickFix campaigns, which continue to proliferate. And many people emailed me to say, hey, Apple took your advice. Well, they didn't take my advice, but they did the right thing, which Microsoft refuses to do.

**Leo:** We talked about that, yeah.

**Steve:** We've also got the news that more than one in seven Reddit postings are now being posted by AI bots. And the CEO of Reddit is not happy, and he doesn't know what to do except something that Reddit users really don't want to do. And then we're going to talk a little bit about, well, a lot about what was going on behind this LiteLLM disaster that was averted, but only because the people who coded it, the malware, were apparently in too big a hurry.

**Leo:** It was vibe coded.

**Steve:** They made a mistake, yup, they made a mistake that allowed it to reveal itself almost immediately, I mean, which was a good thing because, again, this absolutely constitutes dodging another bullet. And how many are we going to dodge before we get hit by one?

**Leo:** I'm not even sure how much was dodged. I mean, we don't know yet how many people - we know 47,000 people downloaded it. We don't know how many of them got bit.

**Steve:** Well, which is not to say that nobody got hurt.

**Leo:** Right, could have been worse.

**Steve:** But when you're downloading three point some million per day, it could have exploded, yeah.

**Leo:** Right. Well, and there was a new one this morning in Axios, which is an NPM library, compromise. This is not the - it's just the beginning. This is not...

**Steve:** Well, and we'll be talking about why because there's a, as usual, takeaway that we try to find here. And to my way of thinking, this is the tradeoff we are still making in preferring convenience over security.

**Leo:** Yeah.

**Steve:** It's like we're hoping for the best. And so far, you know, maybe AI will change it. We're not sure.

**Leo:** Well, that was the interesting thing at RSAC is the number of companies that were proposing AI-based defense. We're using AI agents for defense. It was very, you know, it was probably the number one topic at RSAC, so it was very interesting, yeah.

Well, we're going to get to the Picture of the Week, something you should avoid with wet-nosed doggies, apparently. I only saw the caption. But that's coming up in just a little bit.

**Steve:** So...

**Leo:** Picture time?

**Steve:** Yes. I gave this, as you noted, I gave this picture the caption "This solution is not recommended if your dog has a wet nose."

**Leo:** Okay. So I'm thinking it's going to have something to do with electricity here. Let me just scroll up, and we can discover it together. There's a nail. Uh-oh. Oh, boy. That's, okay, so this is an interesting solution if you have the wrong country's power.

**Steve:** Yes. And Leo, really, all of these solutions are interesting. Benito and I were talking about this before we began recording. You know, there's that one with the two nail clippers that I thought was particularly inventive.

**Leo:** Yeah, yeah, yeah.

**Steve:** But these, I think if we were to stand back and look at the 20-year-plus history of the podcast, it would be people who seem to have the wrong plug for their outlet and the mystery of fence gates standing alone in the middle of fields seem to be...

**Leo:** Very popular.

**Steve:** ...two of the overriding themes of our Pictures of the Week. So for those who are listening and are not seeing, we've got a - the individual here is trying to plug a European-style AC plug that's got the round pins into, well, he would like to plug it in apparently in the U.S., or somewhere where we have the parallel straight slots.

**Leo:** I've got to tell you that's the worst-looking power strip I've ever seen.

**Steve:** Everything about this. I mean, and when you use the term "rusty nail," you're normally not really literal. But here...

**Leo:** This is literal.

**Steve:** I don't even know how he's getting a connection, there's so much rust on these nails. Because of course rust is an oxide which is an insulator. But anyway, so we basically sort of have a Jacob's Ladder made with two nails stuck into the slots of this power strip. And then pushed down between the nails so that they're sort of splayed apart, thus creating the Jacob's Ladder effect, is this round plugged European-style plug which just sort of hovers there. And my point was, if you have a curious dog that likes to go around sniffing in the corners, I don't think even its nose would need to be wet. It would get a very rude surprise if it were to stick its nose across the power.

**Leo:** And how did they get the nails in without shocking themselves? They must have had rubber gloves or something. I mean, this is insane.

**Steve:** It's nuts, yes. Hopefully the outlet strip has a switch that we don't see offscreen.

**Leo:** Oh, there you go, of course. There you go.

**Steve:** Or just could not be plugged into its normal American-style outlet where it gets its power. Anyway, thank you again, listeners for another entertaining Picture of the Week. Always appreciate them.

Okay. So our first news of the week was inspired by a question I actually received from a listener. It relates to much of our recent discussions about Internet age verification and specifically to its recent escalation, which we've been seeing everywhere, to include, as just happened, operating system platforms themselves. 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." He said: "Since FreeDOS is the OS distributed with SpinRite, I was wondering how this would affect you when the new law takes effect. Thanks, Fred."

Okay. So the good news is SpinRite is not age-restricted content, so I don't think we have a problem, even if we were going to have a problem, which I don't think we would. But his question refers to California's Assembly Bill 1043. And it's unclear to me why this issue suddenly and recently popped up on everyone's radar. But the Internet is currently buzzing about it, and our listeners have been sending their questions and opinions to me, which I appreciate. The Bill in question was approved by California's governor back on October 13th of last year, of 2025; and it doesn't take effect until the start of next year, January 1st of 2027.

So why all of the sudden awareness of it? Because, I mean, it was a while ago. And so I did some looking around over the past month. The only thing I could find was that the very popular and respected and widely read Tom's Hardware site did post an article about this on March 1st, which sort of seems maybe to have been the catalyst for everyone going, what? What are you talking about? So Tom's Hardware's headline was "California introduces age verification law for all operating systems, including Linux and SteamOS, user age verified during OS account setup."

Okay. So, you know, 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, the way Windows does, Mac, Android, and iOS, they reasoned, Linux users reasoned there would be no way for that to happen. Right? Right?

So, okay. Since California legislators have also recently proposed, as we talked about, requiring all 3D printers to somehow magically identify and refuse to print any component that might be part of a handgun - no one knows how that could be made possible either. Unfortunately our, and I say "our" because Leo and I are both residents of California, our legislators here in California do seem to be having fun asking for things they cannot realistically have. Not that that's stopping them from, you know, asking.

So, okay. First let's first step back and take a look at what this legislation is because it does exist. It was signed into law on October 13th, and it is coming into effect on January 1st. That's all happening. The section heading in this bill is "Age verification signals: Software applications and online services."

And the section's overview, just the overview of the detailed, you know, point-by-point 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 and, you know, 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."

And of course, Leo, the other thing that happened just in the last week was Meta and Google with YouTube losing those major cases. And there was also one in Arizona, I think, which is beginning to hold these, you know, big tech accountable for the design practices in their applications, which do exactly what this California Age-Appropriate Design Code Act says they shouldn't do.

So they wrote: "This bill, beginning January 1, 2027," which did get signed, you know, on October 13th, they said, "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 (API) 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."

Okay. So that's as much as I'm going to quote from that. So while it's true that details matter, I'll first note that this is, like what this bill is asking for, is what we've been suggesting Apple with iOS and Google with Android should both somehow manage to provide. And the way this should be done, at least for the use case of smartphones, is beginning to take shape. We're beginning to see this manifested in Apple's apparently reluctant incremental movement on this front.

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, and I'll say optionally should be able to, but like, so that the point is the platform would provide the capability, but it should be at their discretion. From that point 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 operating system, the user, the kid, may cause the user's operating system to display a clear - or, I'm sorry, the age-restricted content provider may cause the user's operating system to display a clear and uniform pop-up asking for an age bracket to be provided.

If the user wishes to - again, not automatically, not unless they set it that way, but if they want control, if they choose to, they may then allow the operating system to inform the requested application on their behalf whether its user is under 13, between 13 and 15, between 16 and 18, or over 18. Those are the brackets California specifies and which the world seems to sort of be settling on. If the user declines to provide their 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 probably not deliver this age-restricted content.

So what seems right about this is that this solution places the handling and responsibility of their young child's age into the parents' hands, where it should be. Not the government. Not the OS. Not the platform provider. The platform provider provides the capability to configure the device to do this if the parent or guardian should so choose. And all it requires of Google and Apple, and Apple's, like, they're both almost there now, 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 - again, if it's been configured to do that - to any entity that inquires. And as I said, both companies, apparently reluctantly, have been moving incrementally in this direction.

So at this point I cannot, given what's happening on the legal side in local and national governments, I can't find any sympathy for someone who complains that this, like what I described, would represent an invasion of an online user's absolute privacy, which is what we see. There's a lot of that on the 'Net. Opening the front door of your 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. You know, and with the vast resources that are now online, there's a lot of stuff that needs arguably young people, you know, parents should have the right to decide if that's something that they want their children to have access to.

So as a society we're now working to more fully incorporate all of the many facets of the Internet, which are many now, into our daily lives. So to do that responsibly means that a user's age, although it hasn't previously been, it's going to have to be taken into account moving forward.

Okay. So, but what happens when we leave the mostly well and clearly and cleanly defined realm of personal-use smartphones, you know, which have per-user accounts?

Things become a lot less clear and clean, and I argue, I mean, I agree with everyone who's upset about California and what this means for Linux that we're stepping into a huge mess. So here's what Tom's Hardware wrote, which may have been, as I thought, the catalyst for this recent upsurge of interest and outrage.

They said: "California's Digital Age Assurance Act, Assembly Bill 1043, signed by Governor Gavin Newsom in October 2025, requires every operating system provider in California" - and I don't even know, like, if Linux has an operating system provider. Right?

**Leo:** Well, I mean, Ubuntu is, for instance, that's a company that makes a distro. It would have to be by distro.

**Steve:** Yeah.

**Leo:** Also Android operating systems like Graphene, Graphene's already said we're not going to do this.

**Steve:** Well, and did you see that it got stuck into System D and then got yanked?

**Leo:** Yeah, yeah.

**Steve:** So, I mean...

**Leo:** Nobody wants this. But there's no enforcement mechanism. There's no - it's not even clear how they would know. And it doesn't require ID. Right?

**Steve:** Correct.

**Leo:** It just says you say how old you are.

**Steve:** Correct.

**Leo:** So what good is that?

**Steve:** Correct.

**Leo:** It's silly.

**Steve:** Yeah.

**Leo:** Just nonsense.

**Steve:** Yeah. So they said, Tom's Hardware said: "The law's broad definition of '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," well, that means Smart TVs, too; right? So, like, okay, you're going to tell your TV how old the viewer is?"

**Leo:** Oh, yeah, right. Tizen and WebOS, all the TV operating systems.

**Steve:** Yeah.

**Leo:** Wow.

**Steve:** And so Tom's said: "...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' (API) that categorizes users into four brackets." Those are the ones that we mentioned. Developers who receive the signal "deemed to have actual knowledge." So if the OS makes a claim, then they're kind of off the hook. Well, the OS said this was the age of the user. So they are at that point deemed to have actual knowledge of their users' age range under the law, which shifts legal liability for age-appropriate content decisions onto them. So that says, if they've been told, then they must act accordingly.

And Tom's wrote: "Penalties for non-compliance run up to \$2,500 per affected child for negligent violations and \$7,500 for intentional violations." And it's important, though, that this is all enforced by the California Attorney General. And that was a point made elsewhere, that it means random groups can't sue on behalf, you know, under this law.

**Leo:** That's good.

**Steve:** It's only the California Attorney General.

**Leo:** That gives them a lot of discretion about who they pursue.

**Steve:** Exactly. So Tom's said: "The law does not require," as you said, Leo, "photo ID uploads or facial recognition, with users instead simply self-reporting their age." What?

**Leo:** 99.

**Steve:** So he says: "This sets AB 1043 apart from similar laws passed in Texas and Utah that require" - and we've seen this one, we've talked about this - "'commercially reasonable' [whatever that means] verification methods, such as government-issued ID checks. California 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."

---

**Leo:** That's kind of like a, "yeah, sure, why not" vote.

**Steve:** Yeah, it was like...

**Leo:** Okay.

**Steve:** Like, this is an easy one.

**Leo:** Yeah.

**Steve:** However, Gavin was a little circumspect. Tom's wrote: "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" - right, a streaming service on a Smart TV - "streaming service and game developers about 'complexities such as multi-user accounts shared by a family member and user profiles utilized across multiple devices.'" In other words, you know, we're talking about the same thing, a version of the same thing we've been talking about with networks since the beginning of the podcast, authentication. On one hand, there's identity authentication. Now we're facing age authentication. And it's just as messy because you're remote. And these are just not easy problems to solve.

So Tom's said: "Whether amendments will materialize before January 2027 remains to be seen. Enforcement against Linux distributions, however, is likely to be problematic," wrote Tom's. "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." It's easy to do. "So a more realistic outcome for non-compliant distros is a disclaimer that the software is not intended for use in California." And maybe eventually anywhere.

**Leo:** Really, do not use this anywhere.

**Steve:** Really. You can't use this on Earth. So good luck. So I spent some time reading everything I could because this is, again, this cross-network authentication of anything is hard. I found a posting made by the Reason Foundation a few months ago that it's worth sharing. 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 also created.

So they wrote: "California Governor Gavin Newsom signed the Digital Age Assurance Act (Assembly Bill 1043) into law on October 13th, 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."

They wrote: "California's AB 1043 mandates" - and so, yeah, we got the four age-bracket thing. So they said: "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 - I think it's in the summer - while Texas has been temporarily blocked by a federal judge on constitutional grounds.

"Two federal bills, the App Store Accountability Act introduced by Senator Mike Lee (R-Utah), and the Parents Over Platforms Act introduced by Representative Jake Auchincloss, who is a Massachusetts Democratic Senator, included 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." Which that's a problem; right? So what they're saying here is that, by having state and even federal laws which say you must do the best job you can, unfortunately, the best job you can is very intrusive of privacy.

So they said: "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." And again, it's like, why are they not deleting this the moment they've determined someone's age? But okay.

"In October, global messaging platform Discord, as we know, 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 four 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, [California Assembly Bill] 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." That is, all it ever discloses is brackets. "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."

**Leo:** Boy, I really like that.

**Steve:** Yes.

**Leo:** That's really good.

**Steve:** I know, it sounds like you, Leo. That's exactly the point that you've often been making here. They said: "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."

And they finish, writing: "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."

So 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. Never its date of birth. Just where you are in those brackets, you know, which bracket you're in.

The idea would be that the various operating systems would implement a simple API. And, you know, iOS is there. Android is there. I mean, they're like right there, that could be queried by applications running on the platform. If that application's 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 the application's app store, it would learn which applications to list and allow to be downloaded, and which should simply be filtered and not shown to someone underage. 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.

Now, we would need - for that 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, which the browser would have received by an API from the underlying platform. But even that should be trivial. I mean, that would not take more than a day to define.

So as for the non-smartphone platforms such as Windows, macOS, Linux, and Steam, and of course Smart TVs, all of those platforms, at least the OS platforms, not - I don't know about Smart TVs, but I guess it could be added - operate with the concept of a "root" or an "admin," right, whose account should not be used as the daily driver, and daily users who work with far more safe and reduced privilege user accounts. So those platforms could easily arrange to add date of birth awareness to their user accounts, and then an API would be added to surface the brackets of those to the online requestor of that information. So basically, following exactly the model of the smartphone, that would give parents who wished to govern what their young children were able to do online, you know, what they saw and where they went, a clear and clean means for doing so.

And I so much favor date of birth over age because that automatically changes the bracket at the user's birthday, rather than needing to constantly update the age after a birthday. And of course parents could set whatever birth date they wanted. If they felt that their child was more mature than the typical child at that age, they could say that they were born earlier, which would then move them into a later bracket sooner.

So, you know, all of the online fury and indignation raging over the idea of California attempting to effectively outlaw, as I've seen online, any platform that doesn't provide these services, that all disappears when it's just an optional feature capability that a system's admin, you know, 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.

So I think it's also worth noting that this solution also nicely resolves the whole VPN backlash dilemma that is also beginning to appear. We're hearing the legislators saying, well, you know, those VPNs are being used to bypass, you know, the laws we've put in place, so we need to outlaw those. You know, a VPN in this case would be of no benefit, since the user's platform, not their IP address, would be producing the age bracket indication. So I thought that was an interesting take that Reason published, and I thought it was good that Newsom said, well, I'm going to sign this, but I hope we maybe make some modifications before it goes into law. And in any event, I don't think Linux people have anything to worry about. I mean, it is open source.

And we did see somebody kind of very quickly added the capability into user accounts on Linux, and it was - Linux was immediately forked without that provision in there, even though, I mean, it didn't even have an API. It didn't have a UI. It wasn't in any of the desktops. It was just down basically in the JSON structure they added a field for date of birth. And, you know, a lot of people in the Linux community freaked out over that. So it's like, okay, I mean, I get it. But again, there's no question, we need to have strong identity online. We've needed that for a long time. Everyone wants it to be anonymous. We're trying to hold onto that. Age-gating and age verification is coming to the Internet. And let's hope we do it in a responsible way.

Okay. So before we leave this discussion of age verification, I just wanted to note that Apple has started requiring age verification for their users in the UK and South Korea with the rollout of this latest 26.4 version of iOS. 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 asking that, like by looking for other signals like the length of time they've had an account, which would sort of automatically say, well, they've got to be an adult by now, then no other information is needed. But otherwise, they're going to get intrusive.

So, you know, as I said before, if this has to happen, I trust Apple more than any other third party to protect its users' privacy. Apple really does appear to be doing everything as right as they can. I've got links in the show notes for anyone who wants more information, like of an Apple support page that just says, if you are asked to verify your

age, we need to do that. I heard from a couple of our listeners who are in the UK that they had a variant of a driver's license which Apple software did not recognize. So they were having a problem with that.

So it's looking like, there's like on-phone, you know, image recognition of UK government-issued IDs and driver's licenses which Apple is able to ingest and then use to satisfy the UK's law. This is all, you know, it's not like Apple wants to be doing this. They definitely don't want to be doing this. Yeah, if you're asked to confirm that you're an adult, and then you proceed. So this is now happening, you know, by Apple, when they're operating in countries that require it.

**Leo:** It's selfish of me to say this, but I'm glad that they're testing this in UK and Korea so that we don't have to deal with it until the kinks are worked out a little bit.

**Steve:** Yeah, yeah.

**Leo:** Wow.

**Steve:** But it's, you know, it's coming.

**Leo:** Ultimately, this could be better; right? I mean, they're a choke point; right?

**Steve:** Yes.

**Leo:** So Apple and Google, because everything goes - and it should be for app stores, not for desktop operating systems, not for TV sets and things like that. But I can see why, with the app stores, you might want to have that kind of...

**Steve:** And also because a smart phone is inherently more of a personal use device.

**Leo:** Right.

**Steve:** You know, kids have their, the fact that we use the pronoun "their" smartphone, says that they're bound to it. It's their social media that is on that phone, and, you know, their use of the phone. You know, they take it into their bedrooms with them. So it makes sense that the parents could work with Apple to establish a secret date of birth, and the phone only divulges brackets when necessary. So I think that's where we're going to end up being. And again, if anybody had to be doing this, I would trust Apple, much as you and I both, Leo, are increasingly annoyed by some of what Apple is doing. As I said, I've lost - I don't even know how my photo app works anymore on my phone.

**Leo:** Yeah, I know.

**Steve:** I get into some strange mode where I can't get rid of - half the screen is information about the photo. And I try, like, to push it down. It won't go away. It's like, what happened?

Anyway, 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 that we are seeing everywhere of politicians and technology. The Risky Business Newsletter covered this puzzler by writing: "The Russian government is working on a law that would require all mobile operators [in Russia, all Russian mobile operators], to use a custom, domestically developed encryption algorithm for the country's 5G mobile network. If the bill passes" - and it's expected to - "all phones sold in Russia going forward will need to support the NEA-7 algorithm" - apparently because 1 through 6 were no good - "or they will not be able to connect to Russian mobile networks which, by 2032, will only support NEA-7.

"Foreign algorithms such as SNOW (used in Europe), AES (used in the U.S.), 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 regulation 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" - which is so stupid - "of Ukrainian drones and missiles" - stupid, you'll see why in a second - "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 existing mobile network, since they can always fall back to older protocols" - not the 5G protocols - "LTE and 3G," both of which will continue to function.

"On the other hand," writes Risky Business, "the proposed law represents a 'patriotic' legislative flex. It's the type of unrealistic stuff," they wrote, "that's been happening in the Russia Duma recently, to show that Russia is important and still matters on the global stage." That's right. Show that by withdrawing from the rest of the world. Wow. "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." Why would they?

"There's 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." Because it's going to have to design and implement all of the base tower technology, too, to add NEA-7 support. They said: "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." Okay, I suppose that's possible.

"On Ukraine's side, the answer is likely to be the same as with Russia's rollout of MAX." Remember that was the Russian-only messaging system, and Russia demanded that everybody use it, and then began cutting off access to all the others, as they are now. "The Ukrainian intelligence services were delighted that Russia was mandating everyone in their country use an incredibly insecure and easy-to-hack mobile app." Meaning MAX. "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."

So I think, as I look at this and think of what like Russia's doing, it occurs to me that one of the most important lessons taught by the Industrial Revolution is the incredible power that comes from standardization. For example, if every country had their own screw thread standard, then 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.

**Leo:** Weren't train gauges incompatible for a long time?

**Steve:** Perfect, another great example, yes, yes.

**Leo:** A little hard to go from country to country.

**Steve:** Right. And Leo, think about it. As it is, we do have separate metric and imperial threading, and look what a mess that creates. Just...

**Leo:** Yup, have to have both sockets, yup.

**Steve:** Yes. Just that. So, you know, another example of standards failing when they differ is, as our Picture of the Week showed, AC outlet plugs around the world. Although I'll admit that they have provided a terrific supply of Pictures of the Week for this podcast. But that further demonstrates the failure; right? So my point is that the standards that the world has agreed to, the standards surrounding the Internet, Ethernet, and USB all being examples, they've resulted in astonishing economies thanks to their interchangeability and interconnectivity, which we get free of charge, simply by choosing not to roll your own.

So I think this clearly demonstrates the insanity of what Mother Russia is choosing to do. After 2032, five years from now, Russian citizens will likely be stuck with Russian-made Android smartphones with godawful hardware and no choice in the matter. If they want 5G, they've got to use their Russki phone or, you know...

**Leo:** Russki phone.

**Steve:** That's right.

**Leo:** In Soviet Union, Russki phone call you.

**Steve:** And that's not progress. Wow.

**Leo:** That's hysterical, wow. Some people drive on the left side of the road. Some people drive on the right side of the road.

**Steve:** And oh, and boy, have someone drive you if you're in one of those countries because every instinct you have is wrong.

**Leo:** Merging on the freeway is really tough.

**Steve:** Oh, oh.

**Leo:** For me anyway, yeah.

**Steve:** Okay. So speaking of Russia, it seems that Russian intelligence services somehow arranged to install some spying hardware into a Ukrainian drone factory. Now, the device was embedded inside a thermostat, but it did much more than control the room's temperature since it included a camera, a microphone, and a little router. The story becomes even 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 which they received from Ukrainian intelligence services. So the Russian surveillance device was installed, after which TechEx worked with Ukrainian intelligence to supply a constant stream of disinformation, which was regarded as highly trusted because the Russian spies were certain that nobody knew about it. So why would they be making stuff up in front of the thermostat? Whoops.

**Leo:** That's smart, yeah.

**Steve:** Love it. Okay. So last year, as we recall, we had some fun looking at a very clear deconstruction of the claims being made about quantum factorization. Remember that, you 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 crypto. Last year it appeared that the world had a much longer way to go than was assumed because that takedown of all of the progress that was being claimed which we examined carefully convincingly revealed that not a little bit but a lot of sleight of hand had been going on behind the scenes with the use of, for example, highly contrived factorization targets.

Now, Google appears to disagree. Or perhaps they're just taking the "better to be safe than sorry" approach. The news of last week is that Google has moved what they called the so-called "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 this decade. Okay. They've introduced a 2029 timeline to secure their products, that is, as their deadline to finish securing their products with post-quantum crypto (PQC) protections. Both Chrome and Google Cloud already have PQC (post-quantum crypto) protections in place, and Android is getting them later this year. We also know that Apple and Signal have both already added post-quantum crypto to their messaging platforms.

In addition, Cloudflare, AWS, Azure, Meta, and Zoom all have PQC in place today. Plus TLS v1.3, the current and latest version of TLS, is already capable of negotiating post-quantum crypto encrypted connections, and Cloudflare tells us that more than half of all the traffic moving through Cloudflare is now already quantum safe. So hats off, you know, we've been covering this move and the need to move toward quantum safety for years now, with the cryptographers getting to work on post-quantum algorithms way before it seemed that we had a problem.

I still think it's way before we have a problem, even now, based on the real evidence that we've seen. But, hey, our chips have the power, our processors have the power, no reason not to do dual crypto schemes where you encrypt with both a pre- and a post-quantum crypto to be safe. And in that case I don't know what the NSA is going to do with all that data that they've been sucking down, Leo. It's got, I mean, I guess historically the older pre- post-quantum crypto communications they could decrypt, if it's still of any value.

**Leo:** It's going to be old, yeah.

**Steve:** Yeah, really old.

**Leo:** Do you really think that quantum's going to happen by 2039?

**Steve:** No. I do not. Well, Google is saying 2029.

**Leo:** I'm sorry, 2029, yeah.

**Steve:** Yeah. I just don't see - to me it doesn't look like we're even close. And it's not as if you're able to break down the prime factorization problem into smaller pieces. If you could, we would have.

**Leo:** Right.

**Steve:** You know, we would have already decomposed it into something that classic computers can solve. It's intractable right now. So, you know, if they're jumping up and down about factoring 31, and then we find out they cheated, it's like, okay. I have a hard time getting worked up about this. You know, I might get surprised. Okay.

**Leo:** It's prudent to have post-quantum crypto available.

**Steve:** Why not?

**Leo:** Don't see any reason not to.

**Steve:** Why not, exactly. It doesn't cost us anything at this point. Our chips are fast enough. We've got the algorithms. And we're not just using them. We're using both. So if a problem is found in either one, the other one protects us.

**Leo:** Right.

**Steve:** So why not?

**Leo:** Right.

**Steve:** Okay. So last Tuesday, during the annual RSA security conference, where you and Lisa were present to hobnob with many of the podcast network's supporters, the CEO of the UK's NCSC, which is the UK's cybersecurity agency, spoke to the conference. The publication The Record wrote about his presentation. What they said was: "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, they wrote, "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,' he said.

"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 followed 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 software-as-a-service (SaaS) platforms.

"Horne said during his speech: '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 itself 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." And I'm going to be talking about that as soon as I finish with this because I think this is really interesting.

They said: "Describing the February sell-off as 'a billion-dollar wobble' - and referencing the term 'SaaSocalypse'" yes, the 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." We've talked about this, all this outsourcing that is now being done.

"In its blog post on Tuesday" - that's a week ago - "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 original 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 article finishes, saying: "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 cannot 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."

Okay. So there are a number of interesting takeaways here, I think. First is the obvious-when-you-look-at-it threat that vibe-coded replacements for software-as-a-service represent. Okay, think about it. Why would any large enterprise rent, under an expensive recurring subscription, what a handful of their in-house coders could whip up overnight using the benefit of vibe-coding AI to create bespoke software that more perfectly fits their needs? I hadn't really 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 homegrown in-house-coded alternatives that vibe coding now makes so easy to create.

You know, remember we heard that C-suite executives saying: "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 outsource 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?" No more subscription fees. So it does seem pretty clear that this is going to be an accelerating trend in the future.

But the other shoe to drop was this NCSC CEO's primary concern, which was that the threat that carefully created, refined, and secured SaaS solutions, which is what we have today from third parties who wrote these things 10 years ago, carefully and with human programmers and coders, and have since worked all the bugs out, but they're not free, that they 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? Well, because they appear to work and solve a problem, and the price is right. It's zero.

But the truth is that everyone is just holding their breath and hoping for the best. Right? Hoping that this library they pulled isn't malware. No, doesn't seem to be. Nobody else says it is. So, okay. But that's not the way security is obtained and maintained. On the other hand, it doesn't cost anything. It's free.

It's going to be very interesting, Leo, to see what happens as enterprises develop, for in-house use, the various systems that they've been outsourcing. Because you know they're going to. And I overall think it's probably going to be a win, but I imagine there will be a few stumbles along the way.

**Leo:** It's not like, you know, the SaaS software you buy from these big companies is necessarily secure, robust, and reliable. We talk about all the problems they have all the time on this show. So...

**Steve:** That's true.

**Leo:** You're trusting somebody, unless you write it yourself. And nobody can afford to write it themselves. Doing it in-house is really hard. I think, you know.

**Steve:** Well, yes. But vibe offers the opportunity of a bunch of programmers saying, okay, Claude, here's what we need.

**Leo:** Right.

**Steve:** We need a customer relations management system, and it needs to take our database, and here's the schema, and we want to have this UI, and blah blah blah. And presto bango, you've got an app. I mean, that's Claude.

**Leo:** Yeah. I've been very tempted to write a sales system for TWiT. We had it, it was written by a low-level employee many years ago, in .NET. And he knew what he was doing, I guess. But when he left he said, "I'm not maintaining it, so you're on your own." So, and it has little bugs. Like two people can't use it at the same time or it crashes, and you have to have a hard reboot and stuff. So...

**Steve:** And that's often the case with, like, with your typical bespoke in-house software.

**Leo:** Somebody wrote it, yeah.

**Steve:** Yup. It got written. And we were using, well, actually we only just retired, we call it Dino Database, I don't know why...

**Leo:** You probably wrote it yourself, though; right?

**Steve:** Actually it was the only coder who ever wrote any code that we actually used, a brilliant guy named Steve Rank, yeah. And he wrote it in Dbase 2. Which we then moved to...

**Leo:** FoxPro probably.

**Steve:** ...FoxPro, exactly. And Sue, as recently as until the release of 6.1, would look old customers up on, you know...

**Leo:** FoxPro.

**Steve:** Yes, it worked great.

**Leo:** Sure. I wrote a lot of Dbase 2 software in my time for the radio station I worked at.

**Steve:** Yup.

**Leo:** Yeah, no. And, you know, that's not even really coding because it's just a database, and you're writing a front end to a database really. But...

**Steve:** Yeah, yeah.

**Leo:** I don't know. I'm...

**Steve:** Go ahead.

**Leo:** I'm very bullish on what Claude Code can do. But obviously, you know, it may introduce errors. And then, yeah, pulling these libraries is nowadays really risky. There are solutions, though. People have found solutions. One guy said, well, look, just pin the version until - say you can't download it until it's been out for a week.

**Steve:** Yes.

**Leo:** And presumably somebody will have caught on by then; right?

**Steve:** Yes. And that was a problem with LiteLLM is it was not pinned.

**Leo:** Right.

**Steve:** And so everybody grabbed the link. Let's take a break, and then we're going to look at an update on the ClickFix campaigns.

**Leo:** Okay.

**Steve:** Because that's still bad.

**Leo:** Well, yeah, I'm curious what you think about Apple's kind of sort of solution was. Which I thought was interesting. Steve?

**Steve:** So 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, they said: "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." Which I think perfectly sums up, like describes the nature of this.

Okay. 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 of 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." So basically this is their, you know, cyber forensics system.

They said: "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 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," you know, high return on investment 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 groups 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."

Okay. Well. We all know how annoyed I am with Microsoft. This entirely preventable, you know, detectable and preventable vulnerability is now three years old, and its use has been accelerating rapidly to the point that this family of readily blocked exploits, as we learned a few weeks ago, now accounts for more than half of all security breaches.

**Leo:** Holy...

**Steve:** Just one technique.

**Leo:** Holy cow.

**Steve:** More than half.

---

**Leo:** That's how effective it is.

**Steve:** It is that effective, exactly. Everybody is going to fall for it unless they have some savvy. And it's like, wait a minute. Why am I to confirm that I'm not a, you know, to confirm that I'm human, why am I opening the Windows Run and pasting this string into and then hitting enter?

**Leo:** So bad.

**Steve:** But again, most people are just script followers. I mean, most Windows users don't really know how Windows works. Right? I mean, I hear Paul saying the same thing. So by comparison, our listener Jeff Adamson sent a note Friday with a link to a story over at [apple.gadgethacks.com](http://apple.gadgethacks.com) with the headline: "macOS 26.4 adds Terminal paste prompt to block pastejacking."

**Leo:** Do you think this is specifically aimed at Clickjack?

**Steve:** Yes, it is.

**Leo:** It is, yeah.

**Steve:** It is exactly aimed at it. And so they called it Pastejacking, they wrote, well, Pastejacking is the term used in the headline. That's 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: "Possible malware, Paste blocked." And it says: "Your Mac has not been harmed. Scammers often encourage pasting text into Terminal to try and harm your Mac or compromise your privacy. These instructions are commonly offered via websites, chat agents, apps, files, or a phone call."

And then you've got two options: Don't Paste is what's highlighted and recommended, or Paste Anyway. So that's, you know, a nice, like stop sign comes up and says, whoa, no, don't just follow these instructions. Let's think about this for a second.

So, you know, I don't suppose that Windows 10 users, who still compose one quarter of the Windows desktop population, will ever see Windows' behavior change; right? Microsoft has moved on. But it would sure be nice if Windows 11 users could have this simple exploit prevented by Microsoft caring, which is all it takes, a little bit of care from Microsoft, as Apple has just demonstrated they do because this is so easy to fix.

I also wanted to highlight the tips near the end of the Recorded Future article that talked about available mitigations for this under Windows. The measures of disabling the Windows Run dialog box via Group Policy Objects and implementing PowerShell Constrained Language Mode, you know, that won't help the general Windows population because they're just using Windows at home. But within any enterprise, I would jump on both of those immediately. You know, unless an enterprise IT staff know that the Windows Run dialog box is needed, and I'm not sure why it would be, disable it.

The good news is you can turn it off for all of your Windows users inside an enterprise and immediately prevent this most easy solution. And then also constrain what you can

do with PowerShell so that it's basically neutered. I mean, what we see here, the problem is that, over time, just like has happened with our iPhones, Windows has gotten incredibly complicated. I mean, it always - it still has everything it ever had, and they just keep adding more stuff. And most users just want to run an app to open Word, or open email, or run, you know - they don't want/need all this other crap and don't - they don't know what it is.

**Leo:** Right.

**Steve:** And it is all dangerous. I don't know, Leo. 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 measures moving forward. And the Reddit users are not happy. PC Mag provided the details under their headline "Reddit Could Soon Require Face ID to Prove You're Not a Bot." They wrote: "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." Okay, so that's more frequent than one out of every seven Reddit posts.

They wrote: "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 TBPN podcast, first spotted by Engadget, Reddit CEO Steve Huffman said this tech is the most 'lightweight way'" - that's his quote - "'to ensure all users are human. Huffman indicated the platform may use decentralized third-party information providers" - oh, boy - "to verify users' personal details."

We've recently been talking about all the uses to which residential proxies could be put. Bouncing AI-bot traffic through such residential proxies makes detecting and blocking them based upon IP address impossible. You just look like any random user, like spread around the globe. So, yeah, some sort of logon-time verification is needed. But we all know the potential downside of using any third-party identity verification system.

PC Mag continues, writing: "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," they write, "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 of 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..."

**Leo:** Well, I don't know when that interview was, but Alexis Ohanian and Kevin Rose recreated Digg.

**Steve:** Yeah?

**Leo:** And had to shut it down last week.

**Steve:** Yup. I know.

**Leo:** Because of the bots. Oh, my god. It's a nightmare out there.

**Steve:** It really, really is. And Leo, when you add AI to the mix, and hundreds of thousands of residential proxies that the AI bots are able to bounce their traffic through, you cannot detect them.

**Leo:** Yeah.

**Steve:** I mean, we have an undetectable bot problem.

**Leo:** Yeah.

**Steve:** So many Reddit users have already expressed grievances with the move, with one user saying: "Tell me you want to kill Reddit without telling me." Meaning this kills Reddit if you start requiring people to deanonymize themselves.

**Leo:** Yeah.

**Steve:** So the article finishes, saying: "Reddit would not 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." And as we know, it wasn't Discord's AI tool. They farmed it out, and those people got hacked. And 70,000 some personal private information got loose. So, I mean, Leo, there's no solution.

**Leo:** Yeah, I'm sympathetic. I don't know what these guys are going to do. I mean, I think there's a lot of - when I'm on Reddit, half the time I see a host, somebody will say, "That's AI. Stop using AI. You're using AI." And I don't know if it's obvious that it's AI or not. If you use bullet points in a post, "AI." If you use certain words, "AI." And I don't know if that's true or not. I don't know how you would know.

**Steve:** And it may once have been. But AI is a moving target.

**Leo:** Right.

**Steve:** I mean, if it's doing something that is getting it called out as AI, it's going to change its behavior.

**Leo:** Right. And I use bullet points and dashes. And occasionally I'll use the word "delve." That doesn't mean I'm AI. So, I mean, the problem is, as AI gets better and better, it looks more and more like average content. That's the whole thing.

**Steve:** This is a problem that has no solution.

**Leo:** Yeah.

**Steve:** And I don't say that often. I mean, I spent seven years devising a solution for online identity authentication because I thought there was one. You know? SQRL was that. But I don't see a solution here. I do not see a solution.

**Leo:** And you're pretty ingenious when it comes to that stuff.

**Steve:** And that's my point is I'd be like saying, well, you know, we could do this or that. No. I don't see a solution.

**Leo:** Somebody fed the Declaration of Independence to an AI detector, and it said, well, about 93% chance that's AI written.

**Steve:** I saw that. Now, wait. Wasn't that 1776? I don't think that we really...

**Leo:** We didn't have AI back then. But that Thomas Jefferson, I don't - actually they said it was 98%, 98% AI generated.

**Steve:** Certain that it was...

**Leo:** Yeah.

**Steve:** Oh.

**Leo:** Yeah. And this is the problem is that these AI detectors aren't really any good; you know?

**Steve:** Right.

**Leo:** That's - they don't recant Detect AI. And I think a lot of people assume that something's AI when it's not. And on Reddit, I mean, how do you know it's one in seven? It could be one in two, or it could be one in a thousand. It's just - you don't know. And I think it's way to draconian to say, okay, well, from now on everybody has to give us a driver's license before they post. That will kill Reddit.

**Steve:** Yes.

**Leo:** A lot of what Reddit's all about is anonymity.

**Steve:** Yeah.

**Leo:** Not for any nefarious purpose.

**Steve:** No, it's just that's what people want on the Internet. They want to be able to say what they want to say.

**Leo:** Yes.

**Steve:** Without being held personally responsible.

**Leo:** I'll give you a completely innocuous example. I hope I'm not - I think she said this on the show. Paris Martineau is a fan of reality TV shows. And she moderates a reality TV show subreddit. But she doesn't want that to be in her real name. That's a guilty secret. She should be able to do that privately, without revealing that, you know, I am she. You know, I should be able to, you know, say that I like, you know, leather boots without having to admit it in public. Wait a minute. I mean, I didn't mean to say that. That was a mistake. Do you want to take a break right now, Steve?

**Steve:** Yup. And then we're going to look at LiteLLM and what happened last week.

**Leo:** Oh, can't wait to hear about this.

**Steve:** The bullet we dodged. Okay. We're going to look at LiteLLM, and we will take our last break here before we charge this. So, okay. So let's start by answering the question "What is LiteLLM and why would we want it?" It was backed initially by Ycombinator, and the LiteLLM page over at Ycombinator describes their project. They said: "LiteLLM is an open-source LLM Gateway with 18K-plus stars on GitHub." Now that's over 41,300. So, yeah, it's very popular. And they wrote: "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-plus" - more than a hundred - "LLM APIs (Bedrock, Azure, OpenAI, Vertex AI, Cohere, Anthropic, and on and on and on) in the OpenAI format." They said: "We've raised a \$1.6 million seed round from Ycombinator, Gravity Fund, and Pioneer Fund."

Over at GitHub, the "About" paragraph for LiteLLM says: "Python SDK, Proxy Server (AI Gateway) to call more than 100 LLM APIs in OpenAI (or native) format, with cost tracking, guardrails, load balancing, and logging." And then they enumerate some - Bedrock, Azure, OpenAI, Vertex AI, Cohere, Anthropic, SageMaker, HuggingFace, vLLM, NVIDIA NIM.

Okay. So the LiteLLM site itself largely echoes this and highlights a couple of testimonials. It quotes David Leen, a Netflix staff software engineer, who says of 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 over at Lemonade says: "Our experience with LiteLLM and Langfuse at Lemonade has been outstanding. LiteLLM streamlines the complexities of managing multiple LLM models."

Okay. So I think everybody 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 and greatest being tomorrow's "you're not still using that, 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 going to eventually wind up being. Essentially, with LLMs being the hottest fungible commercially tantalizing mystery that humankind has ever created, the last thing anyone wants to be is locked into yesterday's less-glamorous, now it's underperforming or it's overpriced, model.

So to their credit, the guys at LiteLLM, the guys who created this idea, they were very quick to see a need and an opportunity. They created what is essentially a universal large language model API translator that allows frontend developers to code to a single fixed model, the one originally developed by OpenAI by default, 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 frontend.

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. 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."

**Leo:** Is that the ice cream truck?

**Steve:** No.

**Leo:** Is that Lorrie?

**Steve:** That is my lovely wife who has forgotten that I'm in the middle of a podcast right now, and she just put her hands over...

**Leo:** Hi, Lorrie.

**Steve:** Oh, she hung up. So anyway, they said: "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." So, you know, basically it's a big, you know, switching hub that decouples what you're doing on the application end using a large language model from whichever large language model you want to use. So it's kind of a

no-brainer; right? Like why would you not want to use this? They've been working on it since the winter of 2023.

And, as you might imagine, the challenge - I don't want the job - of supporting an exploding number of individual varying and evolving AI LLMs, each with their own API, requires a great deal of never-ending work. They're hiring, by the way. But that's the path these guys have taken, and until recently things have been pretty smooth sailing.

So what happened last week? Okay, let's start with TechCrunch's overview, and then we'll dig a bit deeper. TechCrunch wrote: "This week, some really atrocious malware was discovered in an open source project developed by Ycombinator graduate LiteLLM. LiteLLM gives developers easy access to hundreds of AI models," blah blah blah. "It's a breakout hit," writes TechCrunch, "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 40,000 stars on GitHub and thousands of forks.

"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 itself relied upon. It then stole the login credentials of everything it touched. With those credentials" - and this is, as Leo, you know, your point is we don't yet really know how much damage was done. TechCrunch said: "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." As you said, Leo. "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."

Okay. So last Tuesday, as mentioned by TechCrunch, this developer, 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, and it's exactly what we would expect. He said: "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." In other words, current LLMs are just fungible.

So here's what happened. Callum's posting was titled "No Prompt Injection Required," where he's, you know, kind of tongue-in-cheek. He wrote: "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 tens of seconds to load. The CPU was pegged at 100%, all signs I'll be working on my local environment for a time," meaning things got messed up. "After failing to software reset my Mac, I took a final picture for evidence and then hard reset it."

**Leo:** Wow.

**Steve:** So he said: "So far, the clues had been Cursor asking me for network access right as the machine was freezing up. The process list showed a bunch of Python commands all exec-ing a Base64 encoded string..."

**Leo:** Oh, that's not good.

**Steve:** Uh-huh, yeah, "...and 11,000 processes running." He said: "I set ulimit to 16K for machine learning workloads, so this was partly expected." In other words, he has his system configured to allow 16,000 different processes. But he had 11,000 running for no apparent reason at that moment.

He said: "On restart, I asked Claude to investigate. After going down a rabbit-hole on the wrong shutdown due to my forced shutdown" - meaning that Claude started to look at something different because he had done a forced shutdown...

**Leo:** There were two, yeah.

**Steve:** Yes.

**Leo:** And they crashed each other.

**Steve:** Exactly. "Not generating the expected logs," he said, "I presented it with the start of the Base64 string. Just enough to decode 'import subprocess \n import temp file' 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." So he's crediting Claude with helping him, you know, forensically diagnose what it is that happened to him.

He said: "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."

**Leo:** By the way, that's PyPI. I just want to make that - there is something called PyPy.

**Steve:** Oh, okay, yes. PyPI, good, thank you.

**Leo:** That's a library, yeah.

**Steve:** He said: "So, what actually happened?" Okay. So, okay, I'll just interrupt here to note that Callum is about to start referring to MCPs, which is the ModelContextProtocol. The MCP site, the Model Context Protocol site, explains: "It's an open-source standard for

connecting AI applications to external systems. Using MCP, AI applications like Claude or ChatGPT can connect to data sources (local files, databases), tools like search engines and calculators, and workflows, you know, using specialized prompts, which enable them to access key information and perform tasks." And they said: "Think of MCP like a USB-C port for AI applications." Again, another standardization which is so very powerful. They said: "Just as USB-C provides a standardized way to connect electronic devices, MCP provides a standardized way to connect AI applications to external systems."

Okay. So armed with only that much understanding, what Callum explains can make sense, and it's not necessary for us to deeply understand it more. Callum says: "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" - again, because it was unpinned. It wasn't saying I want this version. It was saying give me the latest. Which," he writes, "was malware uploaded to PyPI by hackers just minutes earlier." Minutes earlier. "The seamless ergonomics of `uvx` meant I became one of the lucky beta testers of the freshly released malware."

**Leo:** Congratulations.

**Steve:** 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 like the Log4j was. It was a working piece of tooling, the LiteLLM package. And by being "unpinned," Callum's dependent packages were not saying "we want this exact version." So the default behavior was to grab a copy of the current one. And in this case that "latest and greatest" had been deliberately compromised by bad guys.

Callum continues, saying - this is great, too: "A sloppy, likely vibe-coded mistake in the actual malware implementation led it to turn into [what he called] a fork bomb. It installs a file called `litellm_init.pth` into the site-packages directory. 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, which spawns another, which spawns another. Thus leading to the only sign I would have noticed that the malware was running."

That's where those 11,000 instances came. And the reason his crashed is it got into an infinite loop of spawning these `litellm_init.pth` processes, and the system crashed. 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. So we have to ask ourselves, what if the author of this malware had not made that mistake?

So what's the takeaway? So he writes: "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 file system 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 versus remote MCP servers, and in that case you still need to do what you can to mitigate this risk."

He finishes, saying: "I don't think there is anything new to say here. It's the same thing we've 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 again." He said: "We analyzed the blast radius of this attack. 47,000 downloads in 46 minutes, 88% of dependent packages unprotected."

So Leo, let's take our final break, and then we will continue looking at a little more of the forensics of this mess.

**Leo:** Yeah, wow. This is amazing. You know, I saw Andrej Karpathy's tweet almost instantly, thank goodness, and immediately went to Claude and said, hey, is there any LiteLLM anywhere in my system? And it said no. I mean, you have the name in your package list, but you never downloaded it, so you're okay.

**Steve:** Whoooooo.

**Leo:** I know, it's terrifying. It was terrifying. Now let's get back to Steve Gibson and a further dissection. By the way, I really appreciated Callum's write-up. It was a very good write-up. He did it very quickly, got the word out to the community. Forty-six minutes after he discovered it, it was taken down, which was thank goodness because that...

**Steve:** Yeah. And I also thought, you know, he noted that Claude wrote the email for him. So I realized that speed of action is one of the things that we get from AI also.

**Leo:** Oh, absolutely. It's a lever. It's a tool. And used properly, it really adds to the power of what you can do. It also adds to the power of what bad guys can do. That's the double-edged sword of all this. All right. On we go.

**Steve:** Let's take a closer look at the malware itself. For that, we turn to Trend Micro, 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" - those are the bad guys - "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."

So they led their coverage with three key takeaways: They said: "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."

Second point: "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." In this case a little too rapidly. And third: "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 we learned from Trend Micro. They explained: "On March 24th, production systems running LiteLLM" - and that's exactly last week, last Tuesday. "Production systems running LiteLLM started dying," as happened to Callum, "and engineers saw runaway processes, CPU pegged at 100%, containers killed by out-of-memory 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." And just to pause, just think of if this had not been caught: 3.4 million instances downloaded per day would have been infected with this nasty malware. I mean, this is bad malware.

They wrote: "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 Open VSX in a single coordinated operation. While it did not specifically target AI infrastructure, the campaign's cascade through the developer toolkit 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" - and I'm not going to share all the details because we don't need that. But they wrote: "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."

Okay. So that's much deeper than we need to dive for all that. But what they uncovered and reported about the root source of the vulnerability was interesting. Under their "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, file systems, 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."

Now, so, okay, the point is Trivy was the root of this compromise. So they explain: "Security scanners are uniquely dangerous supply chain targets. By design, they require broad read access into 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 MegaGame 10418 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 1st 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."

Okay. Now, that's an important point, so I want to pause here to explain that. 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, of course, being collections of atoms, are divisible. Not so the atom. So 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 inside itself at once, that would be easy to do. The program would read the value of the thing that's being counted. If it was not already at its upper count limit, then the program would increment it to its next value. If it was already at the upper limit, it would just leave it there.

But now imagine what happens if there's a lot more going on in the program with multiple simultaneous execution threads running around, perhaps because the CPU has multiple cores, or the application itself has many threads running. In this environment, there's a chance that both CPUs would wish to increase the count at the same instant. So they would both be executing the exact 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, that increase by two would move it up past the limit. A very subtle bug.

These sorts of so-called "race conditions" have historically been the source of many hard-to-find problems. You know, they're the sort that never happen while you're watching it, while you're developing the code; but they somehow always occur when you're onstage demonstrating what it is that you've got.

So in our example, that "test the value and maybe increment it," that would need to be made "atomic" so that the testing and the incrementing could not be broken apart and performed separately, even by different processors that are executing at the same time. That operation could only be done by one processor or execution thread at a time. So the other one, the other processor trying to do it would be briefly stalled until the first processor had finished with that atomic operation. And at that point the second processor could proceed. And if it saw that the variable was already at its limit, it would not also increment it.

Okay, so we left off with Trend Micro noting: "Aqua Security disclosed the incident on March 1st and initiated credential rotation. However, according to Aqua's own post-incident analysis, the rotation 'was not atomic, and attackers may have been privy to refreshed tokens.'" In other words, somebody might have still been logged in when a token was updated, and then they would have grabbed that.

Trend Micro then continues: "The gap," that is, this race condition gap, that gap "proved decisive. On March 19th 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," whatever those details mean, but it meant "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). According to analysis by CrowdStrike, the

legitimate Trivy scan still ran afterward, producing normal output, leaving no visible indication of compromise."

Okay. In other words, because Aqua Security was, for whatever reason, 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. Trend Micro finished this portion of their write-up by writing: "This is the meta-attack: a security scanner the tool defenders rely on to catch supply chain compromise itself 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. And LiteLLM was just a coincidental casualty of this."

They said: "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." And what we're now seeing is the bad guys have gotten sophisticated enough to take advantage of that. I mean, it is truly terrifying.

So what we see is that the enabling of this attack on LiteLLM had nothing to do with AI per se. It's just its popularity that allowed it to - that would have allowed it to explode at 3.4 million instances of compromise per day had the bad guys not made that crucial mistake that crashed the machines that it was trying to compromise. So after providing a fully detailed forensic analysis of this malware campaign, Trend Micro 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." So 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."

Trend Micro continued, saying: "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 those 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. They said: "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 very highly skilled, this TeamPCP, these attackers appear to have just been in a bit of a hurry. This led them to deploy otherwise very potent and sophisticated malware that must have taken a lot of time to generate, allowed them to deploy it containing a flaw that unfortunately for them, and thank god for us, immediately caused that malware to draw attention to itself.

---

**Leo:** This is why you want to test before you post to production, man. They should have known.

**Steve:** Yup, they got bit by their own...

**Leo:** Race condition.

**Steve:** You know, rush, yeah, by being in a race. And 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 instances of the infected LiteLLM per day being used, 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, as I said at the top of the show, we have dodged another bullet.

**Leo:** Oh, yeah.

**Steve:** There can be no question that the entire industry has built an ecosystem upon which it has become dependent, if you'll pardon the pun, or double entendre, I guess, whose security guarantees are truly fragile. These are fragile guarantees. We're essentially hoping for the best because the goodies are just too enticing for us to resist. Or 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 direct control. And all we can really do at this point is hope that our luck holds.

**Leo:** It's not just the packages. It's also automation. I mean, it sounds like it was a CI/CD issue.

**Steve:** Yes, yes.

**Leo:** And by the way, this happened a couple of weeks ago, a GitHub CI/CD issue with GitHub actions.

**Steve:** Yup.

**Leo:** And I keep seeing this again and again. And so that's a case of - and I understand why. CI/CD is incredibly useful. It's an automated way of building and delivering your software. That's what it stands for, continuous integration, continuous delivery.

**Steve:** Continuous development, yup.

**Leo:** Development. But if you're automating to that degree, and you're not paying attention, there's some real risk involved.

**Steve:** Well, and AI is bringing us another layer of automation.

**Leo:** Right.

**Steve:** I mean, it's - people are just stunned by what it does for them, like they don't even understand what the installation on their local system is. It's just like...

**Leo:** Well, you know, Claude will automatically push to GitHub, so all my repos are pushed to GitHub. And it was automatically building it. It was setting up GitHub actions and building the software. I wasn't even - I didn't even know how to do that. It just did it for me. And so, yes, we're kind of giving over a lot of our agency to systems, partly because it's so complicated these days.

**Steve:** That's true. We've created a super complicated system.

**Leo:** Which was a security tool.

**Steve:** Scanner, scanner. It was an open source scanner used by these systems to scan themselves for malware. And it itself was compromised.

**Leo:** It was compromised. And then it pushed a bad version of LiteLLM?

**Steve:** Because LiteLLM used it to scan for malware.

**Leo:** Right. And so in the...

**Steve:** And so it had access to...

**Leo:** So in the GitHub actions, the Trivy keys were compromised. They rotated them as best they could, but they didn't get them all apparently.

**Steve:** Yup, yup.

**Leo:** Bad guys, this PCP team got the key and then used that key to compromise Trivy to inject malware into LiteLLM as part of the CI/CD process.

**Steve:** Pipeline; right.

**Leo:** Wow. That's actually pretty sophisticated.

**Steve:** Oh, no, as Trend said, these guys really know what they're doing. I mean, this is a, you know, unfortunately the goodies are so big, I mean, they know how many copies of stuff are being downloaded per day.

**Leo:** This stuff exfiltrated tokens, SSH keys, crypto passwords...

**Steve:** Kubernetes...

**Leo:** Kubernetes keys. It basically took all the secrets on your system and sent them to the bad guys.

**Steve:** Encrypted them and sent them off to a spoofed domain.

**Leo:** I mean, imagine if this had gone for a day or two, the night - I mean, I'm surprised we haven't heard more pain from the 47,000 who installed it. And I understand now why they were in a hurry, because they had a compromised key to Trivy, but they didn't know how long that key would stay good.

**Steve:** Yeah.

**Leo:** So they said we've got to strike while the iron's hot. Quick, get some malware out there.

**Steve:** I mean, and it literally must not have been tested.

**Leo:** No.

**Steve:** Because apparently immediately when you ran it, it crashed your system. I think that explains why the 47,000 instances we haven't heard anything from, I mean...

**Leo:** Oh, because everybody crashed. It didn't work.

**Steve:** It came out of the gate and just stumbled.

**Leo:** Yeah. Didn't work, yeah.

**Steve:** Yeah.

**Leo:** Because they rushed. They said, quick, we've got, you know, we've got one minute to take advantage of this. Let's push something out. And they probably told Claude, write something real quick, get on a system, encrypt the keys, and send them to this address. Wow. Ay ay ay ay ay, caramba. Well, I'm glad, you know it's

really good, thank you. Callum's write-up was great, but I didn't understand the Trivy part of it. So thank you for explaining that Trend Micro report. This is why we listen.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details:  
<http://creativecommons.org/licenses/by-nc-sa/2.5/>